CS405PC: JAVA PROGRAMMING

B.TECH II Year II Sem. **L T P C** 3 1 0 4

Course Objectives:

- To introduce the object-oriented programming concepts.
- To understand object-oriented programming concepts, and apply them in solving problems.
- To introduce the principles of inheritance and polymorphism; and demonstrate how they relateto the design of abstract classes
- To introduce the implementation of packages and interfaces
- To introduce the concepts of exception handling and multithreading.
- To introduce the design of Graphical User Interface using applets and swing controls.

Course Outcomes:

- Able to solve real world problems using OOP techniques.
- Able to understand the use of abstract classes.
- Able to solve problems using java collection framework and I/o classes.
- Able to develop multithreaded applications with synchronization.
- Able to develop applets for web applications.
- Able to design GUI based applications

UNIT - I

Object-Oriented Thinking- A way of viewing world – Agents and Communities, messages and methods, Responsibilities, Classes and Instances, Class Hierarchies- Inheritance, Method binding, Overriding and Exceptions, Summary of Object-Oriented concepts. Java buzzwords, An Overview of Java, Data types, Variables and Arrays, operators, expressions, control statements, Introducing classes, Methods and Classes, String handling.

Inheritance– Inheritance concept, Inheritance basics, Member access, Constructors, Creating Multilevel hierarchy, super uses, using final with inheritance, Polymorphism-ad hoc polymorphism, pure polymorphism, method overriding, abstract classes, Object class, forms of inheritance- specialization, specification, construction, extension, limitation, combination, benefits of inheritance, costs of inheritance.

UNIT - II

Packages- Defining a Package, CLASSPATH, Access protection, importing packages.
Interfaces- defining an interface, implementing interfaces, Nested interfaces, applying interfaces, variables in interfaces and extending interfaces.

Stream based I/O (java.io) – The Stream classes-Byte streams and Character streams, Reading console Input and Writing Console Output, File class, Reading and writing Files, Random access file operations, The Console class, Serialization, Enumerations, auto boxing, generics.

UNIT - III

Exception handling - Fundamentals of exception handling, Exception types, Termination or resumptive models, Uncaught exceptions, using try and catch, multiple catch clauses, nested try statements, throw, throws and finally, built- in exceptions, creating own exception sub classes.

Multithreading- Differences between thread-based multitasking and process-based multitasking, Java thread model, creating threads, thread priorities, synchronizing threads, inter thread communication.

UNIT - IV

The Collections Framework (java.util)- Collections overview, Collection Interfaces, The Collection classes- Array List, Linked List, Hash Set, Tree Set, Priority Queue, Array Deque. Accessing a Collection via an Iterator, Using an Iterator, The For-Each alternative, Map Interfaces and Classes, Comparators, Collection algorithms, Arrays, The Legacy Classes and Interfaces- Dictionary, Hashtable,Properties, Stack, Vector
More Utility classes, String Tokenizer, Bit Set, Date, Calendar, Random, Formatter, Scanner

UNIT - V

GUI Programming with Swing – Introduction, limitations of AWT, MVC architecture, components, containers. Understanding Layout Managers, Flow Layout, Border Layout, Grid Layout, Card Layout, Grid Bag Layout.

Event Handling- The Delegation event model- Events, Event sources, Event Listeners, Event classes,Handling mouse and keyboard events, Adapter classes, Inner classes, Anonymous Inner classes.

A Simple Swing Application, Applets – Applets and HTML, Security Issues, Applets and Applications, passing parameters to applets. Creating a Swing Applet, Painting in Swing, A Paint example, Exploring Swing Controls- JLabel and Image Icon, JText Field, **The Swing Buttons**- JButton, JToggle Button, JCheck Box, JRadio Button, JTabbed Pane, JScroll Pane, JList, JCombo Box, Swing Menus, Dialogs.

TEXT BOOKS:

1. Java The complete reference, 9th edition, Herbert Schildt, McGraw Hill Education (India) Pvt.Ltd.
2. Understanding Object-Oriented Programming with Java, updated edition, T. Budd, PearsonEducation.

REFERENCE BOOKS:

1. An Introduction to programming and OO design using Java, J. Nino and F.A. Hosch,John Wiley & sons
2. Introduction to Java programming, Y. Daniel Liang, Pearson Education.
3. Object Oriented Programming through Java, P. Radha Krishna, University Press.
4. Programming in Java, S. Malhotra, S. Chudhary, 2nd edition, Oxford Univ. Press.
5. Java Programming and Object-oriented Application Development, R. A. Johnson, CengageLearning.

KOMMURI PRTAP REDDY INSTITUTE OF TECHNOLOGY
Ghanpur (v),Ghatkesar(m),Medchal dist pin:-504304

Department Computer Science And Engineering

JAVA Programming - LESSON PLAN

Course Code: CS405PC | **Course Title: JAVA Programming**
Class: II – B.Tech – II - Sem | **Academic Year: 2021 -2022**
Instructor: A.simhadri Babu | **Designation: Asst. Professor**

UNIT I Syllabus

Object-Oriented Thinking- A way of viewing world – Agents and Communities, messages and methods, Responsibilities, Classes and Instances, Class Hierarchies- Inheritance, Method binding, Overriding and Exceptions, Summary of Object-Oriented concepts. Java buzzwords, An Overview of Java, Data types, Variables and Arrays, operators, expressions, control statements, Introducing classes, Methods and Classes, String handling.

Inheritance– Inheritance concept, Inheritance basics, Member access, Constructors, Creating Multilevel hierarchy, super uses, using final with inheritance, Polymorphism-ad hoc polymorphism, pure polymorphism, method overriding, abstract classes, Object class, forms of inheritance- specialization, specification, construction, extension, limitation, combination, benefits of inheritance, costs of inheritance.

Session No.	Date	Topic Proposed to be Covered	Text /Reference Book	Chapter No. & Page No.	Web Resources	COs Achieved
1	21.03.2022	A Way of Viewing World — Agents, Responsibility, Messages, Methods, History of Java,	T1	Ch1 & P 3 -9	https://www.tutorialspoint.com/java/java_overview.html	CO1
2	22.03.2022	Java Buzzwords,	T1	CH-1 & P 10 - 13	http://www.w3professors.com/java-tutorials/java-introduction/java-buzzwords/	
3	24.03.2022	JRE, JVM, JDK	T2	Ch-1 & P14	https://howtodoinjava.com/java/basics/jdk-jre-jvm/	
4	24.03.2022	**Object Oriented Thinking and Java Basics**: Need for OOP Paradigm, Summary of OOP Concepts- Over view of JAVA	T2	Ch-1 & P 17-34	http://web.engr.oregonstate.edu/~budd/Books/oopintro3e/info/slides/chap01/java.pdf	
5	26.03.2022	Data Types, Variables,	T1	Ch-3 & p35 -45	http://www.pskills.in/java/data-types-java.jsp	
6	28.03.2022	Scope and Life Time of Variables, type conversion and casting	T1	Ch-3 & p45- 47	https://www.tutorialspoint.com/java/ja	

					va_overview .html	
7	03.04.2022	Arrays	T1	Ch-3 & p51- 58	https://www.tutorialspoint.com/java/java_overview.html	
8	04.04.2022	Operators, Expressions	T1	Ch-4& p61- 79	https://sites.google.com/site/parishudh/corejavat/4-java-operators	
9	5.04.2022	Control Statements, Simple Java Program,	T1	Ch-5& p81- 106	https://sites.google.com/site/parishudh/corejavat/4-java-operators	
10	6.04.2022	Concepts of Classes, Objects, Constructors	T1	Ch-6 & p109- 032	https://www.javatpoint.com/object-and-class-in-java	
11	7.04.2022	Methods, Access Control, This Keyword, Garbage Collection,	T1	Ch-7& P039- 159	https://www.javatpoint.com/object-and-class-in-java	
03	8.04.2022	Overloading Methods and Constructors, Method Binding,	T1	Ch-7& p035- 155	https://www.javatpoint.com/object-class	
13	10.04.2022	Inheritance, Overriding and Exceptions, super, final	T1	Ch-8 & p161 - 185	https://www.javatpoint.com/object-class	
14	13.04.2022	Parameter Passing, Recursion, Nested and Inner Classes, Exploring String Class.	T1	Ch-7 & p035 - 145	https://www.javatpoint.com/object-class	
15	13.04.2022	Polymorphism-ad hoc polymorphism, pure polymorphism, method overriding, abstract classes, Object class	T1	Ch-8 & p161 - 185		

UNIT II Syllabus

Packages- Defining a Package, CLASSPATH, Access protection, importing packages.
Interfaces- defining an interface, implementing interfaces, Nested interfaces, applying interfaces, variables in interfaces and extending interfaces.
Stream based I/O (java.io) – The Stream classes-Byte streams and Character streams, Reading console Input and Writing Console Output, File class, Reading and writing Files, Random access file operations, The Console class, Serialization, Enumerations, auto boxing, generics.

16	16.04.2022	Packages Introduction	T1	Ch-9 & p 187- 188	https://www.javatpoint.com/package	
17	17.04.2022	Package CLASSPATH	T1	Ch-8 & p 188- 189	https://www.javatpoint.com/package	
18	19.04.2022	Access Protection in packages, importing and creating packages	T1	Ch-8 & p 190- 194	https://www.javatpoint.com/package	
19	22.04.2022	Interface introduction – Defining an interface, implementing interfaces	T1	Ch-9 & p 196- 204	https://www.javatpoint.com/interface-in-java	
20	24.04.2022	Nested interfaces, applying interfaces	T1	Ch-9 & p 205- 203	https://www.javatpoint.com/interface-in-java	
21	25.04.2022	variables in interfaces and extending interfaces. Multiple Inheritance with interface	T1	Ch-8 & p 205- 203	https://www.javatpoint.com/interface-in-java	CO2
22	26.04.2022	**Stream based I/O** (java.io) – The Stream classes-Byte streams and Character streams, Reading console Input and Writing Console Output	T1	Ch-13 & p 304- 308	https://www.inf.unibz.it/~russo/AP/LECT4.pdf	
23	27.04.2022	File class, Reading and writing Files, Random access file operations	T1	Ch 13 & P309-336	https://docs.oracle.com/javase/7/docs/api/java/io/RandomAccessFile.html	
24	28.04.2022	The Console class, Serialization	T1	Ch 13 & P308	https://www.inf.unibz.it/~russo/AP/LECT4.pdf	
25	30.04.2022	Enumerations, auto boxing, annotations, generics.	T1	Ch 03 & P263-292	https://www.javatpoint.com/autoboxing-and-unboxing	

Unite -3 Syllabus

Exception handling - Fundamentals of exception handling, Exception types, Termination or resumptive models, Uncaught exceptions, using try and catch, multiple catch clauses, nested try statements, throw, throws and finally, built- in exceptions, creating own exception sub classes.

Multithreading- Differences between thread-based multitasking and process-based multitasking, Java thread model, creating threads, thread priorities, synchronizing threads, inter thread communication.

26	04.05.2022	**Exception Handling: Introduction**	T1	Ch 10& p 213 - 232	https://www.javatpoint.com/exception-handling-in-java	CO3
27	05.05.2022	Concepts of Exception Handling, Benefits of Exception Handling,	T1	Ch 10& p 213 - 232	https://www.javatpoint.com/exception-handling-in-java	
28	5.05.2022	Termination or Resumptive Models, Exception Hierarchy,	T1	Ch 10& p 213 - 232	https://www.javatpoint.com/try-catch-block	
29	05.05.2022	Usage of Try, Catch, Throw, Throws and Finally, Built in Exceptions, Creating Own Exception Sub Classes.	T1	Ch 10& p 213 - 232	https://www.javatpoint.com/try-catch-block	
30	14.05.2022	Differences between Multi-Threading and Multitasking,	T1	Ch 11 & p233 - 259	https://www.javatpoint.com/multithreading http://www.pskills.in/java/user-custom-define.jsp - in-java	
31	5.06.2022	Thread Life Cycle, Creating Threads,	T1	Ch 11 & p233 - 259	https://www.javatpoint.com/life-cycle-of-a-thread	
32	6.06.2022	Thread Priorities, Synchronizing Threads,	T1	Ch 11 & p233 - 259	https://www.javatpoint.com/life-cycle-of-a-thread	
33	7.06.2022	Inter-thread Communication,	T1	Ch 11 & p233 - 259	https://www.javatpoint.com/inter-thread-communication-example	

34	08.06.2022	Thread Groups	T1	Ch 11 & p233 - 259	https://www.javatpoint.com/inter-thread-communication-example	
35	12.06.2022	Daemon Threads. Enumerations,	T1	Ch 11 & p233 - 259	https://www.javatpoint.com/autoboxing-and-unboxing	
Unit -4 Syllabus **The Collections Framework** (java.util)- Collections overview, Collection Interfaces, The Collection classes- Array List, Linked List, Hash Set, Tree Set, Priority Queue, Array Deque. Accessing a Collection via an Iterator, Using an Iterator, The For-Each alternative, Map Interfaces and Classes, Comparators, Collection algorithms, Arrays, The Legacy Classes and Interfaces- Dictionary, Hashtable ,Properties, Stack, Vector More Utility classes, String Tokenizer, Bit Set, Date, Calendar, Random, Formatter, Scanner						
36	13.06.2022	**The Collections Framework** (java.util)- Collections overview, Collection Interfaces	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html	CO4
37	14.06.2022	The Collection classes-	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html	
38	16.06.2022	Array List, Linked List	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html	
39	17.06.2022	Hash Set, Tree Set	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html	
40	18.06.2022	Priority Queue, Array Deque.	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html	
41	20.06.2022	Accessing a Collection via an Iterator, Using an Iterator	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/C	

					oncurrentHashMap.html	
		The For-Each alternative, Map Interfaces and Classes	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html	
42	21.06.2022	Comparators	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html	
43	22.06.2022	Collection algorithms	T1	Ch 11 & P 497 - 577	https://www.tutorialspoint.com/java/java_collection_algorithms.htm	
44	23.06.2022	Arrays, The Legacy Classes and Interfaces- Dictionary	T1	Ch 11 & P 497 - 577	https://www.tutorialspoint.com/java/java_collections.htm	
45	24.06.2022	Hashtable ,Properties	T1	Ch 11 & P 497 - 577	https://beginnersbook.com/2044/07/hashtable-in-java-with-example/	
46	25.06.2022	Stack, Vector More Utility classes	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html	
47	26.06.2022	String Tokenizer,	T1	Ch 11 & P 497 - 577	https://www.geeksforgeeks.org/stringtokenizer-class-java-example-set-1-constructors/	
48	27.06.2022	Bit Set, Date, Calendar	T1	Ch 11 & P 497 - 577	https://www.codota.com/code/java/methods/java.util.Calendar/setTime	
49	29.06.2022	Random, Formatter, Scanner	T1	Ch 11 & P 497 - 577	https://docs.oracle.com/ja	

					vase/8/docs/api/?java/util/Scanner.html	
50	30.06.2022	Revision on Collections - Test				

Unity -5 Syllabus

GUI Programming with Swing – Introduction, limitations of AWT, MVC architecture, components, containers. Understanding Layout Managers, Flow Layout, Border Layout, Grid Layout, Card Layout, Grid Bag Layout.

Event Handling- The Delegation event model- Events, Event sources, Event Listeners, Event classes, Handling mouse and keyboard events, Adapter classes, Inner classes, Anonymous Inner classes.

A Simple Swing Application, Applets – Applets and HTML, Security Issues, Applets and Applications, passing parameters to applets. Creating a Swing Applet, Painting in Swing, A Paint example, Exploring Swing Controls- JLabel and Image Icon, JText Field,

The Swing Buttons- JButton, JToggle Button, JCheck Box, JRadio Button, JTabbed Pane, JScroll Pane, JList, JCombo Box, Swing Menus, Dialogs.

51	1.07.2022	**GUI Programming with Swing** – Introduction, limitations of AWT	T1	Ch 25 & P 797 - 803	https://docs.oracle.com/javase/8/docs/api/java/awt/package-summary.html	CO5
52	2.03.2022	MVC architecture, components, containers.	T1	Ch 25 & P 797 - 803	https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm	
53	2.07.2022	Understanding Layout Managers, Flow Layout, Border Layout,	T1	Ch 26 & P 855 - 865	https://docs.oracle.com/javase/8/docs/api/java/awt/LayoutManager.html	
54	3.07.2022	Grid Layout, Card Layout, Grid Bag Layout.		Ch 26 & P 855 - 865	https://docs.oracle.com/javase/8/docs/api/java/awt/LayoutManager.html	
55	4.07.2022	**Event Handling**: Events Introduction Event Sources		Ch 24 & P 769 - 795	https://docs.oracle.com/javase/8/docs/api/java/awt/event/Action	

					Listener.html	
56	5.07.2022	Event Classes, Event Listeners,	T1	Ch 24 & P 769 - 795	https://www.javatpoint.com/event-handling-in-java	
57	6.07.2022	Delegation Event Model,	T1	Ch 24 & P 769 - 795	https://www.javatpoint.com/event-handling-in-java	
58	7.07.2022	Handling Mouse and Keyboard Events, Adapter Classes.	T1	Ch 24 & P 769 - 795	https://www.javatpoint.com/event-handling-in-java	
59	8.07.2022	**A Simple Swing Application, Applets** – Applets and HTML, Security Issues,	T1	CH 31 & P 1051 - 1037	https://docs.oracle.com/javase/8/docs/api/java/applet/package-frame.html	
60	10.07.2022	Differences between Applets and Applications, Creating Applets,	T1	Ch 23 & P747-767	https://www.javatpoint.com/java-applet	CO6
61	11.07.2022	Passing Parameters to Applets, Creating a Swing Applet, Painting in Swing	T1	Ch 23 & P747-767	https://www.javatpoint.com/java-applet	
62	12.07.2022	Exploring Swing Controls - JLabel and Image Icon, JText Field	T1	Ch 23 & P747-767	https://docs.oracle.com/javase/8/docs/api/javax/swing/package-frame.html	
63	13.07.2022	**The Swing Buttons**- JButton, JToggle Button, JCheck Box, JRadio Button,	T1	Ch 23 & P747-767	https://docs.oracle.com/javase/8/docs/api/javax/swing/package-frame.html	
64	14.07.2022	JTabbed Pane, JScroll Pane, JList, JCombo Box, Swing Menus, Dialogs.	T1	Ch 23 & P747-767	https://docs.oracle.com/javase/8/docs/api/javax/swi	

65	**15.07.2022**	**REVISION**			ng/package-frame.html	

TEXT BOOK:

1. Java The complete reference, 9th edition, Herbert Schildt, McGraw Hill Education (India) Pvt.Ltd.
2. Understanding Object-Oriented Programming with Java, updated edition, T. Budd, Pearson Education.

REFERENCE BOOKS:

1. An Introduction to programming and OO design using Java, J. Nino and F.A. Hosch, John Wiley & sons
2. Introduction to Java programming, Y. Daniel Liang, Pearson Education.
3. Object Oriented Programming through Java, P. Radha Krishna, University Press.
4. Programming in Java, S. Malhotra, S. Chudhary, 2nd edition, Oxford Univ. Press.
5. Java Programming and Object-oriented Application Development, R. A. Johnson, Cengage Learning.

Course Objective

This course is aimed for the student to understanding of OOP concepts and basics of Java programming (Console and GUI based), the skills to apply OOP and ava programming in problem solving and should have the ability to extend his knowledge of java programming further on his/her own.

Course Outcomes

C225	Java Programming	C225.1	Illustrate Object Oriented concepts and basics of java programming
		C225.2	Make use of the concepts of packages and Interfaces
		C225.3	Implement the concepts of multithreading and /or handle run time errors for Java applications
		C225.4	Utilize collection framework and /or file management in Java applications
		C225.5	Design real time applications using event handling concepts.
		C225.6	Develop real time GUI applications using applet, AWT, JDBC and swings

Java - PO s AND CO s MAPPING:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO03		PSO 1	PSO2	PSO3
CO1	3	2	2	1	2	2	2	2	2	2	3	3		CO1	3	3
CO2	2	3	3	1	2	2	2		1	2	3	2		CO2	3	2
CO3	2	2	3	2	2	2	2		2		2	2		CO3	2	3
CO4	2	2	3	2	2	2	2	2	1	2	2	2		CO4	2	2
CO5	1	2	3	2	2	1	2		2	3	2	2		CO5	2	3
CO6	1	2	3	2	2	2	2	3	2	3	3	3		CO6	2	3

Signature of Faculty **HOD**

JAVAPROGRAMMING

B.TECH II YEAR - II
SEM(2021-2022)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(CS405PC)JAVAPROGRAMMING

Objectives:

Thissubjectaimstointroducestudentstothe Javaprogramminglanguage.Uponsuccessfulcompletionofthissubject,studentsshouldbeabletocreateJavaprogramsthatleverage the object-oriented features of the Java language, such as encapsulation, inheritance andpolymorphism;usedatatypes,arraysandotherdatacollections;implementerror-handlingtechniques using exceptionhandling, createand event-driven GUIusingSwingcomponents.

UNIT-I

Object-Oriented Thinking- A way of viewing world – Agents and Communities, messages And methods, Responsibilities, Classes and Instances, Class Hierarchies- Inheritance, Method binding, Overriding and Exceptions, Summary of Object-Oriented concepts. Java buzzwords, An Overview of Java, Data types, Variables and Arrays, operators, expressions, control statements, Introducing classes, Methods and Classes, String handling.

Inheritance– Inheritance concept, Inheritance basics, Member access, Constructors, Creating Multilevel hierarchy, super uses, using final with inheritance, Polymorphism-ad hoc polymorphism, pure polymorphism, method overriding, abstract classes, Object class, forms of inheritance specialization, specification, construction, extension, limitation, combination, benefits of inheritance, costs of inheritance.

UNIT - II

Packages- Defining a Package, CLASSPATH, Access protection, importing packages. Interfaces- defining an interface, implementing interfaces, Nested interfaces, applying interfaces, variables in interfaces and extending interfaces.

Stream based I/O (java.io) – The Stream classes-Byte streams and Character streams, Reading console Input and Writing Console Output, File class, Reading and writing Files, Random access file operations, The Console class, Serialization, Enumerations, auto boxing, generics.

UNIT - III

Exception handling - Fundamentals of exception handling, Exception types, Termination or resumptive models, Uncaught exceptions, using try and catch, multiple catch clauses, nested try statements, throw, throws and finally, built- in exceptions, creating own exception sub classes.

Multithreading- Differences between thread-based multitasking and process-based multitasking, Java thread model, creating threads, thread priorities, synchronizing threads, inter thread communication.

UNIT - IV

The Collections Framework (java.util)- Collections overview, Collection Interfaces, The Collection classes- Array List, Linked List, Hash Set, Tree Set, Priority Queue, Array Deque. Accessing a Collection via an Iterator, Using an Iterator, The For-Each alternative, Map Interfaces and Classes, Comparators, Collection algorithms, Arrays, The Legacy Classes and Interfaces- Dictionary, Hashtable ,Properties, Stack, Vector,More Utility classes, String Tokenizer, Bit Set, Date, Calendar, Random, Formatter, Scanner

UNIT - V

GUI Programming with Swing – Introduction, limitations of AWT, MVC architecture, components, containers. Understanding Layout Managers, Flow Layout, Border Layout, Grid Layout, Card Layout, Grid Bag Layout.

Event Handling- The Delegation event model- Events, Event sources, Event Listeners, Event classes, Handling mouse and keyboard events, Adapter classes, Inner classes, Anonymous Inner classes.

A Simple Swing Application, Applets – Applets and HTML, Security Issues, Applets and Applications, passing parameters to applets. Creating a Swing Applet, Painting in Swing, A Paint example, Exploring Swing Controls- JLabel and Image Icon, JText Field, **The Swing Buttons** JButton, JToggle Button, JCheck Box, JRadio Button, JTabbed Pane, JScroll Pane, JList, JCombo Box, Swing Menus, Dialogs.

TEXT BOOKS:

1. Java The complete reference, 9th edition, Herbert Schildt, McGraw Hill Education (India) Pvt. Ltd.
2. Understanding Object-Oriented Programming with Java, updated edition, T. Budd, Pearson Education.

REFERENCEBOOKS:

1. JavaforProgrammers,P.J.DeitelandH.M.Deitel,PEA(or)Java:HowtoProgram,P.J.Deiteland H.M.Deitel, PHI
2. ObjectOrientedProgrammingthroughJava,P.RadhaKrishna,UniversitiesPress.
3. ThinkinginJava, Bruce Eckel, PE
4. ProgramminginJava,S.MalhotraandS.Choudhary,OxfordUniversitiesPress.

CourseOutcomes:

- An understanding of theprinciples and practiceof objectoriented analysisand design intheconstruction ofrobust, maintainableprogramswhich satisfytheirrequirements;
- A competence to design, write, compile, test and execute straightforward programs using ahighlevel language;
- Anappreciationoftheprinciplesofobject orientedprogramming;
- Anawarenessoftheneedforaprofessionalapproachtodesignandtheimportanceofgooddocume ntation to the finishedprograms.
- Beabletoimplement,compile,testandrunJavaprogramscomprisingmorethanoneclass,to addressaparticularsoftwareproblem.
- Demonstratethe abilitytousesimple datastructures likearraysin aJavaprogram.
- Beableto makeuse of members of classes found in the JavaAPI(suchas theMath class).

INDEX

Unit-1

OOPConcepts

Object Oriented Programming is a paradigm that provides many concepts such as ***inheritance,databinding,polymorphism****etc.*

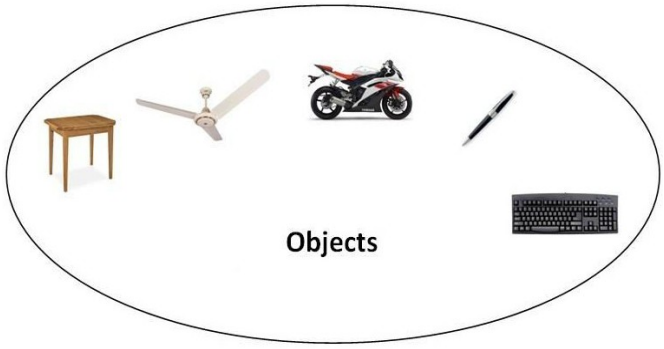
Simula isconsideredasthefirstobject-orientedprogramminglanguage.Theprogrammingparadigmwhereeverythingisrepresentedasanobjectisknownastrulyobject-orientedprogramminglanguage.

Smalltalk isconsideredasthefirst trulyobject-orientedprogramminglanguage.

OOPs(ObjectOrientedProgrammingSystem)

Objectmeansarealwordentitysuchaspen,chair,tableetc.**Object-OrientedProgramming** isa methodology or paradigm to design a program using classes and objects. It simplifies thesoftwaredevelopment and maintenancebyprovidingsomeconcepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation



Object

Any entity that has state and behavior is known as an object. For example: chair, pen, table,keyboard,bikeetc. It can bephysicalandlogical.

Class

Collectionof objectsiscalledclass.Itisalogicalentity.

Inheritance

When one object acquires all the properties and behaviours of parent object i.e. known asinheritance.It providescodereusability.It isusedtoachieveruntimepolymorphism.

Polymorphism

When**onetaskisperformedbydifferentways**i.e.knownaspolymorphism.Forexample:toconvincethecustomer differently, todraw somethinge.g. shapeorrectangleetc.

In java, we use method overloading and method overriding to achieve

polymorphism.Another example can be to speak something e.g. cat speaks meaw, dog

barks woof etc.**Abstraction**

Hidinginternaldetailsandshowingfunctionalityisknownasabstraction.Forexample:phone call,wedon'tknowtheinternalprocessing.

Injava, weuseabstractclass andinterfaceto achieveabstraction.

Encapsulation

Binding(orwrapping)codeanddatatogetherintoasingleunitisknownasencapsulation.Forexample: capsule, it is wrapped with differentmedicines.

Ajavaclassistheexampleofencapsulation.Javabeanisthefullyencapsulatedclassbecauseallthedata membersareprivate here.

Benefitsof Inheritance

- One of the key benefits of inheritance is to minimize the amount of duplicate code in anapplication by sharing common code amongst several subclasses. Where equivalent codeexists in two related classes, the hierarchy can usually be refactored to move the commoncode up to a mutual superclass. This also tends to result in a better organization of code andsmaller,simpler compilation units.
  - Inheritancecanalsomakeapplication code moreflexibleto changebecauseclassesthatinherit from a common superclass can be used interchangeably. If the return type of amethodis superclass
  - **Reusability**-facilitytouse publicmethods of baseclass without rewritingthe same.

- **Extensibility** -extending thebaseclass logic asperbusiness logicofthederivedclass.

- **Datahiding**-baseclass candecide tokeep some data privateso thatit cannotbe alteredbythe derivedclass

Proceduralandobjectorientedprogrammingparadigms

Features	Procedural Oriented Programming (POP)	Object Oriented Programming (OOPS)
Divided into	In POP, program is divided into smaller parts called as functions.	in OOPs , the program is divided into parts known as **objects**.
Importance	In POP, importance is not given to **data** but to functions as well as **sequence** of actions to be done.	In OOPs, Importance is given to the data rather than procedures or functions because it works as a **real world**.
Approach	POP follows **Top Down approach**.	OOPs follows **Bottom Up approach**.
Access Specifiers	POP does not have any access specifier.	OOPs has access specifiers named Public, Private, Protected, etc.
Data Moving	In POP, Data can move freely from function to function in the system.	In OOPs, objects can move and communicate with each other through member functions.
Data Access	In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system.	In OOPs, data can not move easily from function to function,it can be kept public or private so we can control the access of data.
Data Hiding	POP does not have any proper way for hiding data so it is **less secure**.	OOPs provides Data Hiding so provides **more security**.
Overloading	In POP, Overloading is not possible.	In OOPs, overloading is possible in the form of Function Overloading and Operator Overloading.
Examples	C, VB, FORTRAN, Pascal.	C++, JAVA, VB.NET, C#.NET.

ThehistoryofjavastartsfromGreenTeam.Javateammembers(alsoknownas **Green Team**), initiated a revolutionary task to develop a language for digitaldevicessuchas set-topboxes,televisionsetc.

For the green team members, it was an advance concept at that time. But, it wassuitedforinternetprogramming.Later,Javatechnologyasincorporatedby Netscape.

Currently, Java is used in internet programming, mobile devices, games, e-businesssolutionsetc.Therearegiventhemajorpointsthatdescribesthehistoryofjava.

1) **JamesGosling**, **MikeSheridan**,and **PatrickNaughton** initiatedtheJavalanguage project in June 1991. The small team ofsun engineers called **GreenTeam**.

2) Originally designed for small, embedded systems in electronic appliances like set-topboxes.

3) Firstly,itwascalled**"Greentalk"**byJamesGoslingandfileextensionwas.gt.

4) After that, it was called Oak and was developed as a part of the Greenproject.

JavaVersionHistory

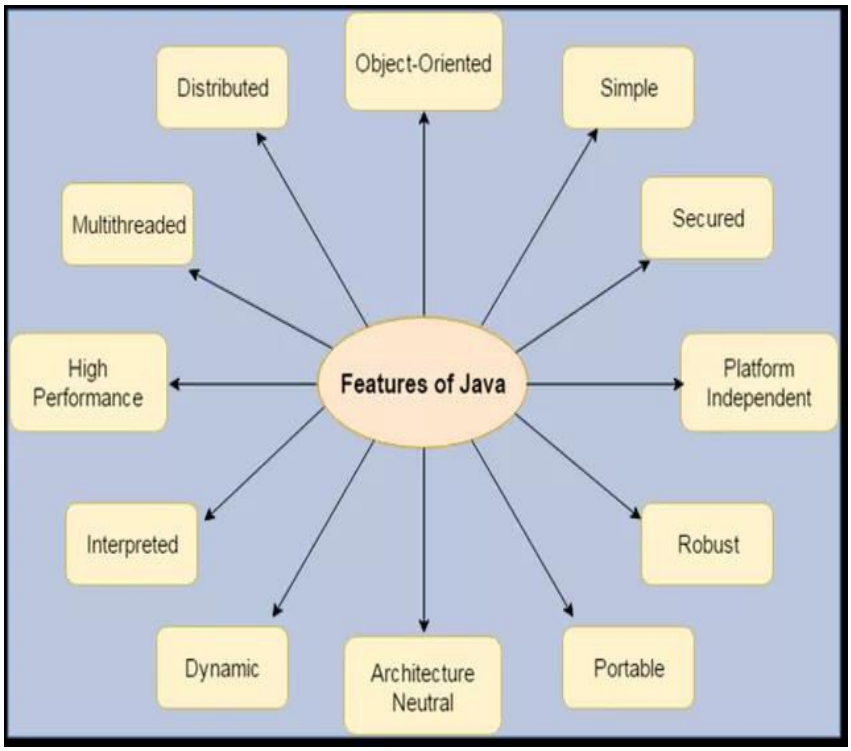
Therearemanyjava versionsthathasbeenreleased.CurrentstablereleaseofJavaisJavaSE8.

1. JDK Alpha and Beta (1995)2.JDK 1.0 (23rd Jan, 1996)3.JDK 1.1 (19th Feb, 1997)4.J2SE 1.2 (8th Dec, 1998)5.J2SE 1.3 (8th May, 2000)6.J2SE 1.4 (6th Feb, 2002)7.J2SE5.0(30thSep,2004)
8. JavaSE6(11thDec, 2006)
9. Java SE 7 (28th July, 2011)10.JavaSE8(18thMarch,2014)

FeaturesofJava

There is given many features of java. They are also known as java buzzwords. The Java Featuresgivenbelow aresimple and easyto understand.

1. Simple
2. Object-Oriented
3. Portable
4. Platformindependent
5. Secured
6. Robust
7. Architectureneutral
8. Dynamic
9. Interpreted
10. HighPerformance
11. Multithreaded
12. Distributed



JavaComments

The java comments are statements that are not executed by the compiler and interpreter. Thecomments can be used to provide information or explanation about the variable, method, class oranystatement.It canalsobeusedto hide programcodeforspecifictime.

TypesofJavaComments

Thereare3types ofcomments in java.

1. SingleLineComment
2. MultiLineComment
3. DocumentationComment

JavaSingleLineComment

Thesingle linecommentis used to commentonlyoneline.

Syntax:

1. //Thisissingle linecomment

Example:

```
publicclassCommentExample1{
public static void main(String[] args)
   {int i=10;//Here, i is a
   variableSystem.out.println(i);
}
}
```

Output:

```
10
```

JavaMultiLineComment

Themultilinecommentis usedto commentmultiple linesof code.

Syntax:

```
/*T
hisis
multi
linecom
ment
*/
```

Example:

```
publicclassCommentExample2{
publicstaticvoidmain(String[] args){
/*Let'sdeclareandprintva
 riableinjava.*/
   int
   i=10;System.out.prin
   tln(i);
}}
```

Output:

```
10
```

JavaDocumentationComment

The documentation comment is used to create documentation API. To create documentation API, you needto use**javadoctool**.

Syntax:

```
/**
This
is
documentation
comment
*/
```

Example:

```
/**TheCalculator class providesmethodstoget additionandsubtractionof given2numbers.*/
publicclassCalculator{
/**Theadd() methodreturns additionof given numbers.*/
publicstatic int add(inta,intb){returna+b;}
/**Thesub() methodreturns subtractionof given numbers.*/
publicstatic int sub(inta,intb){returna-b;}
}
```

Compileitbyjavactool:

```
javacCalculator.java
```

CreateDocumentation APIbyjavadoctool:

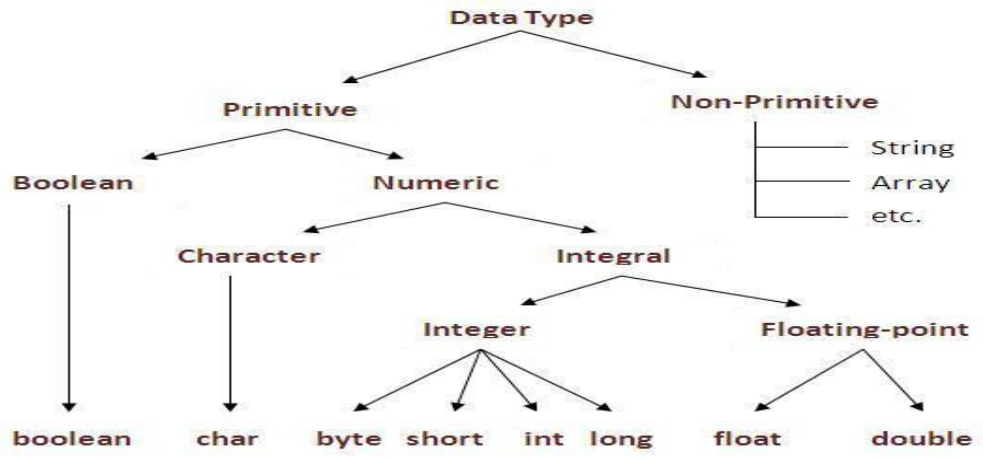
```
javadocCalculator.java
```

Now,therewillbeHTMLfilescreatedforyourCalculatorclassinthecurrentdirectory.OpentheHTMLfilesandseetheexplanation ofCalculatorclass provided throughdocumentationcomment.

DataTypes

Datatypesrepresentthedifferentvaluesto bestoredinthevariable.Injava, therearetwotypesofdatatypes:

- Primitivedatatypes
- Non-primitivedatatypes



DataType	DefaultValue	Defaultsize
boolean	false	1bit
char	'\u0000'	2byte
byte	0	1byte
short	0	2byte
int	0	4byte
long	0L	8byte
float	0.0f	4byte
double	0.0d	8byte

JavaVariableExample:AddTwoNumbers

```
classSimple{
publicstaticvoid main(String[] args){
int
a=10;int
b=10;intc
=a+b;
System.out.println(c);
}}
```

Output:20

VariablesandDataTypes inJava

Variableisanameofmemorylocation.Therearethreetypesofvariablesinjava:local,instanceandstatic.

Therearetwotypes ofdata typesin java:primitive andnon-primitive.

TypesofVariable

Therearethreetypes of variablesinjava:

- localvariable
- instancevariable
- staticvariable

1) LocalVariable

Avariablewhichisdeclaredinsidethemethodiscalledlocalvariable.

2) InstanceVariable

Avariable which is declared insidethe class but outsidethe method, iscalledinstancevariable.Itisnot declaredas static.

3) Staticvariable

A variable that is declared as static is called static variable. It cannot be

local.Wewill havedetailed learningof thesevariables in next chapters.

Exampletounderstandthetypesofvariablesinjava

```
classA{
int data=50;//instance
variablestatic int m=100;//static
variablevoid method(){
intn=90;//localvariable
}
}//endofclass
```

ConstantsinJava

A constant is a variable which cannot have its value changed after declaration. It uses the **'final'keyword.**

Syntax

```
modifierfinaldataType variableName=value;//globalconstant

modifierstaticfinal dataTypevariableName=value;//constantwithin ac
```

ScopeandLifeTimeofVariables

The scope of a variable defines the section of the code in which the variable is visible. As ageneral rule, variables that are defined withina block are not accessible outside that block.Thelifetimeofavariablereferstohowlongthevariableexistsbeforeitisdestroyed.Destroying variables refers to deallocating the memory that was allotted to the variables whendeclaring it. We have written a few classes till now. You might have observed that not allvariables are the same. The ones declared in the body of a method were different from thosethat were declared in the class itself. There are three types of variables: instance variables,formalparametersor localvariablesandlocalvariables.

Instancevariables

Instance variables are those that are defined within a class itself and not in any method orconstructor of the class. They are known as instance variables because every instance of theclass (object) contains a copy of these variables. The scope of instance variables is determinedby the access specifier that is applied to these variables. We have already seen about it earlier.The lifetime of these variables is the same as the lifetime of the object to which it belongs.Object once created do not exist for ever. They are destroyed by the garbage collector of Javawhen there are no more reference to that object. We shall see about Java's automatic garbagecollectorlateron.

Argumentvariables

Thesearethevariablesthataredefinedintheheaderoafconstructororamethod.Thescopeof these variables is the method or constructor in which they are defined. The lifetime islimitedtothetimeforwhichthemethodkeepsexecuting.Oncethemethodfinishesexecution,thesev ariablesaredestroyed.

Localvariables

A local variable is theone that is declaredwithina method ora constructor(not in theheader).Thescopeandlifetimearelimitedtothemethoditself.

One important distinction between these three types of variables is that access specifiers canbeappliedtoinstancevariablesonlyandnottoargumentor localvariables.

In addition to the local variables defined in a method, we also have variables that are definedinbockslifeanifblockandanelseblock.Thescopeandisthesameasthatoftheblockitself.

Operatorsinjava

Operator in java is a symbol that is used to perform operations. For example: +, -, *, /

etc.Therearemanytypes ofoperators in javawhichare given below:

- UnaryOperator,
- ArithmeticOperator,
- shiftOperator,
- RelationalOperator,
- BitwiseOperator,
- LogicalOperator,
- TernaryOperatorand
- AssignmentOperator.

OperatorsHierarchy

Operator Precedence

Operators	Precedence
postfix	`expr++ expr--`
unary	`++expr --expr +expr -expr ~ !`
multiplicative	`* / %`
additive	`+ -`
shift	`<< >> >>>`
relational	`< > <= >= instanceof`
equality	`== !=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Expressions

Expressions are essential building blocks of any Java program, usually created to produce a newvalue, although sometimes an expression simply assigns a value to a variable. Expressions arebuiltusingvalues, variables, operatorsand method calls.

TypesofExpressions

While an expression frequently produces a result, it doesn't always. There are three types ofexpressions inJava:

- Thosethat produceavalue, i.e. theresultof(1 +1)
- Thosethat assignavariable,forexample(v =10)
- Those that have no result but might have a "side effect" because an expression can includea wide range of elements such as method invocations or increment operators that modifythestate(i.e. memory)ofaprogram.

JavaTypecastingandTypeconversion

WideningorAutomatic TypeConversion

Wideningconversiontakesplacewhentwodatatypesareautomaticallyconverted.Thishappenswhen:

- Thetwodatatypesarecompatible.
- Whenweassignvalueofasmallerdatatypetoabiggerdata type.

For Example, in java the numeric data types are compatible with each other but no automaticconversion is supported from numeric type to char or boolean. Also, char and boolean are notcompatible with each other.

Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

NarrowingorExplicitConversion

Ifwewanttoassignavalueoflargerdatatypetoasmallerdatatypeweperformexplicittypecastingornarrowing.

- Thisisuseful forincompatible datatypeswhere automaticconversion cannotbedone.
- Here,target-typespecifiesthedesiredtypetoconvertthespecifiedvalue to.

Double –> Float –> Long –> Int –> Short –> Byte

- Narrowing or Explicit Conversion

JavaEnum

Enuminjavais adata typethat containsfixed setof constants.

Itcanbeusedfordaysoftheweek(SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDA
YandSATURDAY),directions(NORTH,SOUTH,EASTandWEST)
etc. The java enum constants are static and final implicitly. It is available from JDK

1.5.JavaEnums can bethought ofas classes that havefixedset ofconstants.

Simpleexampleofjava enum

```
classEnumExample1{
publicenumSeason{WINTER,SPRING, SUMMER,FALL}

public static void main(String[] args)
{for (Season s :
Season.values())System.out.println(s);
}}
```

Output:

```
WINTERSPRIN
GSUMMER
FALL
```

ControlFlow Statements

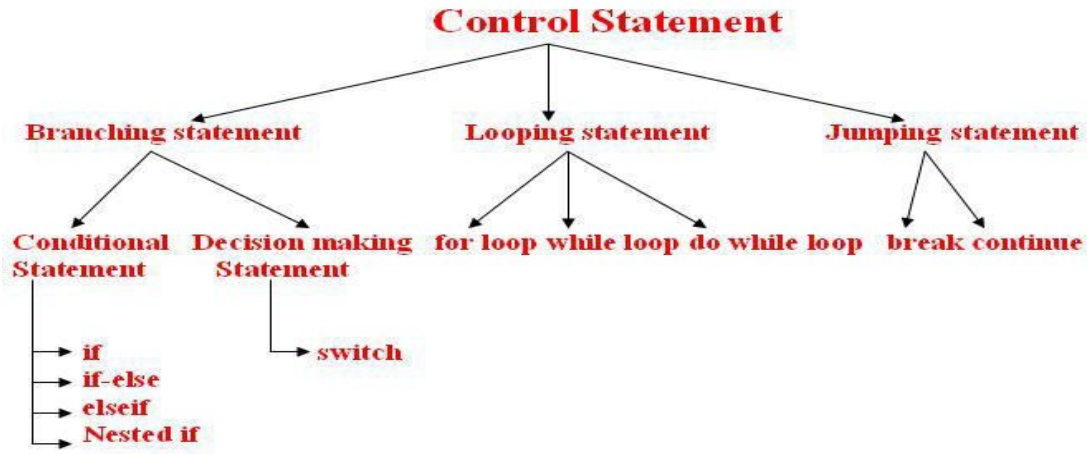
The control flow statements in Java allow you to run or skip blocks of code when specialconditionsaremet.

The"if"Statement

The "if" statement in Java works exactly like in most programming languages. With the help of"if" you can choose to execute a specific block of code when a predefined condition is met. Thestructureof the"if"statement in Javalooks likethis:

```
if (condition){
//executethiscode
}
```

The condition is Boolean. Boolean means it may be true or false. For example you may put amathematicalequation ascondition.Look atthis full example:



CreatingaStand-AloneJavaApplication

1. Write a main method that runs your program. You can write this method anywhere. In thisexample, I'll write my main method in a class called Main that has no other methods.

Forexample:

```
2.      public class
Main3.      {
4.        public static void main(String[]
args)5.{
6.          Game.play();
7.        }}
```

8. Makesureyourcodeiscompiled,andthatyou havetesteditthoroughly.
9. If you're using Windows, you will need to set your path to include Java, if you haven'tdone so already. This is a delicate operation. Open Explorer, and look insideC:\ProgramFiles\Java, and you should see some version of the JDK. Open this folder, andthen open the bin folder. Select the complete path from the top of the Explorer window, andpressCtrl-C to copyit.

Next, find the "My Computer" icon (on your Start menu or desktop), right-click it, and selectproperties. Click on the Advanced tab, and then click on the Environment variables button.Look at the variables listed for all users, and click on the Path variable. Do not delete thecontents of this variable! Instead, edit the contents by moving the cursor to the right end,entering a semicolon (;), and pressing Ctrl-V to paste the path you copied earlier. Then goahead and save your changes. (If you have any Cmd windows open, you will need to closethem.)

10. If you're using Windows, go to the Start menu and type "cmd" to run a program thatbrings up a command prompt window. If you're using a Mac or Linux machine, run theTerminalprogram to bringup acommand prompt.
11. In Windows, type dir at the command prompt to list the contents of the current directory.OnaMacorLinuxmachine, type ls to do this.

12. Now we want to change to the directory/folder that contains your compiled code. Look atthe listing of sub-directories within this directory, and identify which one contains your code.Type cd followed by the name of that directory, to change to that directory. For example, tochangeto a directorycalledDesktop,you would type:

cdDesktop

Tochangetotheparentdirectory,type:

cd..

Every time you change to a new directory, list the contents of that directory to see where to gonext.Continuelistingandchangingdirectoriesuntilyoureachthedirectorythatcontains your.classfiles.

13. IfyoucompiledyourprogramusingJava1.6,but plantorunitonaMac,you'llneedtorecompileyourcodefrom thecommand line, bytyping:

javac-target1.5*.java

14. Nowwe'llcreate asingle JARfilecontainingallofthefilesneeded torunyourprogram.

Arrays

Java provides a data structure, the **array**,which stores afixed-size sequential collection ofelements of the same type. An array is used to store a collection of data, but it is often moreusefulto think ofan arrayasacollection ofvariables ofthesame type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, youdeclareonearrayvariablesuchasnumbersandusenumbers[0],numbers[1],and...,numbers[99] to represent individual variables.

This tutorial introduces how to declare array variables, create arrays, and process arrays usingindexedvariables.

DeclaringArrayVariables:

To use an array in a program, you must declare a variable to reference the array, and you mustspecify the type of array the variable can reference. Here is the syntax for declaring an arrayvariable:

```
dataType[] arrayRefVar;// preferred way.or
dataTypearrayRefVar[];//worksbutnotpreferredway.
```

Note: The style **dataType[] arrayRefVar** is preferred. The style **dataTypearrayRefVar[]** comesfromthe C/C++ language andwasadopted inJava toaccommodateC/C++programmers.

Example:

Thefollowingcodesnippets are examples ofthis syntax:

```
double[]myList;        //preferredway.
or
doublemyList[];        //worksbutnotpreferred way.
```

CreatingArrays:

Youcan createan arraybyusingthe new operator with the followingsyntax:

```
arrayRefVar=newdataType[arraySize];
```

Theabovestatement does two things:

- ItcreatesanarrayusingnewdataType[arraySize];
- Itassignsthereferenceofthenewlycreated arrayto thevariablearrayRefVar.

Declaringanarrayvariable,creatinganarray,andassigningthereferenceofthearraytothevariablecan becombined in one statement, as shown below:

```
dataType[]arrayRefVar=newdataType[arraySize];
```

Alternativelyyoucancreatearraysasfollows:

```
dataType[]arrayRefVar={value0,value1,...,valuek};
```

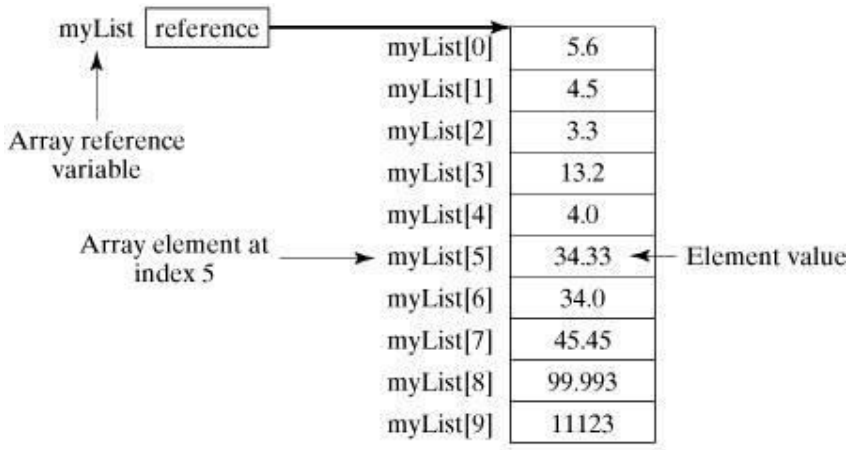
Thearrayelementsareaccessedthroughthe**index**.Arrayindicesare0-based;thatis,theystartfrom0 to **arrayRefVar.length-1**.

Example:

Followingstatementdeclaresanarrayvariable,myList,createsanarrayof10elementsofdoubletypeand assigns its referenceto myList:

```
double[]myList=newdouble[10];
```

Followingpicturerepresentsarraymy List.Here,myListholdstendoublevaluesandtheindicesarefrom 0 to 9.



ProcessingArrays:

Whenprocessingarrayelements,weoftenuseeitherforlooporforeachloopbecausealloftheelementsin an arrayare ofthe same typeand thesizeof thearrayis known.

Example:

Hereisacompleteexample ofshowinghowto create,initializeand processarrays:

```
publicclass TestArray
{
  public static void main(String[] args)
    {double[]myList={1.9,2.9,3.4,3.5};
          //Printallthearrayelementsfor(in
    t i=0;i<myList.length; i++){
      System.out.println(myList[i]+"");
    }
    // Summing all
    elementsdoubletotal =0;
    for (int i = 0; i <myList.length; i++)
      {total+=myList[i];
    }
    System.out.println("Totalis"+total);
    // Finding the largest
    elementdoublemax=myList[
    0];
    for (int i = 1; i <myList.length; i++)
      {if(myList[i]>max)max =myList[i];
    }
    System.out.println("Maxis"+max);
  }
```

Thiswouldproducethefollowingresult:

```
1.9
2.9
3.4
3.5
Totalis 11.7
Maxis 3.5
```

```
publicclass TestArray{
public static void main(String[] args)
    {double[]myList ={1.9,2.9,3.4,3.5};
 //Print all the arrayelements
    for(doubleelement:myList){S
      ystem.out.println(element);
}}}
```

JavaConsoleClass

TheJavaConsoleclassisbeusedtogetinputfromconsole.Itprovidesmethodstoreadtextsandpasswords.

Ifyou readpasswordusingConsoleclass,itwillnotbedisplayedtotheuser.

Thejava.io.Consoleclassisattachedwithsystemconsoleinternally.TheConsoleclassisintroducedsince 1.5.

Let'sseeasimpleexampleto read textfrom console.

```
1.      Stringtext=System.console().readLine();
2.      System.out.println("Textis: "+text);
```

JavaConsoleExample

```
importjava.io.Console;
classReadStringTest{
public static void main(String
args[]){Console
c=System.console();System.out.println("E
nter your name: ");String
n=c.readLine();System.out.println("Welco
me"+n);                                    }    }
```

Output

```
Enter your name: Nakul
JainWelcomeNakul Jain
```

Constructors

Constructorin javaisa*specialtypeofmethod* thatisused toinitializetheobject.

Java constructor is *invoked at the time of object creation*. It constructs the values i.e. provides dataforthe object that is whyit is known as constructor.

Therearebasicallytwo rules defined for theconstructor.

1. Constructorname mustbesameas itsclass name
2. Constructormusthavenoexplicitreturntype

Typesofjavaconstructors

Therearetwotypes ofconstructors:

1. Defaultconstructor(no-argconstructor)
2. Parameterizedconstructor

JavaDefaultConstructor

Aconstructorthat havenoparameterisknownasdefaultconstructor.

Syntaxofdefaultconstructor:

1. <class_name>(){}

Exampleofdefaultconstructor

In this example, we are creating the no-arg constructor in the Bike class. It will be invoked atthetimeof object creation.

```
classBike1{
Bike1(){System.out.println("Bikeiscreated");}
public static void main(String
```

```
args[]){Bike1b=new Bike1();
}}
```

Output: Bikeis created

ExampleofparameterizedconstructorIn this example, we have created the constructor of Student class that have two parameters. Wecan haveanynumberofparameters in the constructor.

```
classStudent4{
intid;
Stringname;

Student4(int i,String
n){id =i;
name=n;
}
voiddisplay(){System.out.println(id+""+name);}

public static void main(String
args[]){Student4 s1 = new
Student4(111,"Karan");Student4 s2 = new
Student4(222,"Aryan");s1.display();
s2.display();
}}
```

Output:

```
111Karan
222Aryan
```

ConstructorOverloading inJava

Constructor overloading is a technique in Java in which a class can have any number ofconstructors that differ in parameter lists.The compiler differentiates these constructors bytakinginto account thenumberof parameters in the list and their type.

ExampleofConstructorOverloading

```
classStudent5{
intid;String
name;intage
;
Student5(int i,String
n){id =i;
name=n;
}
Student5(int i,String n,int
a){id =i;
name =
n;age=a;
}
voiddisplay(){System.out.println(id+""+name+""+age);}

public static void main(String
args[]){Student5 s1 = new
Student5(111,"Karan");Student5 s2 = new
Student5(222,"Aryan",25);s1.display();
```

```
s2.display();
}}
```

Output:

```
111Karan0
222Aryan25
```

JavaCopyConstructor

Thereisnocopyconstructorinjava.But,wecancopythevaluesofoneobjecttoanotherlikecopyconstructor in C++.

There aremanyways tocopythevalues ofoneobject into another in java.Theyare:

- Byconstructor
- Byassigningthe values of oneobject into another
- Byclone()method ofObject class

Inthisexample,wearegoingtocopythevaluesofoneobjectintoanotherusingjavaconstructor.

```
classStudent6{
intid;
Stringname;
Student6(int i,String
n){id =i;
name=n;
}

Student6(Student6
s){id =s.id;
name=s.name;
}
voiddisplay(){System.out.println(id+""+name);}

public static void main(String
args[]){Student6 s1 = new
Student6(111,"Karan");Student6s2
=newStudent6(s1);s1.display();
s2.display();
}}
```

Output:

```
111Karan
111Karan
```

Java-Methods

A Java method is a collection of statements that are grouped together to perform an operation.WhenyoucalltheSystem.out.**println()** method,forexample,thesystemactuallyexecutesseveralstatements in order to displayamessageonthe console.

Now you will learn how to create your own methods with or without return values, invoke amethodwith or without parameters, and applymethod abstraction in theprogram design.

CreatingMethod

Consideringthe followingexampleto explainthesyntaxof amethod−

Syntax

```
publicstaticint methodName(inta, intb) {
  // body
}
```

Here,

- **publicstatic**−modifier
- **int**−returntype
- **methodName** −nameofthemethod
- **a,b**−formalparameters
- **inta, intb**−listofparameters

Methoddefinitionconsistsofamethodheaderandamethodbody.Thesameisshowninthe followingsyntax−

Syntax

```
modifierreturnTypenameOfMethod(Parameter List){
  // methodbody
}
```

Thesyntaxshownabove includes−

- **modifier**− Itdefinestheaccesstypeofthemethodand itisoptionalto use.
- **returnType**−Methodmayreturnavalue.

- **nameOfMethod** − Thisisthe methodname. Themethodsignature consistsof the method nameand theparameter list.

- **Parameter List** − The list of parameters, it is the type, order, and number ofparametersofamethod. Theseareoptional, method maycontainzero parameters.
- **methodbody** −Themethodbodydefineswhatthemethoddoeswith thestatements.

CallbyValueandCallbyReferenceinJava

There is only call by value in java, not call by reference. If we call a method passing a value, itis known as call by value. The changes being done in the called method, is not affected in thecallingmethod.

Exampleofcallbyvaluein java

Incaseof callbyvalueoriginal valueis notchanged.Let'stake asimpleexample:

```
classOperation{
 intdata=50;
 voidchange(intdata){
 data=data+100;//changes willbein thelocalvariableonly
 }
 public static void main(String
  args[]){Operationop=newOperation();System.
  out.println("before change
  "+op.data);op.change(500);
  System.out.println("afterchange"+op.data);
 }
}
```

```
Output:beforechange50
       afterchange50
```

InJava,parameters arealways passedbyvalue. Forexample, followingprogramprintsi = 10,j = 20.

```
//Test.javacla
ssTest{
  //swap()doesn'tswapiandj
  public static void swap(Integer i, Integer j)
    {Integertemp=newInteger(i);
    i = j;
    j= temp;
  }
  public static void main(String[] args)
    {Integeri=newInteger(10);
    Integerj=newInteger(20);sw
    ap(i,j);
    System.out.println("i="+i + ", j="+j);
```

```
    }
}
```

StaticFieldsandMethods

The **static keyword** in java is used for memory management mainly. We can apply java statickeyword with variables, methods, blocks and nested class. The static keyword belongs to the classthan instanceof the class.

Thestatic canbe:

1. variable(alsoknownasclassvariable)
2. method(alsoknownas classmethod)
3. block
4. nestedclass

Javastaticvariable

Ifyoudeclareanyvariableasstatic,itisknownstaticvariable.

- Thestaticvariablecanbeusedtoreferthecommonpropertyofallobjects(thatisnotuniqueforeachobject) e.g. companyname ofemployees,collegenameofstudents etc.
- Thestatic variablegets memoryonlyonce inclassarea atthetime ofclassloading.

Advantageof static variable

Itmakesyourprogram**memoryefficient** (i.eitsavesmemory).

Understandingproblemwithoutstaticvariable

```
1. class Student{
2.     introllno;
3.     Stringname;
4.     String
college="ITS";5.}
```

Exampleofstaticvariable

```
//Programof staticvariable
class Student8{
```

```
int rollno;
```

```
  Stringname;
  static String college
  ="ITS";Student8(int r,String
  n){rollno= r;
  name=n;
  }
 voiddisplay(){System.out.println(rollno+""+name+""+college);}
 public static void main(String
 args[]){Student8 s1 = new
 Student8(111,"Karan");Student8s2=newStude
 nt8(222,"Aryan");

 s1.display();
 s2.display();
 }}
```

```
Output:111KaranITS
       222AryanITS
```

Javastaticmethod

Ifyouapplystatickeyword withanymethod,it isknown asstaticmethod.

- Astaticmethodbelongs tothe classrather thanobject ofaclass.
- Astaticmethodcan beinvoked without theneed forcreatinganinstanceofaclass.
- staticmethodcan accessstaticdata memberand canchangethe valueof it.

Exampleofstaticmethod

```
//Programof changingthecommon propertyof allobjects(staticfield).

class
  Student9{in
  t
  rollno;String
  name;
  staticStringcollege= "ITS";
  static void
  change(){college="B
```

```
BDIT";
}
Student9(int r, String
n){rollno=r;
name=n;
```

```
}
 void display(){System.out.println(rollno+""+name+""+college);}
public static void main(String
args[]){Student9.change();
Student9 s1 = new Student9
(111,"Karan");Student9 s2 = new Student9
(222,"Aryan");Student9 s3 = new Student9
(333,"Sonoo");s1.display();
s2.display();
s3.display();
}}
```

```
Output:111 Karan
    BBDIT222AryanBB
    DIT
```

Javastaticblock

- Isusedtoinitializethestaticdatamember.
- Itis executedbefore mainmethod at thetime ofclassloading.

Example of static

```
blockclassA2{
static{System.out.println("static block is
invoked");}public static void main(String
args[]){System.out.println("Hello main");
}}
```

```
Output: static block is
    invokedHellomain
```

AccessControl

AccessModifiersinjava

Therearetwotypesofmodifiersinjava:**accessmodifiers**and**non-accessmodifiers**.

Theaccessmodifiersinjavaspecifiesaccessibility(scope)of adatamember,method,constructororclass.

Thereare4typesof java accessmodifiers:

1. private
2. default
3. protected
4. public

privateaccess modifier

Theprivate access modifieris accessibleonlywithinclass.

Simpleexampleofprivateaccess modifier

In this example, we have created two classes A and Simple. A class contains private datamember and private method. We are accessing these private members from outside the class,sothereis compile timeerror.

```
classA{
privateintdata=40;
privatevoid msg(){System.out.println("Hellojava");}}
publicclassSimple{
 public static void main(String
  args[]){Aobj=newA();
  System.out.println(obj.data);//Compile Time
  Errorobj.msg();//CompileTimeError
  }}
```

2) *defaultaccessmodifier*

Ifyoudon'tuseany modifier,itistreatedas **default** bydefault.Thedefaultmodifierisaccessibleonlywithin package.

Exampleofdefaultaccessmodifier

In this example, we have created two packagespack and mypack. We are accessing the Aclass from outside its package, since A class is not public, so it cannot be accessed from outsidethepackage.

```
//save by
A.javapackage
pack;classA{
  voidmsg(){System.out.println("Hello");}
}

//save by
B.javapackage
```

```
mypack;importp
ack.*;
```

```
classB{
 public static void main(String
 args[]){A obj = new A();//Compile
 Time
 Errorobj.msg();//CompileTimeError}}
```

In theabove example, the scope of class A and its method msg() is default so it cannot beaccessedfrom outsidethepackage.

3) *protectedaccess modifier*

The **protected access modifier** is accessible within package and outside the package but throughinheritanceonly.

The protected access modifier can be applied on the data member, method and constructor. It can'tbeapplied on the class.

Exampleofprotectedaccessmodifier

Inthisexample,wehavecreatedthetwopackagespackandmypack.TheAclassofpackpackageispublic,so canbeaccessedfromoutsidethepackage.Butmsgmethodofthispackageisdeclaredasprotected,soit canbeaccessedfrom outsidethe classonlythroughinheritance.

```
//save by
A.javapackage
pack;publicclas
sA{
protectedvoid msg(){System.out.println("Hello");}}
//save by
B.javapackage
mypack;import
pack.*;classBexten
dsA{
 public static void main(String
 args[]){Bobj =newB();
 obj.msg();
 }}
```

Output:Hello

4) *publicaccessmodifier*

The**publicaccessmodifier** isaccessibleeverywhere.Ithasthewidestscopeamongallothermodifiers.

Exampleofpublicaccess modifier

```
//save by
A.javapackage
pack;publicclas
sA{
publicvoidmsg(){System.out.println("Hello");}}
//save by
B.javapackage
mypack;import
pack.*;classB{
  public static void main(String
   args[]){Aobj =newA();
   obj.msg();
  }}
```

Output:Hello

Understandingalljavaaccess modifiers

Let'sunderstandtheaccessmodifiers byasimpletable.

AccessModifier	within class	withinpackage	outside package bysubclassonly	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

thiskeywordinjava

Usageof javathiskeyword

Hereis giventhe6usage ofjavathiskeyword.

1. thiscanbeusedtorefer current classinstancevariable.
2. thiscanbeusedto invokecurrentclass method(implicitly)
3. this()can beusedto invokecurrent classconstructor.
4. thiscanbepassed asan argumentinthemethodcall.
5. thiscanbepassed asargument intheconstructor call.
6. thiscanbeusedtoreturnthecurrent classinstancefrom themethod.

```
class
Student{int
rollno;String
name;floatfee
;
Student(introllno,Stringname,floatfee){
this.rollno=rollno;
this.name=name;t
his.fee=fee;
}
voiddisplay(){System.out.println(rollno+""+name+""+fee);}
}
classTestThis2{
public static void main(String
args[]){Student s1=new
Student(111,"ankit",5000f);Student s2=new
Student(112,"sumit",6000f);s1.display();
s2.display();
}}
```

Output:

```
111 ankit5000
112 sumit 6000
```

Differencebetweenconstructorandmethodin java

JavaConstructor	*JavaMethod*
Constructorisused toinitializethestateofan object.	Methodisusedtoexposebehaviourofan object.
Constructormustnothavereturn type	Methodmusthavereturn type
Constructorisinvokedimplicitly.	Methodisinvokedexplicitly.
Thejavacompilerprovidesadefaultconstructorifyoudon't haveanyconstructor.	Method is not provided by compiler inanycase.

Constructornamemustbesameas the classname.	Methodnamemayormaynotbe sameasclassname.

Therearemanydifferencesbetweenconstructorsand methods.Theyare givenbelow

ConstructorOverloadinginJava

Constructor overloading is a technique in Java in which a class can have any number ofconstructors that differ in parameter lists.The compiler differentiates these constructors bytakinginto account thenumberof parameters in the list and their type.

ExampleofConstructorOverloading

```
class Student5{
int id;String
name;int age
;
Student5(int i,String
n){id =i;
name=n;
}
Student5(int i,String n,int
a){id =i;
name =
n;age=a;
}
void display(){System.out.println(id+""+name+""+age);}

public static void main(String
args[]){Student5 s1 = new
Student5(111,"Karan");Student5 s2 = new
Student5(222,"Aryan",25);s1.display();
s2.display();
}
}
```

Output:

```
111Karan0
222Aryan25
```

MethodOverloadinginjava

Ifaclasshasmultiplemethodshavingsamenamebutdifferentinparameters,itisknownas**Method Overloading**.

Ifwehavetoperformonlyoneoperation,havingsamenameofthemethodsincreasesthereadabilityof theprogram.

MethodOverloading:changingno.ofarguments

Inthisexample,wehavecreatedtwomethods,firstadd()methodperformsadditionoftwonumbersandsecond addmethod performs addition of threenumbers.

Inthisexample,wearecreatingstaticmethodssothatwedon'tneedtocreateinstanceforcallingmethods.

```
class Adder{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));Syst
em.out.println(Adder.add(11,11,11));
}}
```

Output:

```
22
33
```

MethodOverloading:changingdatatypeofarguments

Inthisexample,wehavecreatedtwomethodsthatdiffersindatatype.Thefirstaddmethodreceivestwo integer arguments andsecond add methodreceives twodoublearguments.

RecursioninJava

Recursion in java is a process in which a method calls itself continuously. A method in java thatcallsitself is called recursive method.

JavaRecursion Example1: FactorialNumber

```
publicclassRecursionExample3 {
 staticintfactorial(intn){
  if  (n  ==
   1)return
   1;else
  return(n*factorial(n-1));
  }}
public  static  void  main(String[]  args)
{System.out.println("Factorialof5is:"+factorial(5));
}}
```

Output:

```
Factorialof5 is:120
```

JavaGarbageCollection

Injava, garbagemeansunreferencedobjects.

GarbageCollectionisprocessofreclaimingtheruntimeunusedmemoryautomatically.Inotherwords, it is a wayto destroythe unusedobjects.

Todoso,wewereusingfree()functioninClanguageanddelete()inC++.But,injavaitisperformed automatically.So, javaprovidesbettermemorymanagement.

Advantageof Garbage Collection

- Itmakesjava**memoryefficient** becausegarbagecollectorremovestheunreferencedobjectsfrom heap memory.
- Itis**automaticallydone**bythegarbagecollector(apartofJVM)sowedon'tneedtomakeextraeffort s.

gc() method

The gc() method is used to invoke the garbage collector to perform cleanup processing. Thegc()is found in System and Runtime classes.

```
publicstatic void gc(){}
```

Simple Example of garbage collection in javapublicclass TestGarbage1{

```
publicvoidfinalize(){System.out.println("objectisgarbagecollected");}
public static void main(String
args[]){TestGarbage1 s1=new
TestGarbage1();TestGarbage1 s2=new
TestGarbage1();s1=null;
s2=null;Syst
em.gc();
}}
```

```
objectisgarbagecollectedo
bjectisgarbagecollected
```

JavaString

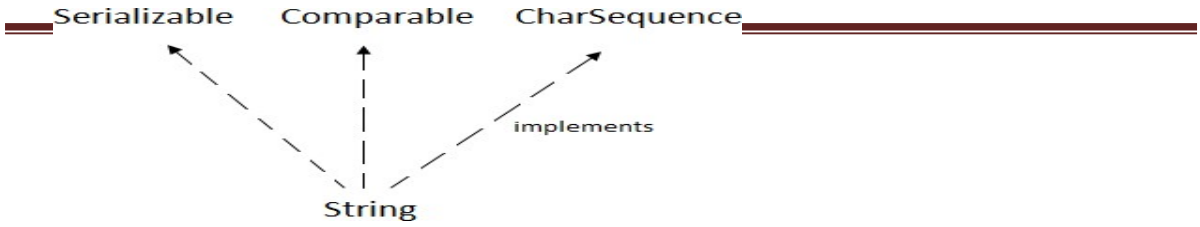
stringisbasicallyanobjectthatrepresentssequenceofcharvalues.Anarrayofcharactersworkssameas javastring.Forexample:

```
1.char[]ch={'j','a','v','a','t','p','o','i','n','t'};
2.String s=new
```

String(ch);ssameas:

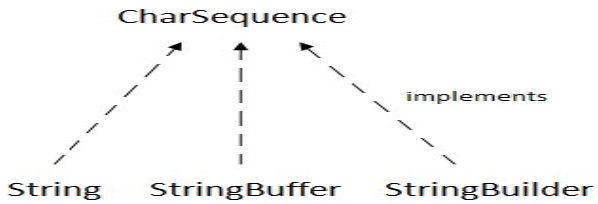
1. Strings="javatpoint";
2. **Java String** class provides a lot of methods to perform operations onstring such ascompare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring()etc.

3. The java.lang.String classimplements *Serializable*,*Comparable*and *CharSequence*interfaces.



CharSequenceInterface

The CharSequence interface is used to represent sequence of characters. It is implemented byString, StringBuffer and StringBuilder classes. It means, we can create string in java by usingthese3 classes.



The java String is immutable i.e. it cannot be changed. Whenever we change anystring,anewinstanceiscreated.Formutablestring,youcanuseStringBufferandStringBuilderclasses.

Therearetwoways tocreate Stringobject:

1. Bystringliteral
2. Bynewkeyword

StringLiteral

JavaStringliteral is created byusingdouble quotes. ForExample:

```
1.Strings="welcome";
```

Each time you create a string literal, the JVM checks the string constant pool first. If the stringalready exists in the pool, a reference to the pooled instance is returned. If string doesn't exist inthepool, anew stringinstanceis createdand placed in thepool. Forexample:

```
1. Strings1="Welcome";
2. Strings2="Welcome";//will not createnewinstance
```

Bynewkeyword

```
1.Strings=new String("Welcome");//createstwoobjectsandonereferencevariable
```

In such case, JVM will create a new string object in normal (non pool) heap memory and theliteral "Welcome" will be placed in the string constant pool. The variable s will refer to the objectinheap (non pool).

JavaStringExample

```
publicclassStringExample{
publicstaticvoid main(Stringargs[]){
Strings1="java";//creatingstringbyjavastringliteral
charch[]={'s','t','r','i','n','g','s'};
Strings2=newString(ch);//convertingchararraytostring
String s3=new String("example");//creating java string by new
keywordSystem.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}}
```

Java stringse xample

ImmutableStringin Java

In java, **string objects are immutable**. Immutable simply means unmodifiable or

unchangeable.Once string object is created its data or state can't be changed but a new string

object is created.Let's tryto understand theimmutabilityconcept bytheexamplegiven below:

```
classTestimmutablestring{
  public static void main(String
    args[]){Strings="Sachin";
    s.concat(" Tendulkar");//concat() method appends the string at the
    endSystem.out.println(s);//willprintSachinbecause stringsareimmutableobjects
  }}
Output:Sachin
classTestimmutablestring1{
 public static void main(String
   args[]){String s="Sachin";
   s=s.concat("
   Tendulkar");System.out.pr
   intln(s);
 }}Output:Sachin Tendulkar
```

InheritanceinJava

Inheritance in java is a mechanism in which one object acquires all the properties and behaviorsofparent object.Inheritancerepresents the**IS-A relationship**, alsoknownas*parent-child*relationship.

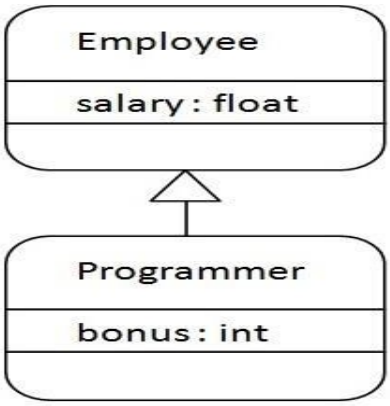
Whyuseinheritanceinjava

- ForMethod Overriding(so runtimepolymorphismcan beachieved).
- ForCodeReusability.

SyntaxofJavaInheritance

```
1.class Subclass-name extends Superclass-
name2.{
3.      //methods and
fields4.}
```

The**extendskeyword**indicatesthatyouaremakinganewclassthatderivesfromanexistingclass.Themeaningof "extends"is to increasethefunctionality.



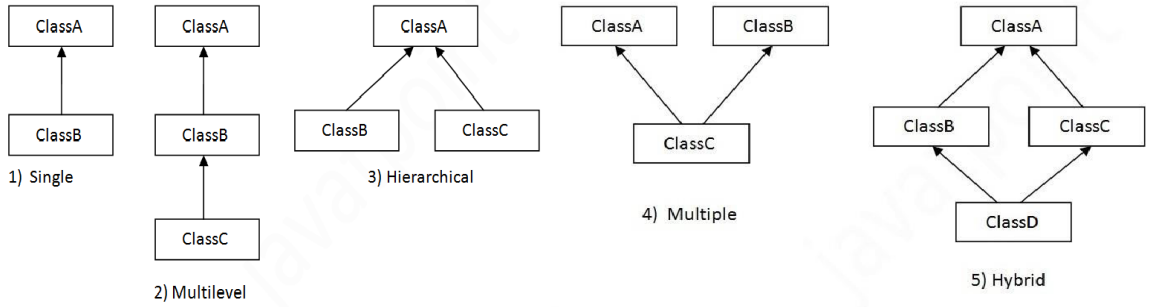

```
classEmployee{
 floatsalary=40000;
}
classProgrammerextendsEmployee{
 intbonus=10000;
 public static void main(String
   args[]){Programmerp=new
   Programmer();
   System.out.println("Programmer salary
   is:"+p.salary);System.out.println("BonusofProgrammeris
```

```
  :"+p.bonus);
}}
```

```
Programmersalaryis:40000.0
```

Bonusofprogrammer is:10000

Types ofinheritanceinjava



SingleInheritanceExample

File:TestInheritance.java

```
classAnimal{
voideat(){System.out.println("eating...");}
}
classDogextendsAnimal{
voidbark(){System.out.println("barking...");}
}
classTestInheritance{
public static void main(String
args[]){Dogd=newDog();
d.bark();
d.eat();
}}
```

Output:

```
barking...
eating...
```

MultilevelInheritanceExample

File:TestInheritance2.java

```
classAnimal{
voideat(){System.out.println("eating...");}
}
classDogextendsAnimal{
voidbark(){System.out.println("barking...");}
}
classBabyDogextendsDog{
voidweep(){System.out.println("weeping...");}
}
classTestInheritance2{
```

```
public static void main(String
args[]){BabyDogd=newBabyDog();d.
weep();
d.bark();
d.eat();
}}
```

Output:

```
weeping...
barking...
eating...
```

HierarchicalInheritanceExample

File:TestInheritance3.java

```
classAnimal{
voideat(){System.out.println("eating...");}
}
classDogextends Animal{
voidbark(){System.out.println("barking...");}
}
classCatextends Animal{
voidmeow(){System.out.println("meowing...");}
}
classTestInheritance3{
public static void main(String
args[]){Catc=newCat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

Output:

```
meowing...
eating...
```

Memberaccess andInheritance

Asubclassincludes allofthemembersof itssuperclassbutit cannotaccess thosemembers ofthe super class that have been declared as private. Attempt to access a private variable wouldcause compilation error as it causes access violation. The variables declared as private, is onlyaccessiblebyother members ofits own class. Subclass havenoaccess to it.

superkeywordinjava

The**super**keywordinjavaisareferencevariablewhichisusedtoreferimmediateparentclassobject.

Wheneveryoucreatetheinstanceofsubclass,aninstanceofparentclassiscreatedimplicitlywhichis referredbysuper referencevariable.

Usageof javasuperKeyword

1. supercanbe usedto referimmediateparentclassinstance variable.
2. supercanbe usedto invoke immediate parentclassmethod.
3. super()canbeusedtoinvokeimmediateparent classconstructor.

superisusedtoreferimmediateparentclassinstancevariable.

```
classAnimal{
Stringcolor="white";
}
class Dog extends
Animal{Stringcolor="black
";
voidprintColor(){System.out.println(color);//prints
colorofDogclass
System.out.println(super.color);//printscolorofAnimalclass
}
}
classTestSuper1{
public static void main(String
args[]){Dogd=newDog();
```

```
d.printColor();
}}
```

Output:

```
black
white
```

FinalKeywordinJava

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context.Finalcanbe:

1. variable
2. method
3. class

The final keyword can be applied with the variables, a final variable that have no value it is called blankfinal variable or uninitialized final variable. It can be initialized in the constructor only. The blank finalvariablecan bestatic alsowhich willbeinitialized inthe static blockonly.

ObjectclassinJava

The **Object class** is the parent class of all the classes in java by default. In other words, it is thetopmostclass ofjava.

The Object class is beneficial if you want to refer any object whose type you don't know. Noticethatparent classreferencevariablecanrefer thechild classobject, knowas upcasting.

Let's take an example, there is getObject() method that returns an object but it can be of any typelikeEmployee,Studentetc, wecan useObject classreferenceto referthat object.Forexample:

```
1. Objectobj=getObject();//wedon'tknowwhatobjectwillbereturnedfromthismethod
```

The Object class provides some common behaviors to all the objects such as object can becompared,object can becloned, object can benotifiedetc.

MethodOverridinginJava

If subclass (child class) has the same method as declared in the parent class, it is knownas**method overriding injava**.

Usageof JavaMethodOverriding

- Methodoverridingisusedtoprovidespecificimplementationofamethodthatisalreadyprovided byits superclass.
- Methodoverridingisused forruntimepolymorphism

RulesforJavaMethodOverriding

1. methodmust havesame nameas in theparentclass
2. methodmust havesame parameteras in theparent class.
3. mustbeIS-Arelationship(inheritance).

Example of method overridingClassVehicle{

```
voidrun(){System.out.println("Vehicleisrunning");}
}
classBike2extendsVehicle{
voidrun(){System.out.println("Bikeisrunningsafely");}
public static void main(String
args[]){Bike2obj =newBike2();
obj.run();
}
```

Output:Bike is runningsafely

```
1.classBank{
intgetRateOfInterest(){return0;}
}
classSBIextendsBank{
intgetRateOfInterest(){return8;}
}
 class ICICIextendsBank{
intgetRateOfInterest(){return7;}
}
classAXISextendsBank{
intgetRateOfInterest(){return9;}
}
classTest2{
public static void main(String
args[]){SBIs=newSBI();
ICICI i=new
ICICI();AXISa=newA
XIS();
System.out.println("SBI Rate of Interest:
"+s.getRateOfInterest());System.out.println("ICICI Rate of Interest:
"+i.getRateOfInterest());System.out.println("AXISRateofInterest:"+
a.getRateOfInterest());
}}
```

Output:
SBIRateofInterest:8

```
ICICIRateofInterest:7A
XISRateofInterest:9
```

AbstractclassinJava

A class that is declared with abstract keyword is known as abstract class in java. It can haveabstract and non-abstract methods (method with body). It needs to be extended and its methodimplemented. Itcannot beinstantiated.

Exampleabstractclass

```
1.abstractclass A{}
```

abstractmethod

```
1.abstractvoidprintStatus();//nobodyandabstract
```

Exampleofabstractclassthathasabstractmethod

```
abstractclassBike{
  abstractvoid run();
}
classHonda4extendsBike{
voidrun(){System.out.println("runningsafely..");}
public static void main(String
 args[]){Bikeobj =newHonda4();
 obj.run();
}
1.}
```

```
runningsafely..
```

Interface inJava

An**interfacein java** isablueprintofaclass.It hasstaticconstantsandabstractmethods.

The interface in java is **a mechanism to achieve abstraction**. There can be only abstract methodsin the java interface not method body. It is used to achieve abstraction and multiple inheritance inJava.

Java Interface also **represents IS-A**

relationship.Itcannotbeinstantiated

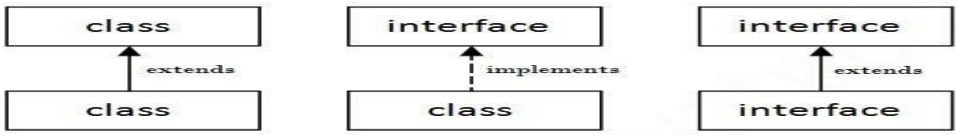
justlikeabstract class.

Therearemainlythreereasons to useinterface. Theyare given below.

- Itisusedto achieveabstraction.
- Byinterface,wecansupport the functionalityof multiple inheritance.
- Itcanbeusedtoachieve loosecoupling.

Internaladditionbycompiler

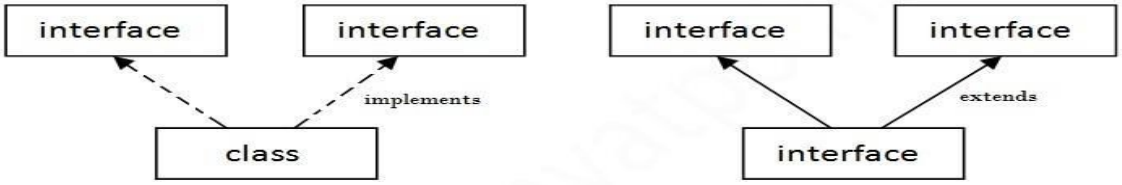
Understandingrelationshipbetweenclassesandinterfaces




```
//Interfacedeclaration: byfirstuser
interfaceDrawable{
voiddraw();
}
//Implementation: bysecond user
classRectangleimplementsDrawable{
publicvoiddraw(){System.out.println("drawingrectangle");}
}
classCircleimplementsDrawable{
publicvoiddraw(){System.out.println("drawingcircle");}
}
//Usinginterface: bythirduser
classTestInterface1{
publicstaticvoid main(Stringargs[]){
Drawable d=new Circle();//In real scenario, object is provided by method e.g.
getDrawable()d.draw();
}}
```

Output:drawingcircle

MultipleinheritanceinJavabyinterface



Multiple Inheritance in Java

```
interfacePrintable{
```

```
voidprint();
}
interfaceShowable{
void show();
}
classA7implementsPrintable,Showable{
public void
print(){System.out.println("Hello");}public void
show(){System.out.println("Welcome");}publicstatic
voidmain(Stringargs[]){
A7 obj = new
A7();obj.print();
obj.show();
 }}
```

Output:Hello
Welcome

Abstractclass	Interface
1)Abstractclasscan**haveabstractandnon-abstract** methods.	Interface can have **only abstract** methods. SinceJava8,itcanhave**defaultandstaticmethods**also.
2) Abstract class **doesn't supportmultipleinheritance**.	Interface**supportsmultipleinheritance**.
3)Abstractclass**canhavefinal,non-final,staticandnon-staticvariables**.	Interfacehas**onlystatic and finalvariables**.
4) Abstract class **can provide theimplementation ofinterface**.	Interface**can'tprovidetheimplementationofabstractclass**.
5)The**abstractkeyword**isusedtodeclare abstract class.	The**interface keyword** used to declare isinterface.

6)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

JavaInnerClasses

Javainnerclass ornestedclassis aclasswhich isdeclaredinsidethe classorinterface.

Weuseinnerclassestologicallygroupclassesandinterfacesinoneplacesothatitcanbemorereadableandmaintainable.

SyntaxofInnerclass

```
1. classJava_Outer_class{
2.  //code
3.  classJava_Inner_class{
```

```
4.    //code
5.  }}
```

Advantageof javainnerclasses

Therearebasicallythree advantages of inner classesin java. Theyareas follows:

1) Nestedclassesrepresentaspecialtypeofrelationshipthatis**itcanaccessallthemembers(data members and methods)of outerclass**includingprivate.

2) Nestedclassesareused **todevelopmorereadableandmaintainablecode**becauseitlogicallygroupclasses and interfaces in oneplaceonly.

3) **CodeOptimization**:Itrequireslesscodetowrite.

Differencebetweennested classand innerclass inJava

Innerclass isapart ofnested class.Non-static nestedclasses areknownasinnerclasses.

TypesofNestedclasses

Therearetwotypesofnestedclassesnon-staticandstaticnestedclasses.Thenon-staticnestedclassesarealso knownas innerclasses.

- Non-staticnestedclass(innerclass)
    1. Memberinner class
    2. Anonymousinnerclass
    3. Localinner class
- Staticnestedclass

JavaPackage

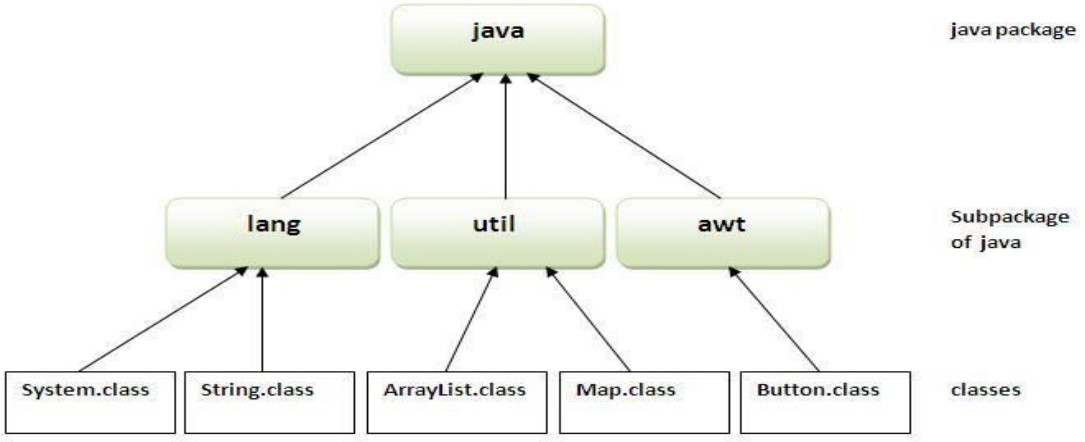
A**javapackage** is agroup of similar typesofclasses,interfaces and sub-packages.Package in java can be categorized in two form, built-in package and user-defined package.There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.**AdvantageofJavaPackage**

1) Javapackageisusedtocategorizetheclassesandinterfacessothattheycanbeeasilymaintained.

2) Javapackageprovides accessprotection.

3) Javapackageremovesnamingcollision.

```
packagemypack;
publicclassSimple{
 public static void main(String
   args[]){System.out.println("Welcometopackage"
   );
  }}
```



Howto compile javapackage

If you are not using any IDE, you need to follow the **syntax** given

below:javac-d directoryjavafilename

How torunjavapackageprogram

ToCompile:javac-d. Simple.java

ToRun:javamypack.Simple

Usingfully qualifiedname

Exampleofpackagebyimport fullyqualified name

```
//save by
A.javapackage
pack;publicclas
sA{
  publicvoidmsg(){System.out.println("Hello");}}
//save by
B.javapackage
mypack;classB{
  publicstaticvoid main(Stringargs[]){
   pack.A obj = new pack.A();//using fully qualified
   nameobj.msg();
  }
}
```

Output:Hello

UNIT-3

ExceptionHandling

The **exceptionhandlinginjava** isoneofthepowerful *mechanismtohandletheruntimeerrors*so that normal flow ofthe application canbemaintained.

Whatis exception

In java, exception is an event that disrupts the normal flow of the program. It is an object which isthrownat runtime.

AdvantageofExceptionHandling

The core advantage of exception handling is **to maintain the normal flow of the application**.Exception normally disrupts the normal flow of the application that is why we use exceptionhandling.

TypesofException

Therearemainlytwotypesofexceptions:checkedanduncheckedwhereerrorisconsideredasuncheckedexception. Thesunmicrosystem says therearethreetypes of exceptions:

1. CheckedException
2. UncheckedException
3. Error

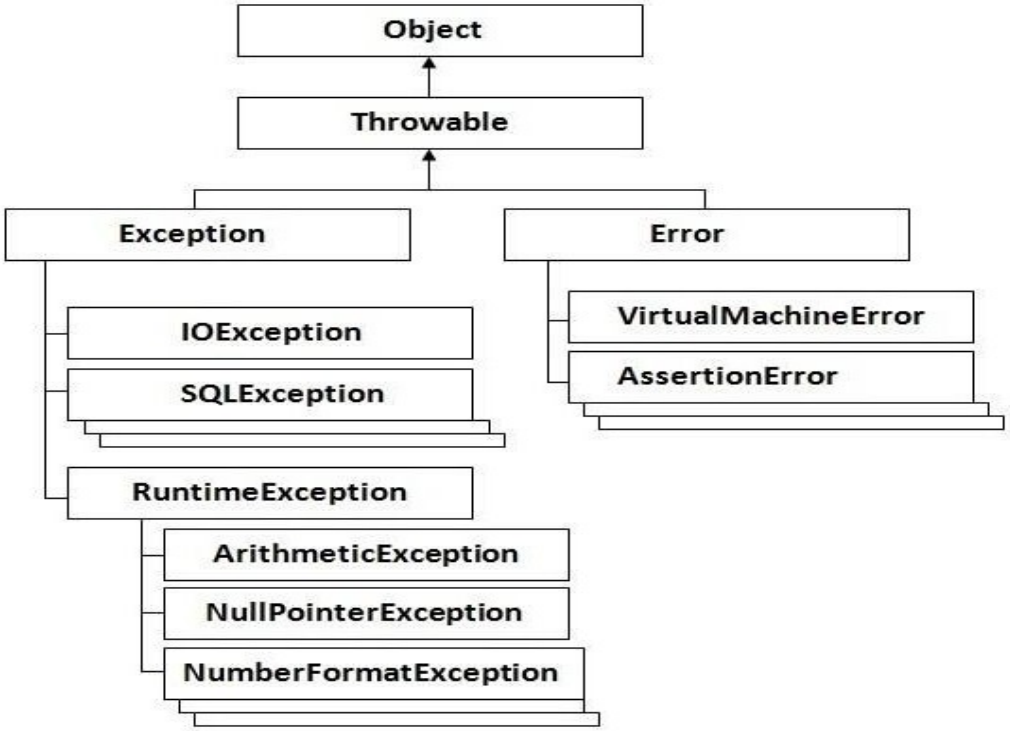
Differencebetweencheckedanduncheckedexceptions

1) Checked Exception: The classes that extend Throwable class except RuntimeException and Errorare known as checked exceptions e.g.IOException, SQLException etc. Checkedexceptions are checkedatcompile-time.

2) UncheckedException:
TheclassesthatextendRuntimeExceptionareknownasuncheckedexceptionse.g.ArithmeticException,NullPointerException,ArrayIndexOutOfBoundsExceptionetc.Unchecked exceptions are notchecked atcompile-timeratherthey are checkedatruntime.

3) Error:Errorisirrecoverablee.g.OutOfMemoryError,VirtualMachineError,AssertionErroretc.

HierarchyofJavaExceptionclasses



CheckedandUnCheckedExceptions

Checked Exceptions	Unchecked Exceptions
• Exception which are checked at Compile time called Checked Exception • If a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword	• Exceptions whose handling is NOT verified during Compile time. • These exceptions are handled at run-time i.e., by JVM after they occurred by using the try and catch block
• Examples: o IOException o SQLException o DataAccessException o ClassNotFoundException o InvocationTargetException o MalformedURLException	• Examples o NullPointerException o ArrayIndexOutOfBound o IllegalArgumentException o IllegalStateException

Javatryblock

Java try block is used to enclose the code that might throw an exception. It must be used withinthe method.

Javatryblock must befollowed byeither catch or finallyblock.

Syntaxofjavatry-catch

```
1. try{
2. //codethat maythrowexception
3. }catch(Exception_class_Name
```

ref){}Syntax of try-finallyblock

```
1. try{
2. //codethat maythrowexception
3. }finally{}
```

Javacatchblock

Java catch block is used to handle the Exception. It must be used after the try block

only.Youcan usemultiple catch block with asingletry.

Problemwithoutexceptionhandling

Let'stryto understand theproblem if wedon't usetry-catch block.

```
publicclassTesttrycatch1{
  publicstaticvoidmain(Stringargs[]){int
    data=50/0;//may throw
    exceptionSystem.out.println("restofthec
    ode...");
}}
```

Output:

```
Exceptioninthreadmainjava.lang.ArithmeticException:/byzero
```

As displayed in the above example, rest of the code is not executed (in such case, rest of thecode...statement is not printed).

There can be 100 lines of code after exception. So all the code after exception will not beexecuted.

Solutionbyexceptionhandling

Let'sseethe solution of aboveproblem byjavatry-catchblock.

```
publicclassTesttrycatch2{
```

```
 publicstaticvoid main(Stringargs[]){
  try{
    intdata=50/0;
  }catch(ArithmeticException
  e){System.out.println(e);}System.out.println("rest
  ofthecode...");
}}
```

1.Output:

```
Exceptioninthreadmainjava.lang.ArithmeticException:/byzerorest
of the code...
```

Now, as displayed in the above example, rest of the code is executed i.e. rest of the code...statement isprinted.

JavaMulticatchblock

Ifyouhavetoperformdifferenttasksattheoccurrenceofdifferent Exceptions,usejavamulticatchblock.

Let'sseeasimpleexampleofjavamulti-catchblock.

```
1.  publicclassTestMultipleCatchBlock{
2.    publicstaticvoidmain(Stringargs[]){
3.     try{
4.      int a[]=new
int[5];5.    a[5]=30/0;
6.     }
7.   catch(ArithmeticExceptione){System.out.println("task1iscompleted");}
8.   catch(ArrayIndexOutOfBoundsException e){System.out.println("task 2
completed");9.}
10.    catch(Exceptione){System.out.println("commontaskcompleted");
11. }
12.    System.out.println("rest of the
code...");13.}}
```

```
Output:task1
      completedrestof
```

Javanestedtryexample

Let'sseeasimple exampleof javanested tryblock.

```
classExcep6{
 publicstaticvoid main(Stringargs[]){
  try{t
    ry{
     System.out.println("goingtodivide");
     intb =39/0;
    }catch(ArithmeticExceptione){System.out.println(e);}
```

```
try{
```

```
    int a[]=new
    int[5];a[5]=4;
    }catch(ArrayIndexOutOfBoundsException
    e){System.out.println(e);}System.out.println("otherstatement);
   }catch(Exception
   e){System.out.println("handeled");}System.out.println
   ("normalflow..");
  }
 1.}
```

Java finallyblock

Javafinallyblockisablockthatisused*toexecuteimportantcode*suchasclosingconnection,streametc.

Java finally block is always executed whether exception is handled or

not.Javafinallyblock follows tryorcatch block.

UsageofJavafinally

Case1

Let's see the java finally example where **exception doesn't**

```
occur.classTestFinallyBlock{
 publicstaticvoid main(Stringargs[]){
 try{
  int
  data=25/5;System.out.pr
  intln(data);
 }
 catch(NullPointerException
 e){System.out.println(e);}finally{System.out.println("finally
 block is always
 executed");}System.out.println("restofthecode...");
 }
}
```

Output:5
finallyblockisalwaysexecutedre
stof the code...

Javathrowkeyword

TheJavathrowkeywordis used toexplicitlythrow an exception.

We can throw either checked or uncheked exception in java by throw keyword. The throwkeywordismainlyusedtothrow customexception.Wewillseecustom exceptionslater.

Thesyntaxofjavathrow keywordisgivenbelow.

1.**throw**exception;

Javathrowkeywordexample

Inthisexample,wehave createdthevalidatemethodthattakesintegervalueasaparameter. Ifthe age is less than 18, we are throwing the ArithmeticException otherwise print a messagewelcometo vote.

```
1.publicclassTestThrow1{
    staticvoid validate(intage){
    if(age<18)
    thrownewArithmeticException("notvalid");
    else
    System.out.println("welcometovote");
    }
    public static void main(String
    args[]){validate(13);
    System.out.println("restofthecode...");
    }}
```

Output:

```
Exceptioninthreadmainjava.lang.ArithmeticException:notvalid
```

Javathrowskeyword

The **Javathrowskeyword** isusedtodeclareanexception.Itgivesaninformationtotheprogrammer that there may occur an exception so it is better for the programmer to provide theexceptionhandlingcodeso that normal flow can bemaintained.

ExceptionHandlingismainlyusedtohandlethecheckedexceptions.Ifthereoccursanyuncheckedexcepti onsuchasNullPointerException,itisprogrammersfaultthatheisnotperformingcheck up beforethe codebeingused.

Syntaxofjavathrows

```
1. return_typemethod_name()throwsexception_class_name{
2. //method
code3.}
4.
```

Javathrowsexample

Let's see the example of java throws clause which describes that checked exceptions can bepropagated bythrows keyword.

```
importjava.io.IOException;
classTestthrows1{
```

```
voidm()throwsIOException{
  thrownew IOException("deviceerror");//checkedexception
```

```
    }
    void n()throws
      IOException{m();
    }
    void p(){
     try{
      n();
     }catch(Exceptione){System.out.println("exceptionhandled");}
    }
    public static void main(String
     args[]){Testthrows1 obj=new
     Testthrows1();obj.p();
     System.out.println("normal
flow...");}}Output:
```

```
exception
handlednormalflo
```

JavaCustomException

IfyouarecreatingyourownExceptionthatisknownascustomexceptionoruser-definedexception.Javacustomexceptions areusedto customizethe exceptionaccordingto userneed.

By the help of custom exception, you can have your own exception and

message.Let'sseeasimple exampleof java custom exception.

```
class InvalidAgeException extends
 Exception{InvalidAgeException(Strings){
  super(s);
 }}
classTestCustomException1{
  staticvoidvalidate(intage)throws InvalidAgeException{
    if(age<18)
     thrownew InvalidAgeException("notvalid");

    else
     System.out.println("welcometovote");
  }
  publicstaticvoid main(Stringargs[]){
    try{validate
    (13);
    }catch(Exceptionm){System.out.println("Exceptionoccured: "+m);}

    System.out.println("restofthecode...");
 }}
```

Output:Exceptionoccured: InvalidAgeException:notvalidrestofthecode...

Multithreading

Multithreadingin javaisaprocessof executingmultiplethreadssimultaneously.

Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing andmultithreading,both areusedto achievemultitasking.

But we use multithreading than multiprocessing because threads share a common memory area.They don't allocate separate memory area so saves memory, and context-switching between thethreadstakes less timethan process.

JavaMultithreadingis mostlyusedingames,animation etc.

AdvantagesofJavaMultithreading

1) It**doesn'tblocktheuser** becausethreadsareindependentand youcanperformmultipleoperationsat sametime.

2) *You**canperformmanyoperations togethersoitsaves time**.*

3) Threads are**independent** soitdoesn'taffectotherthreadsifexceptionoccurinasinglethread.

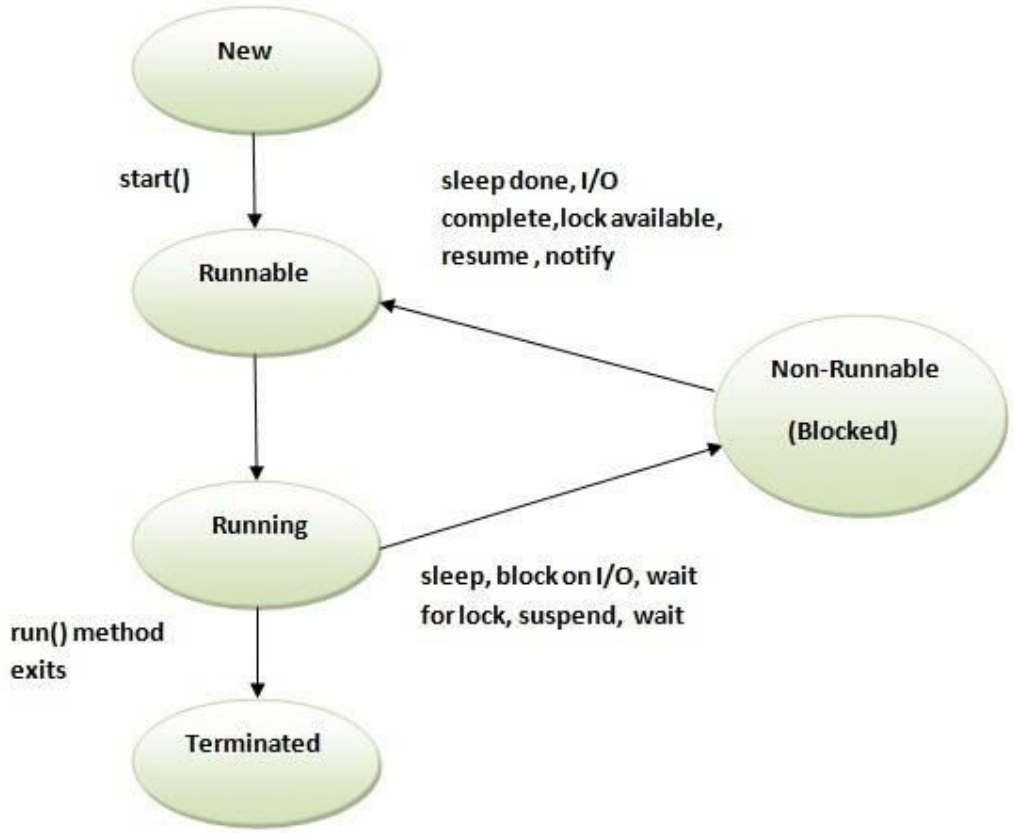
LifecycleofaThread(ThreadStates)

A thread can be in one of the five states. According to sun, there is only 4 states in **thread lifecyclein java** new, runnable, non-runnableandterminated. Thereis norunningstate.

Butforbetter understandingthe threads, we areexplainingit in the5 states.

Thelifecycleof thethread in javaiscontrolled byJVM. Thejavathread states areas follows:

1. New
2. Runnable
3. Running
4. Non-Runnable(Blocked)
5. Terminated



Howto createthread

Thereare twowaystocreateathread:

1. ByextendingThreadclass
2. ByimplementingRunnable interface.

Threadclass:

Thread class provide constructors and methods to create and perform operations on athread.Threadclass extendsObject class andimplements Runnableinterface.

CommonlyusedConstructorsofThreadclass:

- Thread()
- Thread(Stringname)
- Thread(Runnabler)
- Thread(Runnabler,String name)

CommonlyusedmethodsofThreadclass:

1. **publicvoidrun():** isusedtoperformactionforathread.
2. **publicvoidstart():**startstheexecutionofthe thread.JVMcallstherun()methodonthe thread.
3. **publicvoidsleep(longmiliseconds):**Causesthecurrentlyexecutingthreadtosleep(temporarilycease execution) forthe specifiednumber ofmilliseconds.
4. **publicvoidjoin():** waitsforathreadtodie.
5. **publicvoidjoin(long miliseconds):** waitsforathreadto dieforthespecifiedmiliseconds.
6. **publicintgetPriority():**returns thepriorityof thethread.
7. **publicintsetPriority(intpriority):**changesthe priorityofthethread.
8. **publicStringgetName():**returnsthenameofthe thread.
9. **publicvoidsetName(Stringname):**changesthenameofthethread.
10. **publicThreadcurrentThread():**returnsthereferenceof currentlyexecutingthread.
11. **publicintgetId():**returnstheidofthethread.
12. **publicThread.StategetState():**returnsthestateofthethread.
13. **publicbooleanisAlive():**tests ifthethread isalive.
14. **publicvoidyield():**causesthecurrentlyexecutingthreadobjecttotemporarilypauseandallowotherthrea ds to execute.
15. **publicvoidsuspend():**isusedtosuspendthethread(depricated).
16. **publicvoidresume():** is usedtoresumethesuspendedthread(depricated).
17. **publicvoidstop():**isusedtostopthethread(depricated).
18. **publicbooleanisDaemon():** testsifthethreadisadaemonthread.
19. **publicvoidsetDaemon(booleanb):** marksthethreadasdaemonoruserthread.
20. **publicvoidinterrupt():** interruptsthethread.
21. **publicbooleanisInterrupted():**testsifthethreadhasbeeninterrupted.
22. **publicstaticbooleaninterrupted():** testsifthecurrentthreadhasbeeninterrupted.

Runnableinterface:

The Runnable interface should be implemented by any class whose instances are intended to beexecutedbyathread. Runnable interfacehaveonlyonemethod named run().

1. **publicvoidrun():** isusedtoperformactionforathread.

Startingathread:

start() method of Thread class is used to start a newly created thread. It performs followingtasks:

- Anewthread starts(withnew callstack).
- Thethread moves fromNew state tothe Runnablestate.
- Whenthethreadgets achanceto execute, itstarget run() method will run.

JavaThreadExamplebyextendingThreadclass

```
classMultiextendsThread{
publicvoidrun(){System.out.println("threa
disrunning...");
}
public static void main(String
args[]){Multi t1=newMulti();
t1.start();
 } }
```

Output:threadisrunning...

JavaThreadExamplebyimplementingRunnableinterface

```
classMulti3implementsRunnable{public
voidrun(){System.out.println("threadisru
nning...");
}
public static void main(String
args[]){Multi3m1=newMulti3();
Thread t1 =new
Thread(m1);t1.start();
 }}
```

Output:threadisrunning...

PriorityofaThread(ThreadPriority):

Each thread have a priority. Priorities are represented by a number between 1 and 10. In mostcases, thread schedular schedules the threads according to their priority (known as preemptivescheduling). But it is not guaranteed because it depends on JVM specification that whichschedulingit chooses.

3constantsdefinedinThreadclass:

1. publicstaticintMIN_PRIORITY
2. publicstaticintNORM_PRIORITY
3. publicstaticintMAX_PRIORITY

Defaultpriorityof athreadis5(NORM_PRIORITY).ThevalueofMIN_PRIORITYis1andthevalue ofMAX_PRIORITY is 10.

Example of priority of a Thread:

```
class TestMultiPriority1 extends
Thread{publicvoidrun(){
  System.out.println("running thread name
```

```
 is:"+Thread.currentThread().getName());System.out.println("runningthreadpriorityis:"
 +Thread.currentThread().getPriority());
}publicstaticvoid main(Stringargs[]){
```

```
TestMultiPriority1 m1=new
TestMultiPriority1();TestMultiPriority1 m2=new
TestMultiPriority1();m1.setPriority(Thread.MIN_
PRIORITY);m2.setPriority(Thread.MAX_PRIO
RITY);m1.start();
m2.start();
}}
```

Output:running thread name is:Thread-
0runningthread priorityis:10
running thread name is:Thread-
1runningthread priorityis:1

Javasynchronizedmethod

If you declare any method as synchronized, it is known as synchronized

method.Synchronizedmethod isusedtolockan object for anysharedresource.

When a thread invokes a synchronized method, it automatically acquires the lock for that objectandreleases it when the thread completes its task.

Exampleofinterthreadcommunicationinjava

Let'sseethesimple exampleof interthread communication.

```
classCustomer{
intamount=10000;
synchronized void withdraw(int
amount){System.out.println("going to
withdraw...");if(this.amount<amount){
System.out.println("Less balance;waitingfordeposit...");
try{wait();}catch(Exceptione){}
}
this.amount-
=amount;System.out.println("withdrawcomp
leted...");
}
synchronized void deposit(int
amount){System.out.println("going to
deposit...");this.amount+=amount;System.
out.println("deposit completed...
");notify();
}
}
classTest{
public static void main(String
args[]){final Customer c=new
Customer();newThread(){
publicvoidrun(){c.withdraw(15000);}
```

```
}.start();
newThread(){
```

```
publicvoid run(){c.deposit(10000);}
}
start();
}}
```

```
Output:goingto withdraw...
     Lessbalance;waitingfordeposit...g
     oingto deposit...
     deposit
     completed...withdra
```

ThreadGroupinJava

Javaprovidesaconvenientwaytogroupmultiplethreadsinasingleobject.Insuchway,wecansuspend,resume orinterrupt groupofthreads byasingle method call.

Note:Nowsuspend(),resume()andstop()methodsaredeprecated.

Java thread group is implemented by *java.lang.ThreadGroup*

class.ConstructorsofThreadGroupclass

There areonlytwoconstructors ofThreadGroup class.

ThreadGroup(String
name)ThreadGroup(ThreadGroupparent,Stringna
me)

Let'sseea codeto group multiplethreads.

```
1. ThreadGrouptg1=new ThreadGroup("Group A");
2. Threadt1=new Thread(tg1,new MyRunnable(),"one");
3. Threadt2=newThread(tg1,new MyRunnable(),"two");
4. Threadt3=new Thread(tg1,newMyRunnable(),"three");
```

Nowall3threadsbelongtoonegroup.Here,tg1isthethreadgroupname,MyRunnableistheclassthatimple ments Runnableinterface and"one","two"and"three"arethethreadnames.

Nowwecan interruptall threads byasingle lineof codeonly.

```
1. Thread.currentThread().getThreadGroup().interrupt();
```

Exploringjava.netandjava.text

java.net

The term *network programming* refers to writing programs that execute across multiple devices(computers),in which thedevicesare all connected to eachotherusinganetwork.

The java.net package of the J2SE APIs contains a collection ofclasses and interfaces thatprovide the low-level communication details, allowing you to write programs that focus onsolvingthe problem at hand.

Thejava.netpackageprovides supportforthetwocommonnetworkprotocols−

- **TCP**
  −TCPstandsforTransmissionControlProtocol,whichallowsforreliablecommunicationbetweentwoapplications.TCPistypicallyusedovertheInternetProtocol,which is referred to as TCP/IP.
- **UDP**−UDPstandsforUserDatagramProtocol,aconnection-lessprotocolthatallowsforpackets of data to betransmitted betweenapplications.

Thischaptergivesa goodunderstandingonthefollowingtwo subjects−

- **SocketProgramming**−Thisisthemostwidelyusedconceptin Networkingandithas been explained inverydetail.
- **URLProcessing**−This wouldbecoveredseparately.

java.text

The java.text package is necessary for every java developer to master because it has a lot ofclassesthatishelpfulinformattingsuchasdates,numbers,andmessages.

java.textClasses

Thefollowingaretheclassesavailableforjava.textpackage[tabl

e]

Class|Description

SimpleDateFormat|is a concrete class that helps in formatting and parsing of dates.[/table]

UNIT-4

CollectionFrameworkinJava

Collectionsinjavaisaframeworkthatprovidesanarchitecturetostoreandmanipulatethegroupof objects.

All the operations that you perform on a data such as searching, sorting, insertion, manipulation,deletion etc. can beperformed byJavaCollections.

Java Collection simply means a single unit of objects. Java Collection framework provides manyinterfaces(Set,List,Queue,Dequeetc.)andclasses(ArrayList,Vector,LinkedList,PriorityQueue, HashSet,LinkedHashSet, TreeSetetc).

Whatisframeworkinjava

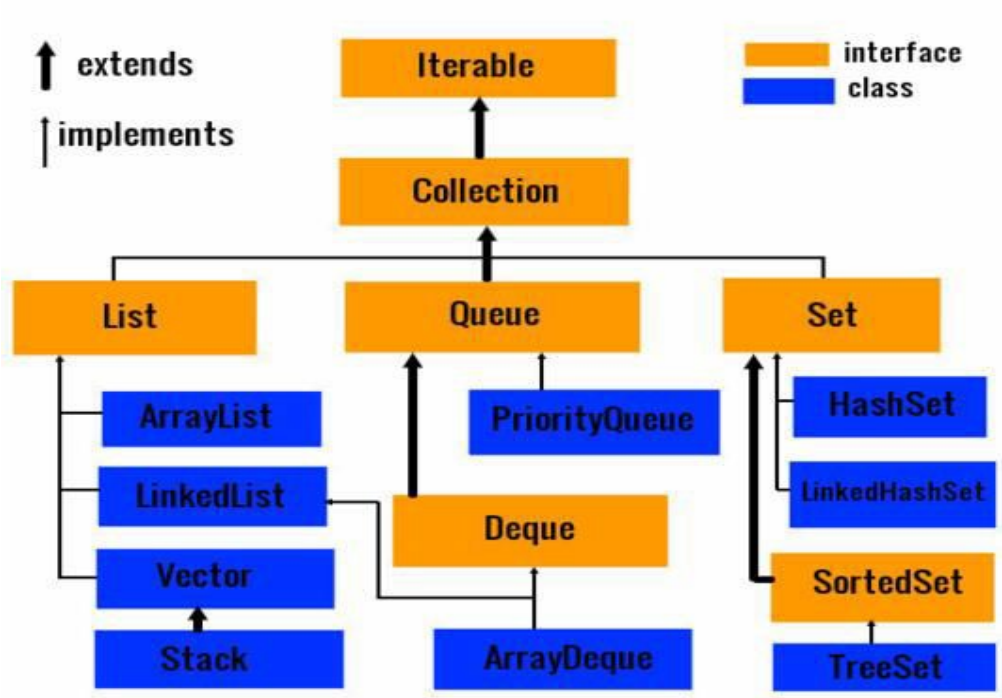
- providesreadymadearchitecture.
- representssetofclassesandinterface.
- is optional.

WhatisCollectionframework

Collection framework represents a unified architecture for storing and manipulating group ofobjects.It has:

1. Interfaces anditsimplementationsi.e.classes
2. Algorithm

HierarchyofCollectionFramework



JavaArrayListclass

JavaArrayListclassusesadynamicarrayforstoringthe elements.ItinheritsAbstractListclassandimplementsList interface.

TheimportantpointsaboutJavaArrayListclassare:

- JavaArrayListclasscancontainduplicateelements.
- JavaArrayListclassmaintainsinsertionorder.
- JavaArrayListclassisnonsynchronized.
- JavaArrayListallowsrandomaccess becausearrayworksat theindexbasis.
- In Java ArrayList class, manipulation is slow because a lot of shifting needs to be occurredif anyelement isremoved from thearraylist.

ArrayListclassdeclaration

Let'sseethedeclarationforjava.util.ArrayListclass.

ConstructorsofJavaArrayList

Constructor	Description
ArrayList()	It is used to build an emptyarraylist.
ArrayList(Collectionc)	Itisusedtobuildanarraylistthatisinitializedwiththeelement sof the collection c.
ArrayList(int capacity)	Itisusedtobuildanarraylistthathasthespecifiedinitialcapac ity.

Java ArrayList

Example

```
import java.util.*;
class TestCollection1{
 public static void main(String args[]){
  ArrayList<String> list=new ArrayList<String>();//Creating
  arraylistlist.add("Ravi");//Addingobject in arraylist
  list.add("Vijay");
  list.add("Ravi");
  list.add("Ajay");
  //Traversing list through
  IteratorIterator
  itr=list.iterator();while(itr.hasNext()){
  System.out.println(itr.next());}}}
```

```
Ravi
Vijay
Ravi
Ajay
```

vector

ArrayListandVectorbothimplementsListinterfaceandmaintainsinsertionorder.

ButthereareManydifferencesbetweenArrayListandVector classesthataregivenbelow.

ArrayList	Vector
1)ArrayListisnotsynchronized.	Vectoris**synchronized**.
2)ArrayList **increments50%** ofcurrentarraysizeifnumberofelementexceedsfromitscapacity.	Vector **increments 100%** means doubles the arraysizeiftotalnumberofelementexceedsthanitscapacity.
3)ArrayListis **notalegacy** class,itis introduced in JDK1.2.	Vectorisa**legacy** class.
4)ArrayListis **fast**becauseitisnon-synchronized.	Vectoris **slow** becauseitissynchronizedi.e.inmultithreading environment, it will hold the otherthreadsinrunnableornon-runnablestateuntilcurrentthread releases thelockofobject.
5)ArrayListusesIteratorinterfacetotraversetheelements.	VectorusesEnumerationinterfacetotraversetheelements.But it canuseIterator also.

ExampleofJavaVector

Let'sseeasimpleexampleofjavaVector classthatusesEnumerationinterface.

```
1. importjava.util.*;
2. classTestVector1{
3.  publicstaticvoidmain(Stringargs[]){
4.   Vector<String>v=newVector<String>();//creatingvector
5.   v.add("umesh");//methodofCollection
6.   v.addElement("irfan");//methodofVector
7.   v.addElement("kumar");
```

8. //traversingelements usingEnumeration

```
9.    Enumeratione=v.elements();
10.   while(e.hasMoreElements()){
11.    System.out.println(e.nextElement());
12.}}}
```

Output:

```
umesh
irfank
umar
```

JavaHashtableclass

JavaHashtableclassimplementsahashtable,whichmapskeystovalues.ItinheritsDictionaryclassand implements theMapinterface.

Theimportantpoints aboutJavaHashtable classare:

- A Hashtable is an array of list. Each list is known as a bucket. The position of bucket isidentifiedbycallingthehashcode()method.AHashtablecontainsvaluesbasedonthekey.
- Itcontains onlyuniqueelements.
- It mayhavenothaveanynull keyor value.
- Itissynchronized.

Hashtableclassdeclaration

Let'sseethedeclaration forjava.util.Hashtable class.

1. **public class** Hashtable<K,V>**extends** Dictionary<K,V>**implements** Map<K,V>, Cloneable, Serializable

HashtableclassParameters

Let'sseetheParametersforjava.util.Hashtable class.

- **K**:Itisthe typeofkeysmaintained bythismap.
- **V**:Itisthetypeofmappedvalues.

ConstructorsofJavaHashtableclass

Constructor	Description
Hashtable()	Itisthedefaultconstructorofhashtableitinstantiatestheclass.
Hashtable(intsize)	Itisusedtoacceptanintegerparameterandcreatesahashtablethathas an initial sizespecifiedbyinteger value size.
Hashtable(int size, float fillRatio)	Itisusedtocreateahashtablethathasaninitialsizespecifiedbysizeand afill ratio specified byfillRatio.

JavaHashtableExample

```
importjava.util.*;
classTestCollection16{
 publicstaticvoidmain(Stringargs[]){Hashtable<Integer,String>
 hm=new Hashtable<Integer,String>();hm.put(100,"Amit");
 hm.put(102,"Ravi");
 hm.put(101,"Vijay");
 hm.put(103,"Rahul");
 for(Map.Entry
  m:hm.entrySet()){System.out.println(m.getKey()+
  ""+m.getValue());
 }}}
```

Output:

```
103Rahul
102Ravi
101 Vijay
100 Amit
```

Stack

Stackisa subclassofVectorthat implementsastandardlast-in,first-outstack.

Stackonlydefinesthedefaultconstructor,whichcreatesanemptystack.Stackincludesallthemethods definedbyVector, and adds several of its

own.

```
Stack()
```

Example

Thefollowingprogram illustrates several ofthemethods supported bythiscollection−

```
importjava.util.*;
publicclassStackDemo{
   staticvoidshowpush(Stack st,int a){
      st.push(newInteger(a));
      System.out.println("push("+ a+ ")");
      System.out.println("stack:"+st);}
   staticvoidshowpop(Stack st){
      System.out.print("pop->");
      Integera=(Integer)st.pop();
      System.out.println(a);
      System.out.println("stack:"+st);}
   publicstaticvoidmain(Stringargs[]){
      Stackst=new Stack();
      System.out.println("stack:"+st);
      showpush(st,42);
      showpush(st,66);
      showpush(st,99);
      showpop(st);
      showpop(st);
      showpop(st);
      try{
         showpop(st);
```

```
}catch(EmptyStackException e){
  System.out.println("emptystack");
}}}
```

Thiswillproducethefollowingresult −

Output

```
stack:[]push(4
2)stack:
[42]push(66)s
tack: [42,
66]push(99)
stack:[42, 66, 99]
pop->99
stack:[42, 66]
pop->66
stack:[42]
pop ->
42stack:[
]
pop ->emptystack
```

Enumeration

TheEnumerationInterface

The Enumeration interface defines the methods by which you can enumerate (obtain one at atime)the elements in acollection ofobjects.
Themethods declared byEnumeration aresummarizedin thefollowingtable −

Sr.No.	*Method&Description*
1	**booleanhasMoreElements()** Whenimplemented,itmustreturntruewhiletherearestillmoreelementstoextract,andfalsewhenall the elements havebeenenumerated.
2	***ObjectnextElement()*** *Thisreturnsthenextobjectintheenumeration asagenericObjectreference.*

Example:Followingis anexampleshowingusageofEnumeration.

iswillproducethefollowingresult −

```
import
java.util.Vector;importjava.ut
il.Enumeration;
publicclassEnumerationTester{
  public static void main(String args[])
    {Enumerationdays;
    VectordayNames=newVector();da
    yNames.add("Sunday");dayName
    s.add("Monday");dayNames.add("
    Tuesday");dayNames.add("Wedn
    esday");dayNames.add("Thursday
    ");dayNames.add("Friday");dayN
    ames.add("Saturday");
    days =
    dayNames.elements();while(days.
    hasMoreElements()){
      System.out.println(days.nextElement());
    } }}
```

Output

Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

It is a universal iterator as we can apply it to any Collection object. By using Iterator, we canperform both read and remove operations. It is improved version of Enumeration with additionalfunctionalityofremove-abilityofaelement.

Iterator must be used whenever we want to enumerate elements in all Collection frameworkimplemented interfaces like Set, List, Queue, Deque and also in all implemented classes of Mapinterface.Iterator isthe **only** cursoravailableforentirecollection framework.
Iteratorobject canbecreatedbycalling*iterator()*methodpresent inCollection interface.

```
//Here "c"is anyCollection object. itr is of
//type
Iteratorinterfaceandrefersto"c"Iteratoritr
```

Iteratorinterfacedefines**three**methods:

```
//Returns trueif theiteration hasmoreelements
publicbooleanhasNext();

//Returns thenext elementin theiteration
// ItthrowsNoSuchElementExceptionifnomore
//elementpresent
publicObjectnext();

//Remove thenext element intheiteration
//This method can becalled onlyonceper call
// to next()

publicvoidremove();
```

***remove()*methodcanthrow twoexceptions**

- *UnsupportedOperationException:*If theremove operationisnotsupportedbythisiterator
- *IllegalStateException:*Ifthenextmethodhasnotyetbeencalled,ortheremovemethodhasalreadyb eencalled after thelast call to thenext method

LimitationsofIterator:

- Onlyforward direction iteratingispossible.
- Replacementandadditionofnewelementisnotsupported byIterator.

StringTokenizerinJava

The**java.util.StringTokenizer**

classallowsyoutobreakastringintotokens.Itissimplewaytobreakstring.

Itdoesn'tprovidethe facilitytodifferentiate numbers,quotedstrings,identifiers etc.

ConstructorsofStringTokenizerclass

Thereare3constructorsdefinedintheStringTokenizerclass.

Constructor	Description
StringTokenizer(Stringstr)	createsStringTokenizerwithspecifiedstring.
StringTokenizer(String str, Stringdelim)	creates StringTokenizer with specified string and delimeter.
StringTokenizer(String str, String delim, boolean returnValue)	createsStringTokenizerwithspecifiedstring,delimeter and returnValue. If return value is true, delimiter charactersareconsideredtobetokens.Ifitisfalse, delimitercharactersservetoseparatetokens.

MethodsofStringTokenizerclass

The6usefulmethodsof StringTokenizerclassareasfollows:

Publicmethod	Description
booleanhasMoreTokens()	checksifthereis moretokensavailable.
StringnextToken()	returnsthenexttokenfromtheStringTokenizerobject.
StringnextToken(String delim)	returnsthenext tokenbasedon thedelimeter.
booleanhasMoreElements()	sameashasMoreTokens()method.
ObjectnextElement()	sameasnextToken()but itsreturntypeisObject.
intcountTokens()	returnsthe total numberoftokens.

SimpleexampleofStringTokenizerclass

Let'sseethesimpleexampleofStringTokenizerclassthattokenizesastring"mynameiskhan"onthe basis of whitespace.

```
importjava.util.StringTokenizer;
publicclassSimple{
 publicstaticvoid main(Stringargs[]){
```

```
StringTokenizerst=newStringTokenizer("mynameiskhan","");
  while (st.hasMoreTokens())
     {System.out.println(st.nextToken()
     );
  }}}
```

Output:my
name
iskh
an

ExampleofnextToken(Stringdelim)methodofStringTokenizerclass

```
importjava.util.*;
publicclassTest{
  publicstaticvoidmain(String[] args){
    StringTokenizerst=newStringTokenizer("my,name,is,khan");
    // printingnext token
    System.out.println("Nexttokenis:"+st.nextToken(","));
  }    }
```

Output:Nexttokenis :my

java.util.Random

- For using this class to generate random numbers, we have to first create an instance of thisclass and then invoke methods such as nextInt(), nextDouble(), nextLong() etc using thatinstance.
- We can generate random numbers of types integers, float, double, long, booleans using thisclass.
- We can pass arguments to the methods for placing an upper bound on the range of thenumbers to be generated. For example, nextInt(6) will generate numbers in the range 0 to 5bothinclusive.

//AJavaprogramtodemonstrate randomnumbergeneration

```
// using
java.util.Random;importj
ava.util.Random;

publicclass generateRandom{

   publicstatic void main(Stringargs[])
   {
      // create instance of Random
      classRandomrand
      =newRandom();

      // Generate random integers in range 0 to
      999intrand_int1 =rand.nextInt(1000);
```

```
        intrand_int2=rand.nextInt(1000);

        // Print random integersSystem.out.println("Random
        Integers:
        "+rand_int1);System.out.println("RandomIntegers:"
        +rand_int2);

        //GenerateRandomdoubles
        double rand_dub1 =
        rand.nextDouble();doublerand_dub2=ra
        nd.nextDouble();

        //Printrandom doubles
        System.out.println("Random Doubles:
        "+rand_dub1);System.out.println("RandomDoubles:"
        +rand_dub2);
    }}
```

Output:

```
RandomIntegers:547
RandomIntegers:126
RandomDoubles: 0.8369779739988428
RandomDoubles: 0.5497554388209912
```

JavaScannerclass

There are various ways to read input from the keyboard, the java.util.Scanner class is one of them.The **JavaScanner** classbreakstheinputintotokensusingadelimiterthatiswhitespacebydefault.It provides manymethods toread and parsevarious primitive values.

Java Scanner class is widely used to parse text for string and primitive types using regularexpression.
JavaScannerclassextendsObjectclassandimplements Iterator andCloseableinterfaces.

CommonlyusedmethodsofScannerclass

ThereisalistofcommonlyusedScannerclass methods:

Method	Description
publicStringnext()	itreturnsthe nexttoken from thescanner.
publicStringnextLine()	itmovesthescannerpositiontothenextlineandr

publicbytenextByte()	itscansthenexttokenasa byte.

publicshort nextShort()	itscansthenexttoken asashort value.
publicintnextInt()	itscans thenext token asan int value.
publiclongnextLong()	itscans thenext token as alongvalue.
publicfloat nextFloat()	itscansthenexttoken asafloat value.
public double nextDouble()	itscans thenext tokenas adouble value.

JavaScannerExampleto getinputfromconsole

Let'sseethesimpleexampleoftheJavaScannerclasswhichreadstheint,stringanddoublevalueas an input:

```
importjava.util.Scanner;
classScannerTest{
public static void main(String
 args[]){Scanner sc=new
 Scanner(System.in);System.out.println(
 "Enter your rollno");int
 rollno=sc.nextInt();System.out.println("
 Enter your name");String
 name=sc.next();System.out.println("Ent
 er your
 fee");doublefee=sc.nextDouble();
 System.out.println("Rollno:"+rollno+" name:"+name+"
 fee:"+fee);sc.close();
}}
```

Output:

```
Enteryourrollno1
11
Enteryourname
Ratan
Enter4
50000
Rollno:111name:Ratanfee:450000
```

JavaCalendarClass

Java Calendar class is an abstract class that provides methods for converting date between aspecific instant in time and a set of calendar fields such as MONTH, YEAR, HOUR, etc. ItinheritsObject class andimplements theComparable interface.

JavaCalendarclassdeclaration

Let'sseethedeclarationofjava.util.Calendarclass.

1. **publicabstractclass** Calendar**extends**Object
2. **implements**Serializable,Cloneable,Comparable<Calendar>

JavaCalendarClassExample

```
importjava.util.Calendar;
publicclassCalendarExample1 {
  public static void main(String[] args)
  {Calendarcalendar=Calendar.getInstance();
  System.out.println("The current date is : "+
  calendar.getTime());calendar.add(Calendar.DATE,-15);
  System.out.println("15 days ago: "+
  calendar.getTime());calendar.add(Calendar.MONTH,4);
  System.out.println("4 months later: "+
  calendar.getTime());calendar.add(Calendar.YEAR, 2);
  System.out.println("2yearslater:"+calendar.getTime());
  }}
```

Output:

```
The current date is : Thu Jan 19 18:47:02 IST
201715daysago: WedJan 0418:47:02IST 2017
4monthslater:ThuMay04 18:47:02IST2017
2yearslater:Sat May0418:47:02IST2019
```

Java-FilesandI/O

The java.io package contains nearly every class you might ever need to perform input and output(I/O)inJava.Allthesestreamsrepresentaninputsourceandanoutputdestination.Thestreaminthe java.io packagesupportsmanydatasuch asprimitives, object,localizedcharacters, etc.

Stream

Astreamcanbedefinedasa sequenceofdata. Therearetwo kindsofStreams−

- **InPutStream**−TheInputStreamisusedtoread datafromasource.
- **OutPutStream**−TheOutputStream isusedforwritingdatatoadestination.

Java provides strong but flexible support for I/O related to files and networks but this tutorialcovers very basic functionality related to streams and I/O. We will see the most commonly usedexamplesonebyone−

ByteStreams

Java byte streams are used to perform input and output of 8-bit bytes. Though there are manyclasses related to byte streams but the most frequently used classesare, **FileInputStream** and **FileOutputStream**. Following is an example which makes useofthesetwo classesto copyan input fileinto an output file−

Example

```
importjava.io.*;
publicclassCopyFile {
  public static void main(String args[]) throws IOException
    {FileInputStream in =null;
    FileOutputStream out =
    null;try{
      in=newFileInputStream("input.txt");
      out=newFileOutputStream("output.txt");i
      ntc;
      while((c =in.read())!=-1){
```

```
      out.write(c);
    }
  }finally{
    if(in !=null){
      in.close();
    }
    if (out != null)
      {out.close();
    }}}}
```

Nowlet'shave afile **input.txt**withthefollowingcontent −

```
Thisistest forcopyfile.
```

As a next step, compile the above program and execute it, which will result in creating output.txtfilewiththesamecontentaswehaveininput.txt.Solet'sputtheabovecodeinCopyFile.javafile and do the following−

```
$javacCopyFile.java
$javaCopyFile
```

CharacterStreams

Java **Byte** streams are used to perform input and output of 8-bit bytes, whereasJava **Character** streams are used to perform input and output for 16-bit unicode. Though therearemanyclassesrelatedto character streams but the most frequently used classesare, **FileReader** and **FileWriter**.ThoughinternallyFileReaderusesFileInputStreamandFileWriter uses FileOutputStream but here the major difference is thatFileReader reads twobytesatatime and FileWriterwritestwo bytes at a time.

Wecanre-writetheaboveexample,whichmakestheuseofthesetwoclassestocopyaninput file(havingunicodecharacters)intoanoutputfile−

Example

```
importjava.io.*;
publicclassCopyFile {
  publicstaticvoid main(Stringargs[])throwsIOException{
```

```
FileReader     in     =
null;FileWriter out =
null;try{
  in = new
  FileReader("input.txt");out = new
  FileWriter("output.txt");intc;
  while ((c = in.read()) != -1)
    {out.write(c);}
}finally{
  if(in !=null){
    in.close();}
  if (out != null)
    {out.close();
  }}}}
```

Nowlet'shave afile **input.txt**withthefollowingcontent −

```
This is test forcopyfile.
```

As a next step, compile the above program and execute it, which will result in creating output.txtfile with the same content as we have in input.txt. So let's put the above code in CopyFile.javafileand do the following−

```
$javacCopyFile.java
$javaCopyFile
```

StandardStreams

All the programming languages provide support for standard I/O where the user's program cantake input from a keyboard and then produce an output on the computer screen. Java provides thefollowingthreestandardstreams −

- **StandardInput**−Thisisusedtofeedthedatatouser'sprogramandusuallyakeyboard isusedasstandardinputstream andrepresentedas**System.in**.

- **Standard Output** − This is used to output the data produced by the user's program andusuallyacomputerscreenisusedforstandardoutputstreamandrepresentedas**System.out**.
- **Standard Error** − This is used to output the error data produced by the user's programandusuallyacomputerscreenisusedforstandarderrorstreamandrepresentedas**System.err**.

Following is a simple program, which creates **InputStreamReader** to read standard input streamuntiltheusertypesa"

Example

```
importjava.io.*;

publicclassReadConsole {
  public static void main(String args[]) throws IOException
    {InputStreamReadercin =null;
    try{
      cin=newInputStreamReader(System.in);System.o
      ut.println("Entercharacters,'q'toquit.");charc;
      do{
        c = (char)
        cin.read();System.ou
        t.print(c);
      }while(c!='q');
    }finally{
 if(cin!=null){
        cin.close();
      }} }}
```

Thisprogram continuesto readand outputthe samecharacter untilwepress'q'−

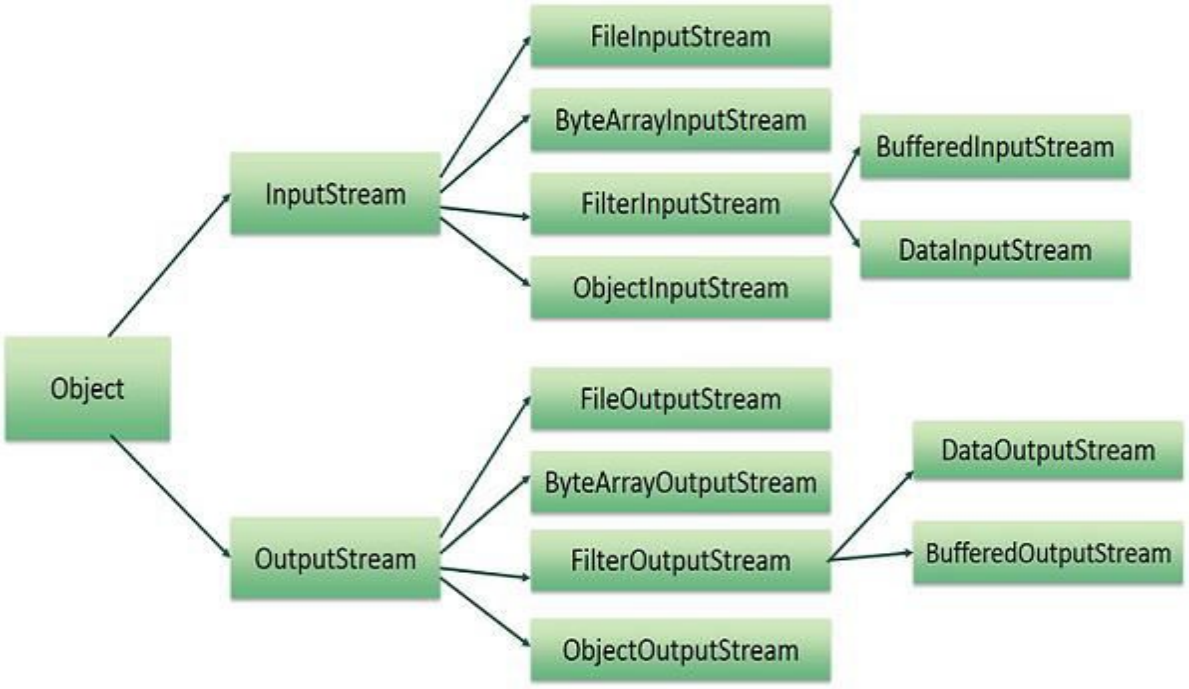
```
$javacReadConsole.java
$javaReadConsole
```

```
Enter characters, 'q' to
quit.1
1
e
e
q
q
```

ReadingandWritingFiles

Asdescribedearlier,astreamcanbedefinedasasequenceofdata.The**InputStream**isusedtoreaddatafro m asourceandthe**OutputStream**isusedforwritingdata toa destination.

Hereis ahierarchyof classes todeal withInput andOutput streams.



Thetwoimportantstreamsare**FileInputStream**and**FileOutputStream**

FileInputStream

Thisstreamisusedforreadingdatafromthefiles.Objectscanbecreatedusingthekeyword**new**and thereareseveral types ofconstructorsavailable.

Followingconstructortakesafilenameasastringtocreateaninputstreamobjecttoreadthe file −

```
InputStreamf=newFileInputStream("C:/java/hello");
```

Following constructortakesafileobjecttocreateaninputstreamobjecttoreadthefile.Firstwe createafileobject usingFile()methodasfollows−

```
File f = new
File("C:/java/hello");InputStreamf=new
```

Onceyouhave *InputStream*objectinhand,thenthereisalistofhelpermethodswhichcanbeusedto read to stream ortodo other operations onthe stream.

- ByteArrayInputStream
- DataInputStream

FileOutputStream

FileOutputStreamisusedtocreateafileandwritedataintoit.Thestreamwouldcreateafile,ifit doesn't alreadyexist, beforeopeningit foroutput.

HerearetwoconstructorswhichcanbeusedtocreateaFileOutputStreamobject.

Followingconstructortakesafilenameasastringtocreateaninputstreamobjecttowritethe file −

```
OutputStreamf=newFileOutputStream("C:/java/hello")
```

Followingconstructortakesafileobjecttocreateanoutputstreamobjecttowritethefile.First, wecreate afileobjectusingFile()methodasfollows−

```
File f = new
File("C:/java/hello");OutputStreamf=newFi
```

Onceyouhave*OutputStream*objectinhand,thenthereisalistofhelpermethods,whichcanbeusedto writeto stream orto doother operations on the stream.

- ByteArrayOutputStream
- DataOutputStream

Example

Followingistheexample todemonstrate InputStreamandOutputStream−

```
importjava.io.*;
publicclassfileStreamTest{
  public static void main(String args[])
    {try{
```

Java.io.RandomAccessFileClass

The**Java.io.RandomAccessFile**classfilebehaveslikealargearrayofbytesstoredinthefilesystem.Instancesof thisclass supportboth reading andwritingtoarandomaccessfile.

Classdeclaration

Followingisthedeclarationfor**Java.io.RandomAccessFile**class−

```
publicclassRandomAccessFilee
  xtendsObject
    implementsDataOutput,DataInput,Closeable
```

Classconstructors

S.N.	*Constructor& Description*
1	**RandomAccessFile(Filefile,Stringmode)** Thiscreatesarandomaccessfilestreamtoreadfrom,andoptionallytowriteto,thefilespecifiedbythe Fileargument.

2	***RandomAccessFile(Filefile,Stringmode)*** Thiscreatesarandomaccessfilestreamtoreadfrom,andoptionallytowriteto,afilewiththespecifiedname.

Methodsinherited

Thisclassinheritsmethodsfromthefollowingclasses−

- Java.io.Object

Java.io.FileClassinJava

The File class is Java's representation of a file or directory path name. Because file and directorynames have different formats on different platforms, a simple string is not adequate to name them.The File class contains several methods for working with the path name, deleting and renamingfiles,creatingnewdirectories,listingthecontentsofadirectory,anddeterminingseveralcommonattributes of files anddirectories.

- Itisan abstract representationoffileanddirectorypathnames.
- A pathname, whether abstract or in string form can be either absolute or relative. The parentofan abstract pathnamemaybeobtained byinvokingthegetParent()method ofthis class.
- First of all, we should create the File class object by passing the filename or directory nameto it. A file system may implement restrictions to certain operations on the actual file-system object, such as reading, writing, and executing. These restrictions are collectivelyknownas access permissions.
- Instances of the File class are immutable; that is, once created, the abstract pathnamerepresentedbyaFileobject will neverchange.

How tocreateaFileObject?

A File object is created by passing in a String that represents the name of a file, or a String oranotherFileobject. For example,

```
Filea=newFile("/usr/local/bin/geeks");
```

definesanabstractfilenameforthegeeksfileindirectory/usr/local/bin.Thisisanabsoluteabstractfilename.

Programtocheckif afileordirectoryphysicallyexistornot.

```
//Inthis program,weacceptsa fileor directoryname from
//commandlinearguments.Then theprogramwillcheckif
// that fileordirectoryphysicallyexistor not and
//itdisplaysthe propertyofthat fileordirectory.
*importjava.io.File;

// Displaying file
propertyclassfileProperty
{
   publicstaticvoidmain(String[] args){
```

```
        //accept file name or directory name through command line
        argsStringfname=args[0];

        //pass the filename or directory name to File
        objectFilef=newFile(fname);

        //apply File class methods on File
        objectSystem.out.println("File name
        :"+f.getName());System.out.println("Path:
        "+f.getPath());System.out.println("Absolute path:"
        +f.getAbsolutePath());System.out.println("Parent:"+f.getP
        arent());System.out.println("Exists:"+f.exists());
        if(f.exists())
        {
           System.out.println("Is
           writeable:"+f.canWrite());System.out.println("Is
           readable"+f.canRead());System.out.println("Is a
           directory:"+f.isDirectory());System.out.println("File
           Sizeinbytes"+f.length());
        }
     }
}
```

Output:

```
File name
:file.txtPath:file.
txt
Absolute
path:C:\Users\akki\IdeaProjects\codewriting\src\file.txtParent:nu
ll
Exists:true
Is
writeable:trueIsr
```

Connceting toDB

WhatisJDBCDriver?

JDBCdriversimplementthedefinedinterfacesintheJDBCAPI,forinteractingwithyourdatabaseserver.

Forexample,usingJDBCdriversenableyoutoopendatabaseconnectionsandtointeractwithitbysending SQLor databasecommands then receivingresults with Java.

The *Java.sql* package that ships with JDK, contains various classes with their behaviours definedand their actual implementaions are done in third-party drivers. Third party vendors implementsthe*java.sql.Driver* interfacein their databasedriver.

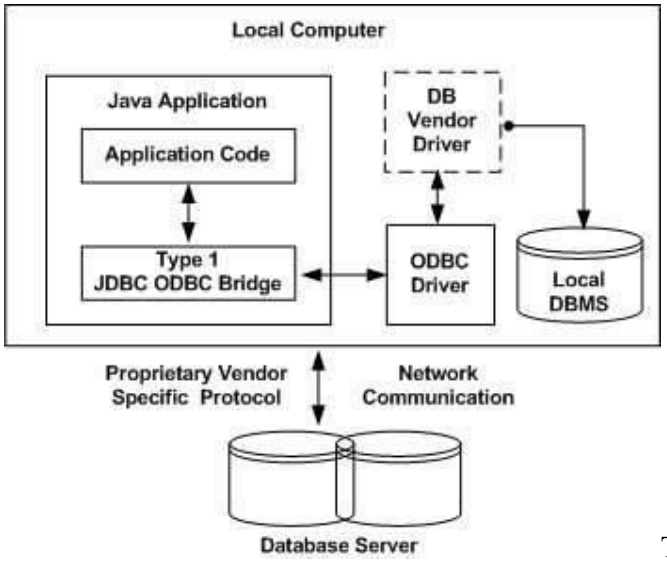
JDBCDriversTypes

JDBCdriverimplementationsvarybecauseofthewidevarietyofoperatingsystemsandhardware platforms in which Java operates. Sun has divided the implementation types into fourcategories,Types1, 2, 3,and 4,which is explained below −

Type1:JDBC-ODBCBridgeDriver

In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each clientmachine. Using ODBC, requires configuring on your system a Data Source Name (DSN) thatrepresentsthetarget database.

When Java first came out, this was a useful driver because most databases only supported ODBCaccess but now this type of driver is recommended only for experimental use or when no otheralternativeis available.

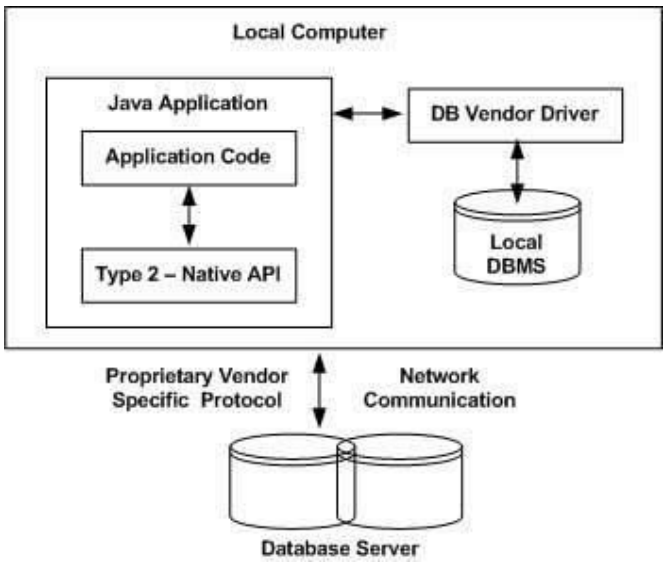


The JDBC-ODBC Bridge that comes withJDK1.2 is agood exampleof this kind ofdriver.

Type2:JDBC-NativeAPI

In a Type 2 driver, JDBC API calls are converted into native C/C++ API calls, which are uniqueto the database. These drivers are typically provided by the database vendors and used in thesame manner as the JDBC-ODBC Bridge. The vendor-specific driver must be installed on eachclientmachine.

If we change the Database, we have to change the native API, as it is specific to a database andthey are mostly obsolete now, but you may realize some speed increase with a Type 2 driver,becauseit eliminates ODBC's overhead.

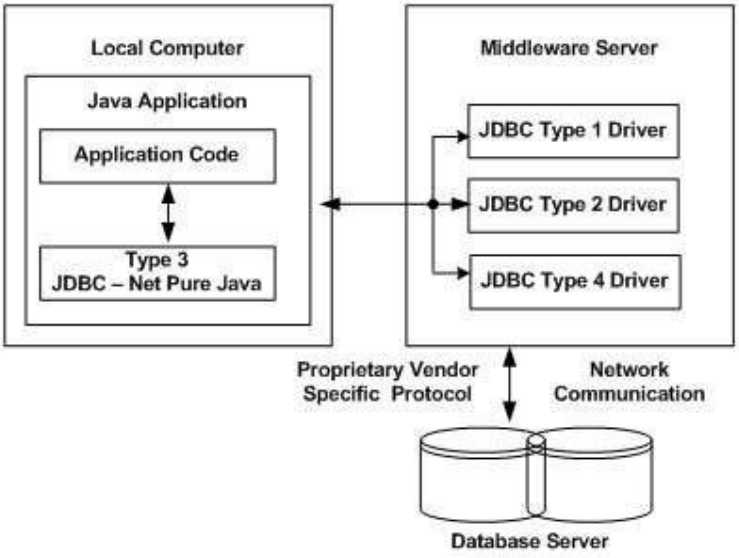


TheOracleCallInterface(OCI)driverisanexampleofaType2driver.

Type3:JDBC-NetpureJava

In a Type 3 driver, a three-tier approach is used to access databases. The JDBC clients usestandard network sockets to communicate with a middleware application server. The socketinformation is then translated by the middleware application server into the call format requiredbythe DBMS, and forwarded to thedatabaseserver.

This kind of driver is extremely flexible, since it requires no code installed on the client and asingledriver canactuallyprovide accessto multiple databases.



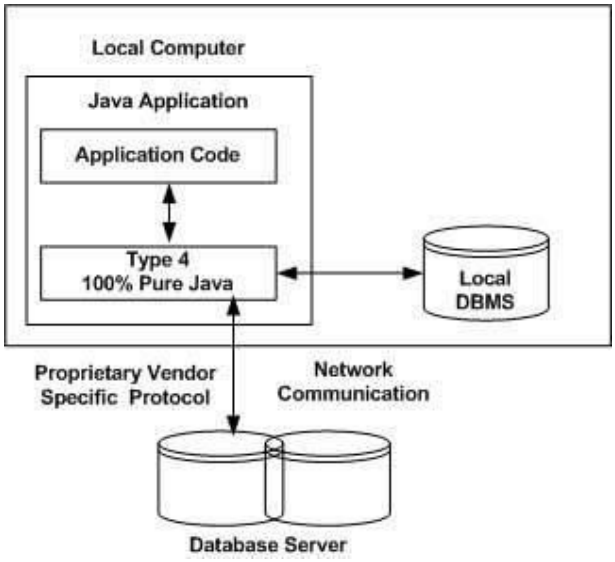
You can think of the application server as a JDBC "proxy," meaning that it makes calls for theclient application. As a result, you need some knowledge of the application server's configurationinorder to effectivelyusethis driver type.

Your application server might use a Type 1, 2, or 4 driver to communicate with the database,understandingthe nuances will provehelpful.

Type4:100%PureJava

In a Type 4 driver, a pure Java-based driver communicates directly with the vendor's databasethrough socket connection. This is the highest performance driver available for the database andis usuallyprovided bythevendoritself.

Thiskindofdriverisextremelyflexible,youdon'tneedtoinstallspecialsoftwareontheclientorserver. Further, thesedrivers can bedownloadeddynamically.



MySQL'sConnector/JdriverisaType4driver.Becauseoftheproprietarynatureoftheirnetworkprotocols, databasevendors usuallysupplytype4 drivers.

WhichDrivershouldbeUsed?

Ifyouareaccessingonetypeofdatabase,suchasOracle,Sybase,orIBM,thepreferreddrivertypeis 4.

IfyourJavaapplicationisaccessingmultipletypesofdatabasesatthesametime,type3isthepreferreddriver.

Type2driversareusefulinsituations,whereatype3ortype4driverisnotavailableyetforyourdatabase.

Thetype1driverisnotconsideredadeployment-leveldriver,andistypicallyusedfordevelopmentand testingpurposes only.

Exampletoconnecttothemysqldatabaseinjava

Forconnectingjavaapplicationwiththemysqldatabase,youneedtofollow5stepstoperformdatabase connectivity.

InthisexamplewareusingMySqlasthedatabase.Soweneedtoknowfollowinginformationsforthe mysql database:

1. **Driverclass:**Thedriver classforthemysqldatabaseis **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql databaseis **jdbc:mysql://localhost:3306/sonoo** wherejdbcistheAPI,mysqlisthedatabase,localhost is the server name on which mysql is running, we may also use IP address, 3306isthe port number and sonooisthe database name. We may use any database, insuchcase,you need toreplacethe sonoo withyour databasename.
3. **Username:**Thedefaultusernameforthemysqldatabaseis**root**.
4. **Password:** Password is given by the user at the time of installing the mysql database. Inthisexample, weare goingto userootas the password.

Let'sfirstcreateatableinthemysqldatabase,butbeforecreatingtable,weneedtocreatedatabase first.

```
1. createdatabasesonoo;
2. usesonoo;
3. create table emp(id int(10),name varchar(40),age
```

int(3));ExampletoConnectJavaApplicationwithmysqldatab

ase

Inthisexample,sonoois thedatabasename,rootis theusername andpassword.

```
importjava.sql.*;
classMysqlCon{
public static void main(String
args[]){try{Class.forName("com.mysql.jd
bc.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://l
```

```
ocalhost:3306/sonoo","root","root");
//heresonoo isdatabasename,rootisusernameand password
Statementstmt=con.createStatement();
ResultSetrs=stmt.executeQuery("select*fromemp");
while(rs.next())
System.out.println(rs.getInt(1)+""+rs.getString(2)+""+rs.getString(3));con.cl
ose();
}catch(Exceptione){System.out.println(e);}
}}
```

Theabove examplewillfetchall therecords of emptable.

Toconnectjavaapplicationwiththemysqldatabasemysqlconnector.jarfileisrequiredtobeloaded.

Twowaysto loadthe jar file:

1. pastethemysqlconnector.jarfileinjre/lib/extfolder
2. setclasspath

1) *pastethe mysqlconnector.jarfilein JRE/lib/extfolder:*

Downloadthemysqlconnector.jarfile.Go tojre/lib/ext folderand pastethejarfilehere.

2) *setclasspath:*

There are two ways to set the

classpath:1.temporary2.permane

nt

Howtosetthetemporaryclasspath

opencommand promptand write:

1.C:>setclasspath=c:\folder\mysql-connector-java-5.0.8-bin.jar;.;

How tosetthepermanentclasspath:

Go toenvironmentvariable then clickonnewtab.Invariable namewrite **classpath** andinvariable value paste the path to the mysqlconnector.jar file by appending mysqlconnector.jar;.; asC:\folder\mysql-connector-java-5.0.8-bin.jar;

JDBC-ResultSets

The SQL statements that read data from a database query, return the data in a result set. TheSELECT statement is the standard way to select rows from a database and view them in a

resultset.The*java.sql.ResultSet*interfacerepresents theresult setofadatabasequery.

AResultSetobjectmaintainsacursorthatpointstothecurrentrowintheresultset.Theterm"resultset"refers to therow and column data contained in a ResultSetobject.

Themethodsof theResultSetinterfacecanbebrokendowninto threecategories−

- **Navigationalmethods:**Usedtomovethe cursor around.
- **Getmethods:** Usedtoviewthedatainthecolumnsofthecurrentrowbeingpointedbythecursor.
- **Updatemethods:** Usedtoupdatethedatainthecolumnsofthecurrentrow.Theupdatescanthen beupdated in theunderlyingdatabaseaswell.

ThecursorismovablebasedonthepropertiesoftheResultSet.Thesepropertiesaredesignatedwhenthe correspondingStatementthat generates the ResultSetis created.

JDBCprovidesthefollowingconnectionmethodstocreatestatementswithdesiredResultSet−

- **createStatement(intRSType,intRSConcurrency);**
- **prepareStatement(StringSQL,intRSType,intRSConcurrency);**
- **prepareCall(Stringsql,intRSType,intRSConcurrency);**

The first argument indicates the type of a ResultSet object and the second argument is one of twoResultSet constantsforspecifyingwhether aresult set is read-onlyor updatable.

TypeofResultSet

ThepossibleRSTypearegivenbelow.IfyoudonotspecifyanyResultSettype,youwillautomaticallyget one that is TYPE_FORWARD_ONLY.

Type	Description
ResultSet.TYPE_FORWARD_ONLY	Thecursor canonlymoveforwardintheresultset.
ResultSet.TYPE_SCROLL_INSENSITIVE	The cursor can scroll forward and backward, andthe result set is not sensitive to changes made byothers to the database that occur after the result setwascreated.

ResultSet.TYPE_SCROLL_SENSITIVE.	The cursor can scroll forward and backward, andthe result set is sensitive to changes made byotherstothedatabasethatoccurafter theresultsetwascreated.

ConcurrencyofResultSet

ThepossibleRSConcurrencyaregivenbelow.Ifyoudonotspecifyany Concurrencytype,youwillautomaticallyget onethat is CONCUR_READ_ONLY.

Concurrency	Description
ResultSet.CONCUR_READ_ONLY	Createsaread-onlyresultset. Thisisthedefault
ResultSet.CONCUR_UPDATABLE	Createsanupdateable resultset.

ViewingaResultSet

The ResultSet interface contains dozens of methods for getting the data of the current row.Thereisa get methodfor each ofthepossibledatatypes,andeachgetmethodhas twoversions

- Onethattakes inacolumnname.
- Onethat takes inacolumn index.

Forexample,ifthecolumnyouareinterestedinviewingcontainsanint,youneedtouseoneof thegetInt()methodsofResultSet−

S.N.	Methods&Description
1	**publicintgetInt(StringcolumnName)throwsSQLException** Returnstheint inthe currentrowin thecolumn namedcolumnName.
2	**publicintgetInt(intcolumnIndex)throwsSQLException** Returnstheintinthecurrentrowinthespecifiedcolumnindex.Thecolumnindexstartsat1, meaningthe first column of arow is 1, thesecond column of arow is 2, and so on.

Similarly, there are get methods in the ResultSet interface for each of the eight Java primitivetypes,as wellascommontypessuchasjava.lang.String,java.lang.Object,andjava.net.URL.

TherearealsomethodsforgettingSQLdatatypesjava.sql.Date,java.sql.Time,java.sql.TimeStamp,java.sql.Clob,andjava.sql.Blob.Checkthedocumentationformoreinformationabout usingtheseSQLdata types.

Forabetter understanding, let us studyViewing-ExampleCode.

UpdatingaResultSet

TheResultSetinterfacecontainsacollectionofupdatemethodsforupdatingthedataofaresultset.

Aswith theget methods, therearetwoupdatemethods foreach datatype−

- Onethattakes inacolumnname.
- Onethat takes inacolumn index.

Forexample,toupdateaStringcolumnofthecurrentrowofaresultset,youwoulduseoneof thefollowingupdateString()methods−

S.N.	Methods&Description
1	**publicvoidupdateString(intcolumnIndex,Strings)throwsSQLException** ChangestheStringinthespecifiedcolumn tothevalueofs.
2	**public void updateString(String columnName, String s) throws SQLException**Similartothepreviousmethod,exceptthatthecolumnisspecifiedbyitsnameinstead ofits index.

There are update methods for the eight primitive data types, as well as String, Object, URL, andtheSQLdatatypes in thejava.sql package.

Updating a row in the result set changes the columns of the current row in the ResultSet object,but not in the underlying database. To update your changes to the row in the database, you needto invokeoneof thefollowingmethods.

S.N.	Methods&Description
1	**publicvoidupdateRow()** Updatesthecurrent row byupdatingthecorrespondingrow in the database.
2	**publicvoiddeleteRow()** Deletesthecurrentrowfromthedatabase
3	**publicvoidrefreshRow()** Refreshesthe data in theresultset toreflect anyrecent changesin thedatabase.
4	**publicvoidcancelRowUpdates()** Cancelsanyupdates madeon thecurrent row.
5	**publicvoidinsertRow()** Insertsarowintothedatabase.Thismethodcanonlybeinvokedwhenthecursorispointingto theinsert row.

UNIT-5

GUIProgrammingwithjava

TheAWTClass hierarchy

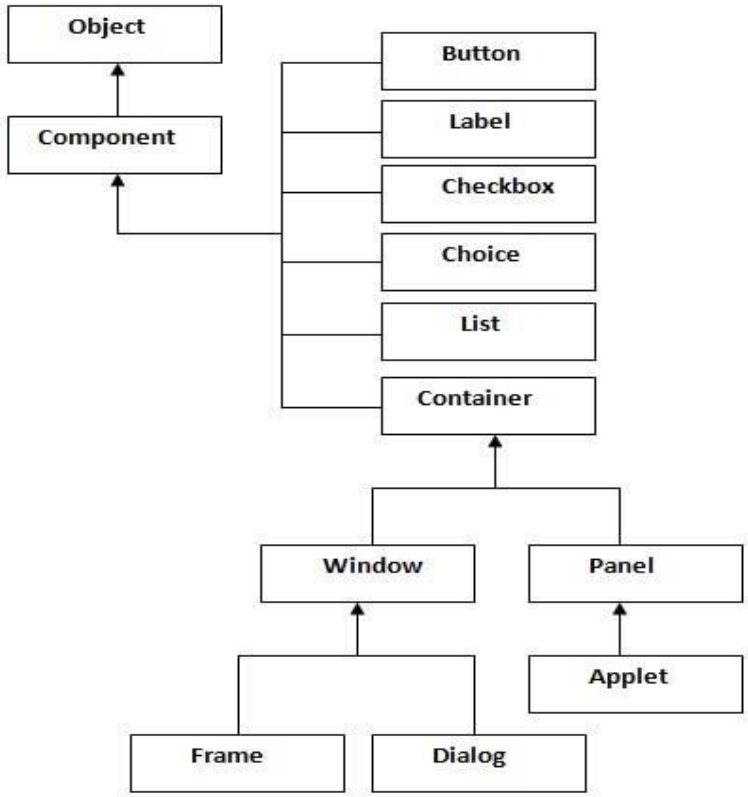
Java AWT(Abstract Window Toolkit) is*an API to develop GUI or window-basedapplications*in java.

JavaAWTcomponentsareplatform-dependenti.e.componentsaredisplayedaccordingtotheviewofoperatingsystem. AWTisheavyweight i.e.itscomponents areusingthe resourcesofOS.

Thejava.awtpackageprovidesclassesforAWTapisuchasTextField,Label,TextArea,RadioButton,CheckBox, Choice,List etc.

JavaAWTHierarchy

Thehierarchyof JavaAWT classes aregivenbelow.



Container

TheContainerisacomponentinAWTthatcancontainanothercomponentslikebuttons,textfields, labels etc. The classes that extends Container class are known as container such asFrame, Dialogand Panel.

Window

Thewindowisthecontainerthathavenobordersandmenubars.Youmustuseframe,dialogoranotherwindowfor creatingawindow.

Panel

ThePanelisthecontainerthatdoesn'tcontaintitlebarandmenubars.Itcanhaveothercomponents likebutton, textfield etc.

Frame

TheFrameisthecontainerthatcontaintitlebarandcanhavemenubars.Itcanhaveothercomponents likebutton, textfield etc.

UsefulMethodsofComponentclass

Method	Description
publicvoidadd(Component c)	insertsa component on this component.
publicvoidsetSize(intwidth,intheight)	setsthesize(width andheight) ofthecomponent.
publicvoidsetLayout(LayoutManagerm)	definesthelayoutmanagerforthe component.
publicvoidsetVisible(booleanstatus)	changesthevisibilityofthecomponent,bydefaultfalse.

JavaAWTExample

Tocreatesimpleawtexample,youneedaframe.Therearetwowaystocreateaframe inAWT.

- ByextendingFrameclass (inheritance)
- Bycreatingtheobject ofFrame class (association)

AWTExamplebyInheritance

Let'sseeasimpleexampleofAWTwhereweareinheritingFrameclass.Here,weareshowingButtoncomponent on theFrame.

```
importjava.awt.*;
class First extends
Frame{First(){
Buttonb=newButton("clickme");b.setBounds(30,100,8
0,30);// setting button positionadd(b);//adding button
into framesetSize(300,300);//frame size 300 width and
300 heightsetLayout(null);//nolayoutmanager
setVisible(true);//nowframewill bevisible,bydefault notvisible
}
public static void main(String
args[]){Firstf=newFirst();
}}
```

The setBounds(int xaxis, int yaxis, int width, int height) method is used in the above example thatsetstheposition ofthe awt button.

JavaSwing

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirelywrittenin java.

UnlikeAWT,JavaSwingprovidesplatform-independent andlightweightcomponents.

Thejavax.swingpackageprovidesclassesforjavaswingAPIsuchasJButton,JTextField,JTextArea,JRadioButton, JCheckbox,JMenu, JColorChooseretc.

DifferencebetweenAWT and Swing.

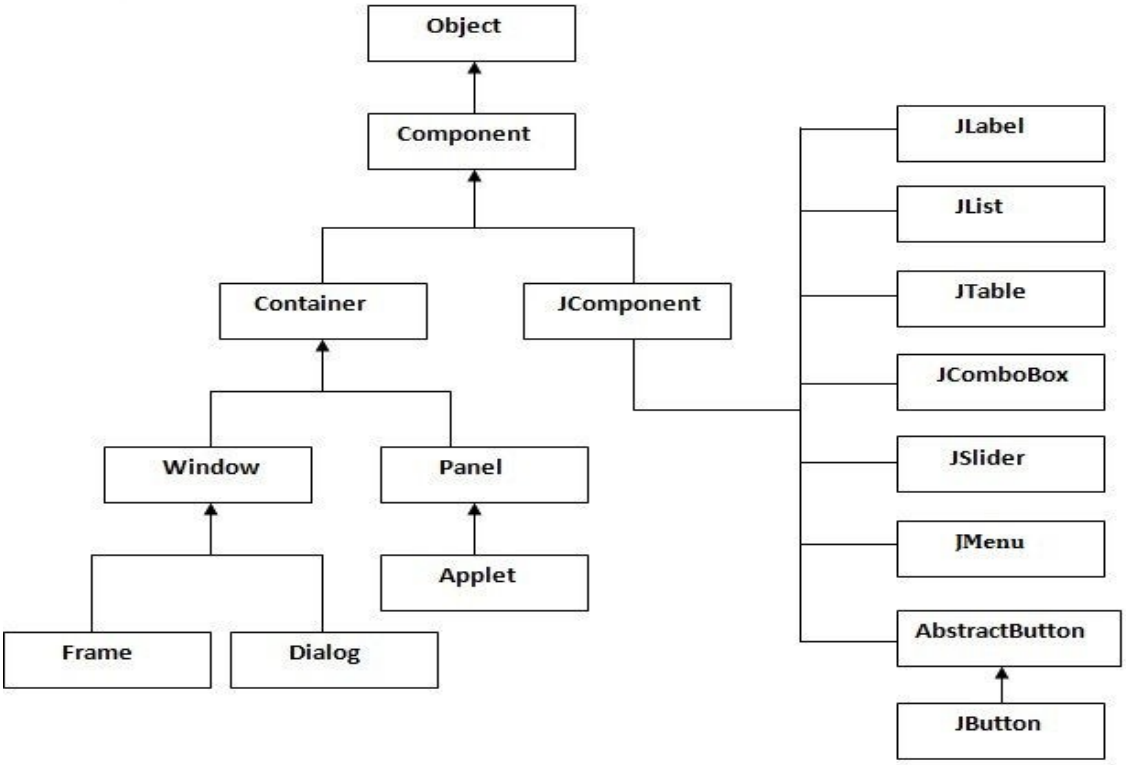
No.	Java AWT	JavaSwing
1)	AWT components are**platform-dependent**.	Javaswingcomponentsare**platform-independent**.
2)	AWTcomponents are**heavyweight**.	Swingcomponentsare **lightweight**.
3)	AWT**doesn'tsupportpluggablelookandfeel**.	Swing**supportspluggablelookandfeel**.
4)	AWT provides**less components**than Swing.	Swingprovides **morepowerfulcomponents**suchastables, lists,scrollpanes,colorchooser,tabbedpaneetc.
5)	AWT **doesn't follows MVC**(Model ViewController) where model represents data,viewrepresentspresentationandcontrolleractsasaninterfacebetweenmodeland view.	Swing**follows MVC**.

CommonlyusedMethodsofComponentclass

Method	Description
publicvoidadd(Component c)	addacomponent onanothercomponent.
publicvoidsetSize(intwidth,intheight)	sets sizeof thecomponent.
publicvoidsetLayout(LayoutManagerm)	setsthelayoutmanager forthecomponent.
publicvoidsetVisible(booleanb)	setsthevisibilityofthecomponent.Itisbydefaultfalse.

HierarchyofJavaSwingclasses

Thehierarchyof javaswingAPIisgiven below.



JavaSwingExamples

Therearetwowaysto createaframe:

- Bycreatingtheobject ofFrame class (association)
- ByextendingFrameclass (inheritance)

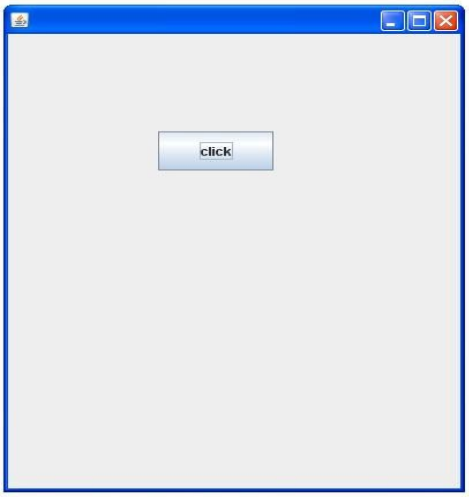
Wecan writethecodeof swinginsidethe main(),constructororanyother method.

SimpleJavaSwingExample

Let'sseeasimpleswingexamplewherewearecreatingonebuttonandaddingitontheJFrameobjectinside themain()method.

File:FirstSwingExample.java

```
importjavax.swing.*;
publicclassFirstSwingExample{
publicstaticvoidmain(String[] args){
JFramef=new JFrame();//creatinginstanceofJFrame
JButton b=new JButton("click");//creating instance of
JButtonb.setBounds(130,100,100, 40);//x axis, y axis, width,
heightf.add(b);//addingbutton in JFrame
f.setSize(400,500);//400 width and 500
heightf.setLayout(null);//using no layout
managersf.setVisible(true);//makingtheframe
visible
} }
```



Containers

JavaJFrame

The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class.JFrameworkslikethemainwindowwherecomponentslikelabels,buttons,textfieldsareaddedtocreateaGUI.

UnlikeFrame,JFramehastheoptiontohideorclosethewindowwiththehelpofsetDefaultCloseOperation(int)method.

JFrameExample

```
import
java.awt.FlowLayout;import
javax.swing.JButton;import
javax.swing.JFrame;import
javax.swing.JLabel;import
javax.swing.Jpanel;publicclass
JFrameExample{
publicstaticvoidmain(Strings[]){
JFrame frame = new JFrame("JFrame
Example");JPanel
panel=newJPanel();panel.setLayout(newFlowLay
out());
JLabel label = new JLabel("JFrame By
```

```
Example");JButton button = new
JButton();button.setText("Button");
panel.add(label);
```

```
panel.add(button);frame.add(panel)
;frame.setSize(200,
300);frame.setLocationRelativeTo(
null);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);frame.setVisibl
e(true);
}}
```

JApplet

AswepreferSwingtoAWT.NowwecanuseJAppletthatcanhaveallthecontrolsofswing.TheJApplet class extends the Applet class.

ExampleofEventHandlinginJApplet:

```
import
java.applet.*;import
javax.swing.*;importjav
a.awt.event.*;
public class EventJApplet extends JApplet implements
ActionListener{JButton b;
JTextFieldtf;
public void
init(){tf=newJTextF
ield();
tf.setBounds(30,40,150,20);
b=new
JButton("Click");b.setBoun
ds(80,150,70,40);add(b);ad
d(tf);b.addActionListener(t
his);setLayout(null);
}
public void actionPerformed(ActionEvent
e){tf.setText("Welcome");
}}Intheaboveexample,wehavecreatedallthecontrolsininit()methodbecauseitisinvokedonlyonce.
```

myapplet.html

```
1. <html>
2. <body>
3. <appletcode="EventJApplet.class"width="300"height="300">
```

</applet></body>
</html>

JDialog

TheJDialogcontrolrepresentsatoplevelwindowwithaborderandatitleusedtotakesomeformof input from the user.It inherits theDialogclass.

UnlikeJFrame,itdoesn't havemaximizeandminimizebuttons.

JDialogclassdeclaration

Let'sseethedeclarationforjavax.swing.JDialogclass.

1. **public class** JDialog **extends** Dialog **implements** WindowConstants, Accessible, RootPaneContainer

CommonlyusedConstructors:

Constructor	Description
JDialog()	Itisusedtocreateamodelessdialogwithoutatitleandwithout aspecifiedFrameowner.
JDialog(Frameowner)	ItisusedtocreateamodelessdialogwithspecifiedFrame as its ownerand an emptytitle.
JDialog(Frameowner,Stringtitle,bo oleanmodal)	Itisusedtocreateadialogwiththespecifiedtitle,owner Frameand modality.

JavaJDialogExample

```
importjavax.swing.*;
importjava.awt.*;
import
java.awt.event.*;public class
DialogExample {private
static JDialog
d;DialogExample(){
JFramef=new JFrame();
d=newJDialog(f,"DialogExample",true);d.setL
ayout(newFlowLayout() );
JButton b = new JButton
("OK");b.addActionListener(newActionLis
tener()
{
publicvoidactionPerformed(ActionEvente)
{
        DialogExample.d.setVisible(false);
}

});

d.add( new JLabel ("Click button to
continue."));d.add(b);
d.setSize(300,300);
d.setVisible(true);
}
publicstaticvoidmain(Stringargs[])
{
newDialogExample();
}}
```

Output:

JPanel

TheJPanelisasimplestcontainerclass.Itprovidesspaceinwhichanapplicationcanattachanyothercomponent.It inherits the JComponents class.

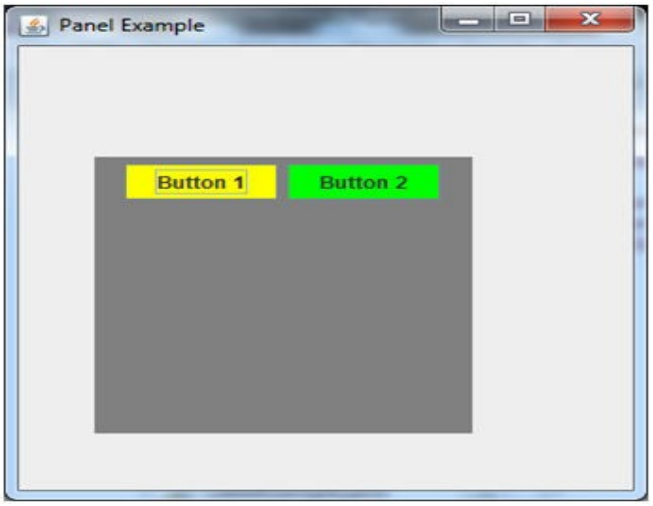
Itdoesn'thavetitlebar.

JPanelclassdeclaration

1.**public class** JPanel **extends** JComponent **implements** Accessible

JavaJPanelExample

```
import java.awt.*;
import javax.swing.*;
public class PanelExample
    {PanelExample()
      {
      JFrame f= new JFrame("Panel
      Example");JPanel panel=new
      JPanel();panel.setBounds(40,80,200,200);
      panel.setBackground(Color.gray);
      JButton b1=new JButton("Button
      1");b1.setBounds(50,100,80,30);b1.se
      tBackground(Color.yellow);JButton
      b2=new JButton("Button
      2");b2.setBounds(100,100,80,30);b2.s
      etBackground(Color.green);panel.add
      (b1); panel.add(b2);f.add(panel);
            f.setSize(400,400);
            f.setLayout(null);f
            .setVisible(true);
      }
      public static void main(String args[])
      {
      new PanelExample();
      }}
```



Overview of some Swing ComponentsJava

JButton

TheJButtonclassisusedtocreatealabeledbuttonthathasplatformindependentimplementation.Theapplicationresultins
ome actionwhenthebuttonis pushed.ItinheritsAbstractButtonclass.

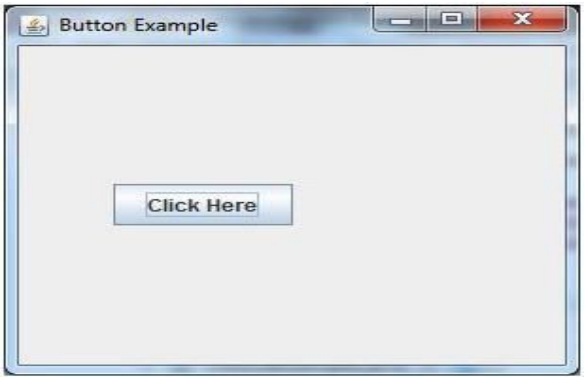
JButtonclassdeclaration

Let'sseethedeclarationforjavax.swing.JButtonclass.

1.**public class** JButton **extends** AbstractButton **implements** Accessible

JavaJButtonExample

```
import javax.swing.*;
public class ButtonExample {
public static void main(String[] args)
  {JFrame f=new JFrame("Button
  Example");JButton b=new JButton("Click
  Here");b.setBounds(50,100,95,30);
  f.add(b);f.setSize(400,4
  00);f.setLayout(null);f.
  setVisible(true);}}
```



JavaJLabel

The object of JLabel class is a component for placing text in a container. It is used to display asingle line of read only text. The text can be changed by an application but a user cannot edit itdirectly.It inherits JComponent class.

JLabelclassdeclaration

Let'sseethedeclarationforjavax.swing.JLabelclass.

1.**public class** JLabel **extends** JComponent **implements** SwingConstants,Accessible

CommonlyusedConstructors:

Constructor	Description
JLabel()	CreatesaJLabelinstancewithnoimageandwithanemptystringforthe title.
JLabel(Strings)	CreatesaJLabel instance with thespecifiedtext.
JLabel(Iconi)	CreatesaJLabelinstancewiththespecifiedimage.

JLabel(String Icon i, int s,ho rizontalAlignment)	CreatesaJLabelinstancewiththespecifiedtext,image,and horizontal alignment.

CommonlyusedMethods:

Methods	Description
StringgetText()	treturns thetext stringthatalabel displays.
void setText(Stringtext)	Itdefinesthe singlelineof textthiscomponentwilldisplay.
void setHorizontalAlignment(int alignment)	Itsetsthealignmentofthelabel'scontentsalongthe Xaxis.
IcongetIcon()	Itreturnsthe graphicimagethatthelabeldisplays.
intgetHorizontalAlignment()	It returns the alignment of the label's contents alongthe Xaxis.

JavaJLabelExample

```
importjavax.swing.*;
class LabelExample
{
publicstaticvoid main(Stringargs[])
   {
   JFrame f= new JFrame("Label
   Example");JLabell1,l2;
   l1=new JLabel("First
   Label.");l1.setBounds(50,50,
   100,30);l2=new JLabel("Second
   Label.");l2.setBounds(50,100,10
   0,30);f.add(l1);
   f.add(l2);f.setSize(300,300);f.set
   Layout(null);f.setVisible(true);
   }
   }
```

JTextField

TheobjectofaJTextFieldclassisatextcomponentthatallowstheeditingofasinglelinetext.ItinheritsJTextComponentclass.

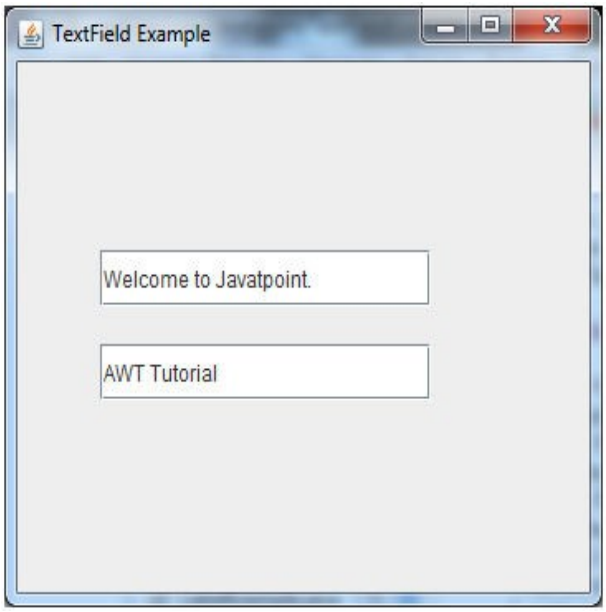
JTextFieldclassdeclaration

Let'sseethedeclarationforjavax.swing.JTextFieldclass.

1.**public**class JTextField**extends** JTextComponent**implements** SwingConstants

JavaJTextFieldExample

```
import javax.swing.*;
class TextFieldExample
{
public static void main(String args[])
  {
  JFrame f= new JFrame("TextField
  Example");JTextField t1,t2;
  t1=new JTextField("Welcome to
  Javatpoint.");t1.setBounds(50,100,200,30);
  t2=new JTextField("AWT
  Tutorial");t2.setBounds(50,150,200,3
  0);f.add(t1);
  f.add(t2);f.setSize(400,400);
  f.setLayout(null);
  f.setVisible(true);
  }  }
```



JavaJTextArea

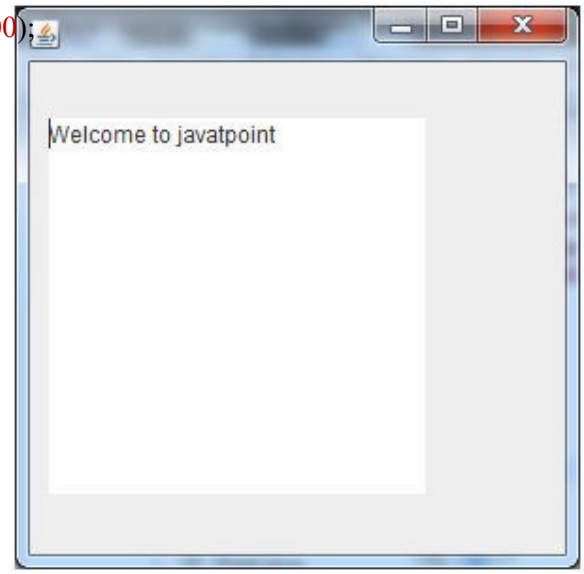
TheobjectofaJTextAreaclassisamultilineregionthatdisplaystext.Itallowstheeditingofmultipleline text.It inherits JTextComponent class

JTextAreaclassdeclaration

Let'sseethedeclarationfor javax.swing.JTextAreaclass.

1. **public**class JTextArea**extends** JTextComponent

```
importjavax.swing.*;
publicclassTextAreaExample
{
    TextAreaExample(){JFram
      ef=newJFrame();
      JTextArea area=new JTextArea("Welcome to
      javatpoint");area.setBounds(10,30,200,200);
      f.add(area);f.setSiz
      e(300,300);f.setLa
      yout(null);f.setVis
      ible(true);
    }
publicstaticvoidmain(Stringargs[])
   {
  newTextAreaExample();
   }}
```



SimpleJavaApplications

```
importjavax.swing.JFrame;importj
avax.swing.SwingUtilities;

public class Example extends JFrame

   {publicExample(){
     setTitle("Simple
     example");setSize(300,
     200);setLocationRelativeTo(
     null);
     setDefaultCloseOperation(EXIT_ON_CLOSE);
   }

   public static void main(String[] args)
     {Example ex = new
     Example();ex.setVisible(true);
   }}
```

LayoutManagement

JavaLayoutManagers

TheLayoutManagersareusedtoarrangecomponentsinaparticularmanner.LayoutManagerisaninterfac ethat is implemented byallthe classesoflayout managers.

BorderLayout

TheBorderLayoutprovides fiveconstantsfor eachregion:

1. **publicstaticfinalint NORTH**
2. **publicstaticfinalintSOUTH**
3. **publicstaticfinalintEAST**
4. **publicstaticfinalintWEST**
5. **publicstaticfinalintCENTER**

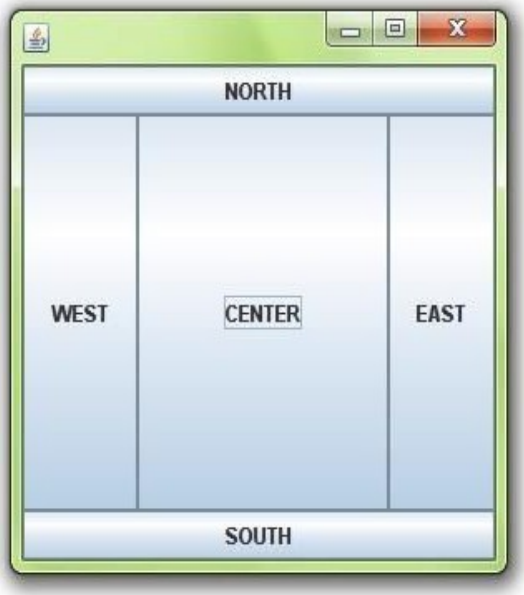
ConstructorsofBorderLayoutclass:

- **BorderLayout():**creates aborderlayoutbut withnogapsbetweenthecomponents.
- **JBorderLayout(inthgap,intvgap):**createsaborderlayoutwiththegivenhorizontalandverticalgap s between thecomponents.

ExampleofBorderLayoutclass:

```
importjava.awt.*;
importjavax.swing.*;
publicclassBorder
 {
JFrame
f;Border(
)
{
   f=newJFrame();
   JButton b1=new
   JButton("NORTH");;JButton b2=new
   JButton("SOUTH");;JButton b3=new
   JButton("EAST");;JButton b4=new
   JButton("WEST");;JButton b5=new
   JButton("CENTER");;f.add(b1,Border
   Layout.NORTH);f.add(b2,BorderLayo
   ut.SOUTH);f.add(b3,BorderLayout.EA
   ST);f.add(b4,BorderLayout.WEST);f.a
   dd(b5,BorderLayout.CENTER);f.setSiz
   e(300,300);
   f.setVisible(true);
}
publicstaticvoid main(String[] args)
```

Output:



```
{
  newBorder();
}}
```

JavaGridLayout

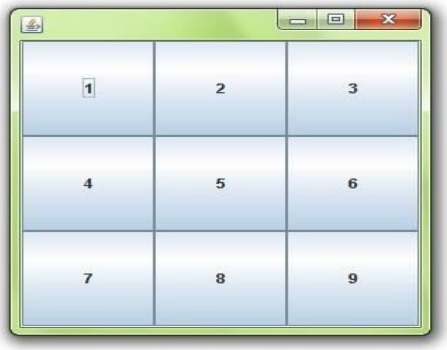
TheGridLayoutisusedtoarrangethecomponentsinrectangulargrid.Onecomponentisdisplayedin eachrectangle.

ConstructorsofGridLayoutclass

1. **GridLayout():** createsagridlayoutwith one columnpercomponentinarow.
2. **GridLayout(introws,intcolumns):**createsagridlayoutwiththegivenrowsandcolumnsbut no gaps between thecomponents.
3. **GridLayout(introws,intcolumns,inthgap,intvgap):**createsagridlayoutwiththegivenrows andcolumnsalongwithgiven horizontaland vertical gaps.

ExampleofGridLayoutclass

```
1. importjava.awt.*;
2. import
javax.swing.*;public class
MyGridLayout{JFramef;
MyGridLayout(){f
   =newJFrame();
   JButton                b1=new
   JButton("1");JButton   b2=new
   JButton("2");JButton   b3=new
   JButton("3");JButton   b4=new
   JButton("4");JButton   b5=new
   JButton("5");JButton   b6=new
   JButton("6");JButton   b7=new
   JButton("7");JButton   b8=new
   JButton("8");JButtonb9=newJ
   Button("9");
   f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
   f.add(b6);f.add(b7);f.add(b8);f.add(b9);
   f.setLayout(new GridLayout(3,3));
   //setting grid layout of 3 rows and 3
   columnsf.setSize(300,300);
   f.setVisible(true);
}
publicstaticvoidmain(String[] args){
 new MyGridLayout();}}
```



JavaFlowLayout

TheFlowLayoutisused toarrange the componentsina line,one afteranother(in a flow).Itisthedefaultlayou of applet or panel.

FieldsofFlowLayoutclass

1. **publicstaticfinalintLEFT**
2. **publicstaticfinalintRIGHT**
3. **publicstaticfinalintCENTER**
4. **publicstaticfinalintLEADING**

5. **publicstaticfinalintTRAILING**

ConstructorsofFlowLayoutclass

1. **FlowLayout():**createsaflowlayoutwithcenteredalignmentandadefault5unithorizontaland verticalgap.
2. **FlowLayout(intalign):**createsaflowlayoutwiththegivenalignmentandadefault5unithorizontal and vertical gap.
3. **FlowLayout(intalign,inthgap,intvgap):**createsaflowlayoutwiththegivenalignmentand thegivenhorizontal and verticalgap.

ExampleofFlowLayoutclass

```
importjava.awt.*;
importjavax.swing.*;public
class
MyFlowLayout{JFramef;
MyFlowLayout(){f
   =newJFrame();
   JButton                b1=new
   JButton("1");JButton  b2=new
   JButton("2");JButton  b3=new
   JButton("3");JButton  b4=new
   JButton("4");JButtonb5=newJ
   Button("5");
   f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);f.s
   etLayout(newFlowLayout(FlowLayout.RIGHT));
   //setting flow layout of right
  alignmentf.setSize(300,300);
   f.setVisible(true);
}
publicstaticvoidmain(String[] args){
newMyFlowLayout();
}}
```

EventHandling

EventandListener(JavaEventHandling)

Changing the state of an object is known as an event. For example, click on button, draggingmouse etc. The java.awt.event package provides many event classes and Listener interfaces foreventhandling.

TypesofEvent

Theeventscan bebroadlyclassified intotwocategories:

- **Foreground Events** - Those events which require the direct interaction of user.They aregenerated as consequences of a person interacting with the graphical components inGraphicalUserInterface.Forexample,clickingonabutton,movingthemouse,enteringacharacter throughkeyboard,selectinganitem fromlist, scrollingthepageetc.

- **BackgroundEvents**-Thoseeventsthatrequiretheinteractionofenduserareknownas

backgroundevents.Operatingsysteminterrupts,hardwareorsoftwarefailure,timerexpires,anoperation completionarethe example of backgroundevents.

EventHandling

Event Handling is the mechanism that controls the event and decides what should happen if anevent occurs. This mechanism have the code which is known as event handler that is executedwhen an event occurs. Java Uses the Delegation Event Model to handle the events. This modeldefines the standard mechanism to generate and handle the events.Let's have a brief introductionto thismodel.

TheDelegation EventModelhasthefollowingkeyparticipantsnamely:

- **Source** -Thesourceisanobjectonwhicheventoccurs.Sourceisresponsibleforproviding information of the occurred event to it's handler. Java provide as with classesforsourceobject.
- **Listener** -Itisalsoknownaseventhandler.Listenerisresponsibleforgeneratingresponse to an event. From java implementation point of view the listener is also anobject. Listener waits until it receives an event. Once the event is received , the listenerprocesstheevent an thenreturns.

EventclassesandListenerinterfaces:

EventClasses	ListenerInterfaces
ActionEvent	ActionListener
MouseEvent	MouseListenerandMouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

StepstoperformEvent Handling

Followingstepsarerequiredtoperformeventhandling:

1. ImplementtheListenerinterfaceandoverridesitsmethods
2. Registerthecomponent withtheListener

ForregisteringthecomponentwiththeListener,manyclassesprovidetheregistrationmethods.Forexample:

- ***Button***
  - publicvoidaddActionListener(ActionListenera){}
- ***MenuItem***
  - publicvoidaddActionListener(ActionListenera){}
- ***TextField***
  - publicvoidaddActionListener(ActionListenera){}
  - publicvoidaddTextListener(TextListenera){}
- ***TextArea***
  - publicvoidaddTextListener(TextListenera){}
- ***Checkbox***
  - publicvoidaddItemListener(ItemListenera){}
- ***Choice***
  - publicvoidaddItemListener(ItemListenera){}
- ***List***
  - publicvoidaddActionListener(ActionListenera){}
  - publicvoidaddItemListener(ItemListenera){}

EventHandlingCodes:

Wecan put the event handlingcodeinto oneof thefollowingplaces:

1. Sameclass
2. Otherclass
3. Annonymousclass

Exampleofeventhandlingwithinclass:

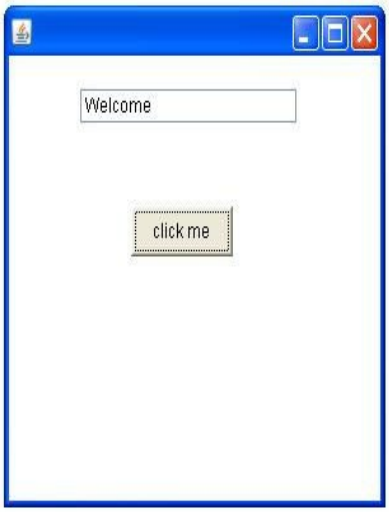
```
importjava.awt.*;
importjava.awt.event.*;
class AEvent extends Frame implements
ActionListener{TextFieldtf;
```

```
AEvent(){
tf=newTextField();tf.setBounds(6
0,50,170,20);Button b=new
Button("click
me");b.setBounds(100,120,80,30);
b.addActionListener(this);add(b);
add(tf);
setSize(300,300);
setLayout(null);s
etVisible(true);
}
public void actionPerformed(ActionEvent
e){tf.setText("Welcome");
}
publicstaticvoid main(Stringargs[]){
new AEvent();
}}
```

publicvoidsetBounds(intxaxis,intyaxis,intwidth,intheight);havebeenusedintheaboveexample that setsthe position of thecomponent it maybebutton, textfield etc.

JavaeventhandlingbyimplementingActionListener

```
importjava.awt.*;
importjava.awt.event.*;
class AEvent extends Frame implements
ActionListener{TextField tf;
AEvent(){
//create
componentstf=new
TextField();
tf.setBounds(60,50,170,20);Button
b=new Button("click
me");b.setBounds(100,120,80,30);
//register
listenerb.addActionListener(this);//passingcurrentin
stance
//add components and set size, layout and
visibilityadd(b);add(tf);
setSize(300,300);
setLayout(null);s
etVisible(true);
}
public void actionPerformed(ActionEvent
e){tf.setText("Welcome");
}
}}
```



```
publicstaticvoid main(Stringargs[]){
new AEvent();
}}
```

JavaMouseListenerInterface

The Java MouseListener is notified whenever you change the state of mouse. It is notified againstMouseEvent.TheMouseListenerinterfaceisfoundinjava.awt.eventpackage.Ithasfivemethods .

MethodsofMouseListenerinterface

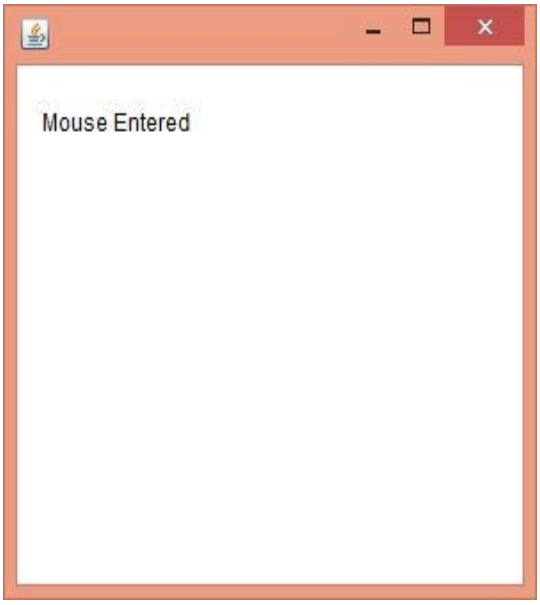
Thesignatureof5methodsfoundinMouseListenerinterface are given below:

1. publicabstractvoidmouseClicked(MouseEvente);
2. publicabstractvoidmouseEntered(MouseEvent e);
3. publicabstractvoid mouseExited(MouseEvent e);
4. publicabstractvoid mousePressed(MouseEvent e);
5. publicabstractvoid mouseReleased(MouseEvente);

JavaMouseListenerExample

```
import
java.awt.*;importjava.a
wt.event.*;
public class MouseListenerExample extends Frame implements
   MouseListener{Labell;
   MouseListenerExample(){ad
      dMouseListener(this);l=ne
      w
      Label();l.setBounds(20,50,
      100,20);add(l);
      setSize(300,300);
      setLayout(null);s
      etVisible(true);
   }
   public void mouseClicked(MouseEvent e)
      {l.setText("MouseClicked");
   }
   public void mouseEntered(MouseEvent e)
      {l.setText("MouseEntered");
   }
   public void mouseExited(MouseEvent e)
      {l.setText("MouseExited");
   }
   public void mousePressed(MouseEvent e)
      {l.setText("MousePressed");
   }
   public void mouseReleased(MouseEvent e)
      {l.setText("MouseReleased");

}}
```



```
    }
public static void main(String[] args)
    {newMouseListenerExample();
```

```
}}
```

JavaKeyListenerInterface

TheJavaKeyListenerisnotifiedwheneveryouchangethestateofkey.ItisnotifiedagainstKeyEvent.The KeyListenerinterfaceisfoundinjava.awt.eventpackage.Ithasthreemethods.

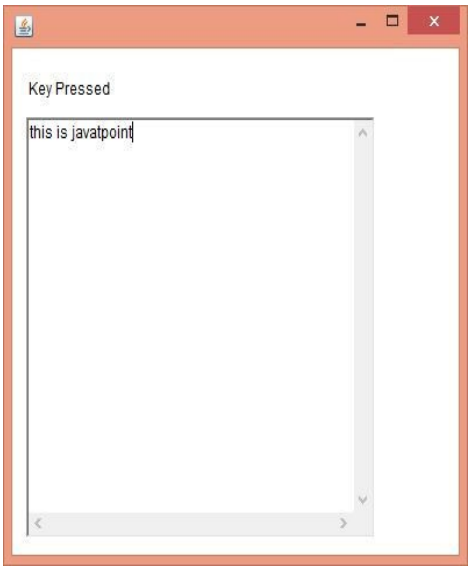
MethodsofKeyListenerinterface

Thesignatureof3methodsfoundinKeyListener interfacearegivenbelow:

1. publicabstractvoidkeyPressed(KeyEvente);
2. publicabstractvoidkeyReleased(KeyEvente);
3. publicabstractvoidkeyTyped(KeyEvent e);

JavaKeyListenerExample

```
import
java.awt.*;importjava.a
wt.event.*;
public class KeyListenerExample extends Frame implements
   KeyListener{Labell;
   TextArea
   area;KeyListenerExamp
   le(){
      l=new
      Label();l.setBounds(20,50,100,2
      0);area=new
      TextArea();area.setBounds(20,8
      0,300,
      300);area.addKeyListener(this);
      add(l);add(area);setSize(400,40
      0);setLayout(null);setVisible(tru
      e);
   }
   public void keyPressed(KeyEvent e)
      {l.setText("KeyPressed");
   }
   public void keyReleased(KeyEvent e)
      {l.setText("KeyReleased");
   }
   public void keyTyped(KeyEvent e)
      {l.setText("KeyTyped");
   }
   public static void main(String[] args)
      {newKeyListenerExample();}}
```



JavaAdapterClasses

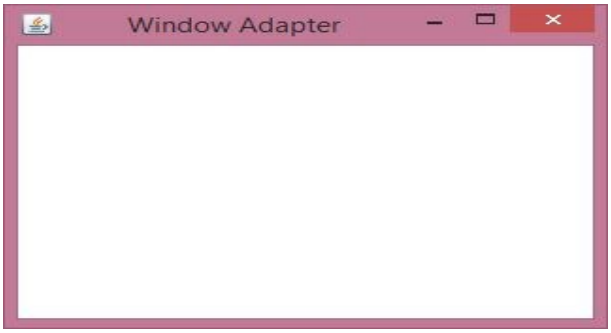
Java adapter classes *provide the default implementation of listener interfaces*. If you inherit theadapter class, you will not be forced to provide the implementation of all the methods of

listenerinterfaces.So it*saves code*.

java.awt.eventAdapterclasses

Adapterclass	Listenerinterface
WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
FocusAdapter	FocusListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
HierarchyBoundsAdapter	HierarchyBoundsListener

JavaWindowAdapterExample




```
1.importjava.awt.*;
importjava.awt.event.*;
public class
   AdapterExample{Framef;A
   dapterExample(){
      f=newFrame("WindowAdapter");f.addWindo
      wListener(newWindowAdapter(){
         public void windowClosing(WindowEvent e)
            {f.dispose();   }    });
      f.setSize(400,400);
      f.setLayout(null);f
      .setVisible(true);
   }
publicstaticvoidmain(String[] args){
   newAdapterExample();
}}
```

Applets

Appletisaspecialtypeofprogramthatisembeddedinthewebpagetogeneratethedynamiccontent.Itruns insidethe browserand works atclientside.

AdvantageofApplet

Therearemanyadvantages of applet. Theyareasfollows:

- Itworksatclient sidesoless responsetime.
- Secured
- Itcanbeexecutedbybrowsersrunningundermanyplateforms,includingLinux,Windows,MacO s etc.

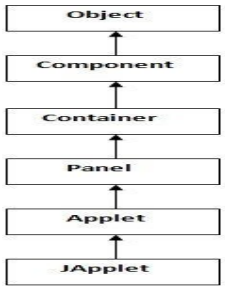
Drawbackof Applet

- Pluginisrequiredatclientbrowsertoexecuteapplet.

LifecycleofJavaApplet

1. Appletisinitialized.
2. Appletisstarted.
3. Appletis painted.
4. Appletisstopped.
5. Appletisdestroyed.

HierarchyofApplet



LifecyclemethodsforApplet:

Thejava.applet.Appletclass4lifecyclemethodsandjava.awt.Componentclassprovides1lifecyclemeth ods foran applet.

java.applet.Appletclass

For creating any applet java.applet.Applet class must be inherited. It provides 4 life cycle methodsofapplet.

1. **publicvoidinit():** isused toinitialized theApplet.Itis invokedonlyonce.
2. **publicvoidstart():**isinvokedaftertheinit()methodorbrowserismaximized.Itisusedtostart theApplet.
3. **publicvoidstop():**isusedtostoptheApplet.ItisinvokedwhenAppletisstoporbrowseris minimized.

4. **publicvoid destroy():** is used todestroytheApplet.It isinvoked onlyonce.

java.awt.Componentclass

TheComponent class provides1 lifecyclemethod of applet.

1.**publicvoidpaint(Graphicsg):**isusedtopainttheApplet.ItprovidesGraphicsclassobjectthat can beusedfor drawingoval,rectangle, arcetc.

SimpleexampleofAppletby htmlfile:

To execute the applet by html file, create an applet and compile it. After that create an html fileandplacetheapplet codein html file. Nowclick the html file.

```
1.//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends
Applet{public void paint(Graphics
g){g.drawString("welcome",150,1
50);
}
}
```

SimpleexampleofAppletbyappletviewertool:

To execute the applet by appletviewer tool, create an applet that contains applet tag in commentand compile it. After that run it by: appletviewer First.java. Now Html file is not required but it isfortestingpurposeonly.

```
1.//First.java
import java.applet.Applet;
import java.awt.Graphics;
publicclassFirstextends Applet{
publicvoidpaint(Graphics
g){g.drawString("welcometoapplet",150,150);
}
}
/*
<appletcode="First.class"width="300"height="300">
</applet>
```

```
*/
```

Toexecute theappletbyappletviewer tool,write in commandprompt:

```
c:\>javacFirst.java
c:\>appletviewerFirst.java
```

DifferencebetweenAppletandApplication programming

	Java Applet	Java Application
User graphics	Inherently graphical	Optional
Memory requirements	Java application requirements plus web browser requirements	Minimal java application requirements
Distribution	Linked via HTML and transported via HTTP	Loaded from the file system or by a custom class loading process
Environmental input	Browser client location and size; parameters embedded in the host HTML document	command-line parameters
Method expected by the virtual Machine	init- initialization method start-startup method stop pause/ deactive method destroy-termination method paint-drawing method	Main - startup method
Typical applications	public-access order-entry systems for the web, online multimedia persentations, web page animation	Network server, multimedia kiosks, developer tools, appliance and consumer electronics control and navigation.

ParameterinApplet

WecangetanyinformationfromtheHTMLfileasaparameter.Forthispurpose,Appletclassprovidesamethod named getParameter(). Syntax:

1.**public**StringgetParameter(String parameterName)

Exampleofusing parameterinApplet:

```
1. importjava.applet.Applet;
2. importjava.awt.Graphics;
3. public class UseParam extends Applet4.{
5.public void paint(Graphics g)6.{
7. Stringstr=getParameter("msg");
8. g.drawString(str,50,50);
9.}}
```

myapplet.html

```
1. <html>
2. <body>
3. <appletcode="UseParam.class"width="300"height="300">
4. <paramname="msg"value="Welcometo applet">
5. </applet>
6. </body>
7. </html>
```