



Java Programming

Introduction

- Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language.
- Java was developed by **Sun Microsystems** (which is now the subsidiary of Oracle) in the year **1995**.
- **James Gosling** is known as the father of Java. Before Java, its name was Oak.
- Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

Application

- Desktop Applications such as acrobat reader, media player, antivirus, etc.
- Web Applications such as irctc.co.in
- Enterprise Applications such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games, etc.

Types of Java Applications

There are mainly 4 types of applications that can be created using Java programming:

1) Standalone Application

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. **AWT and Swing** are used in Java for creating standalone applications.

2) Web Application

An application that runs on the server side and creates a dynamic page is called a web application. Currently, **Servlet, JSP, Struts, Spring, Hibernate, JSF**, etc. technologies are used for creating web applications in Java.

3) Enterprise Application

An application that is distributed in nature, such as banking applications, etc. is called an enterprise application. It has advantages like high-level security, load balancing, and clustering. In Java, **EJB** is used for creating enterprise applications.

4) Mobile Application

An application which is created for mobile devices is called a mobile application. Currently, **Android and Java ME** are used for creating mobile applications.

Java Platforms / Editions

There are 4 platforms or editions of Java:

1) Java SE (Java Standard Edition)

It is a Java programming platform. It includes Java programming APIs such as `java.lang`, `java.io`, `java.net`, `java.util`, `java.sql`, `java.math` etc. It includes core topics like OOPs, [String](#), Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

2) Java EE (Java Enterprise Edition)

It is an enterprise platform that is mainly used to develop web and enterprise applications. It is built on top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, [JPA](#), etc.

3) Java ME (Java Micro Edition)

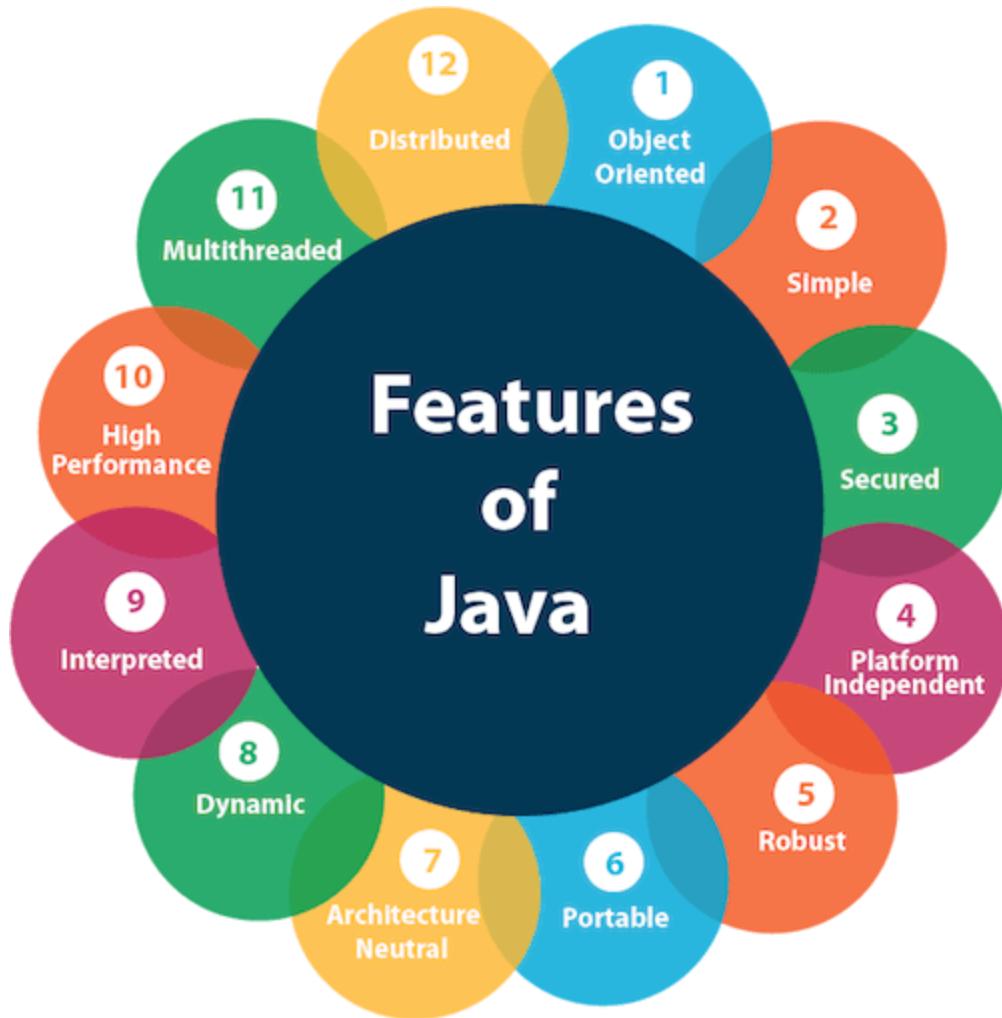
It is a micro platform that is dedicated to mobile applications.

4) JavaFX

It is used to develop rich internet applications. It uses a lightweight user interface API.

History of Java

- 1) **James Gosling, Mike Sheridan, and Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.
- 2) Initially it was designed for small, [embedded systems](#) in electronic appliances like set-top boxes.
- 3) Firstly, it was called "**Greentalk**" by James Gosling, and the file extension was .gt.
- 4) After that, it was called **Oak** and was developed as a part of the Green project.
- 5) **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.
- 6) In 1995, Oak was renamed as "**Java**" because it was already a trademark by Oak Technologies.
- 7) Why had they chose the name Java for Java language? The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA", etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell, and fun to say.
According to James Gosling, "Java was one of the top choices along with **Silk**". Since Java was so unique, most of the team members preferred Java than other names.
- 8) Java is an island in Indonesia where the first coffee was produced (called Java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having a cup of coffee nearby his office
- 9) Notice that Java is just a name, not an acronym.
- 10) Initially developed by James Gosling at [Sun Microsystems](#) (which is now a subsidiary of Oracle Corporation) and released in 1995.
- 11) In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.
- 12) JDK 1.0 was released on January 23, 1996. After the first release of Java, there have been many additional features added to the language. Now Java is being used in Windows applications, Web applications, enterprise applications, mobile applications, cards, etc. Each new version adds new features in Java.



Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Microsystem, Java language is a simple programming language because:

- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

Object-oriented

Java is an [object-oriented](#) programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.

Basic concepts of OOPs are:

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

Platform Independent

Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

There are two types of platforms software-based and hardware-based. Java provides a software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on top of other hardware-based platforms. It has two components:

- Runtime Environment
- API(Application Programming Interface)

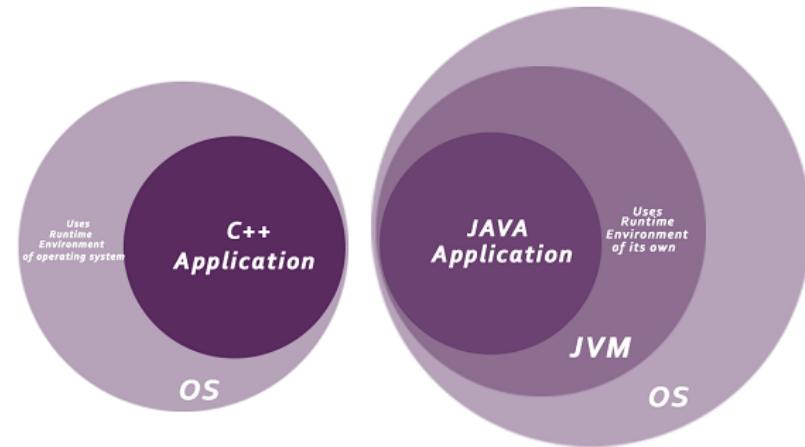
Java code can be executed on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere (WORA).

Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

No explicit pointer

Java Programs run inside a virtual machine sandbox



Classloader: Classloader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.

Bytecode Verifier: It checks the code fragments for illegal code that can violate access rights to objects.

Security Manager: It determines what resources a class can access such as reading and writing to the local disk.

Java language provides these securities by default. Some security can also be provided by an application developer explicitly through SSL, JAAS, Cryptography, etc

Robust

The English mining of Robust is strong. Java is robust because:

- It uses strong memory management.
- There is a lack of pointers that avoids security problems.
- Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

Portable

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

Distributed

Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

Dynamic

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in Windows-based, web-based, enterprise, and mobile applications.
Design Goal	C++ was designed for systems and applications programming. It was an extension of the C programming language .	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed to be easy to use and accessible to a broader audience.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by using interfaces in java .
Operator Overloading	C++ supports operator overloading .	Java doesn't support operator overloading.
Pointers	C++ supports pointers . You can write a pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses both compiler and interpreter. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform-independent.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.
Documentation comment	C++ doesn't support documentation comments.	Java supports documentation comment (/** ... */) to create documentation for java source code.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not to override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
unsigned right shift >>	C++ doesn't support >> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.
Inheritance Tree	C++ always creates a new inheritance tree.	Java always uses a single inheritance tree because all classes are the child of the Object class in Java. The Object class is the root of the inheritance tree in java.
Hardware	C++ is nearer to hardware.	Java is not so interactive with hardware.
Object-oriented	C++ is an object-oriented language. However, in the C language, a single root hierarchy is not possible.	Java is also an object-oriented language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.

JVM (Java Virtual Machine) Architecture

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

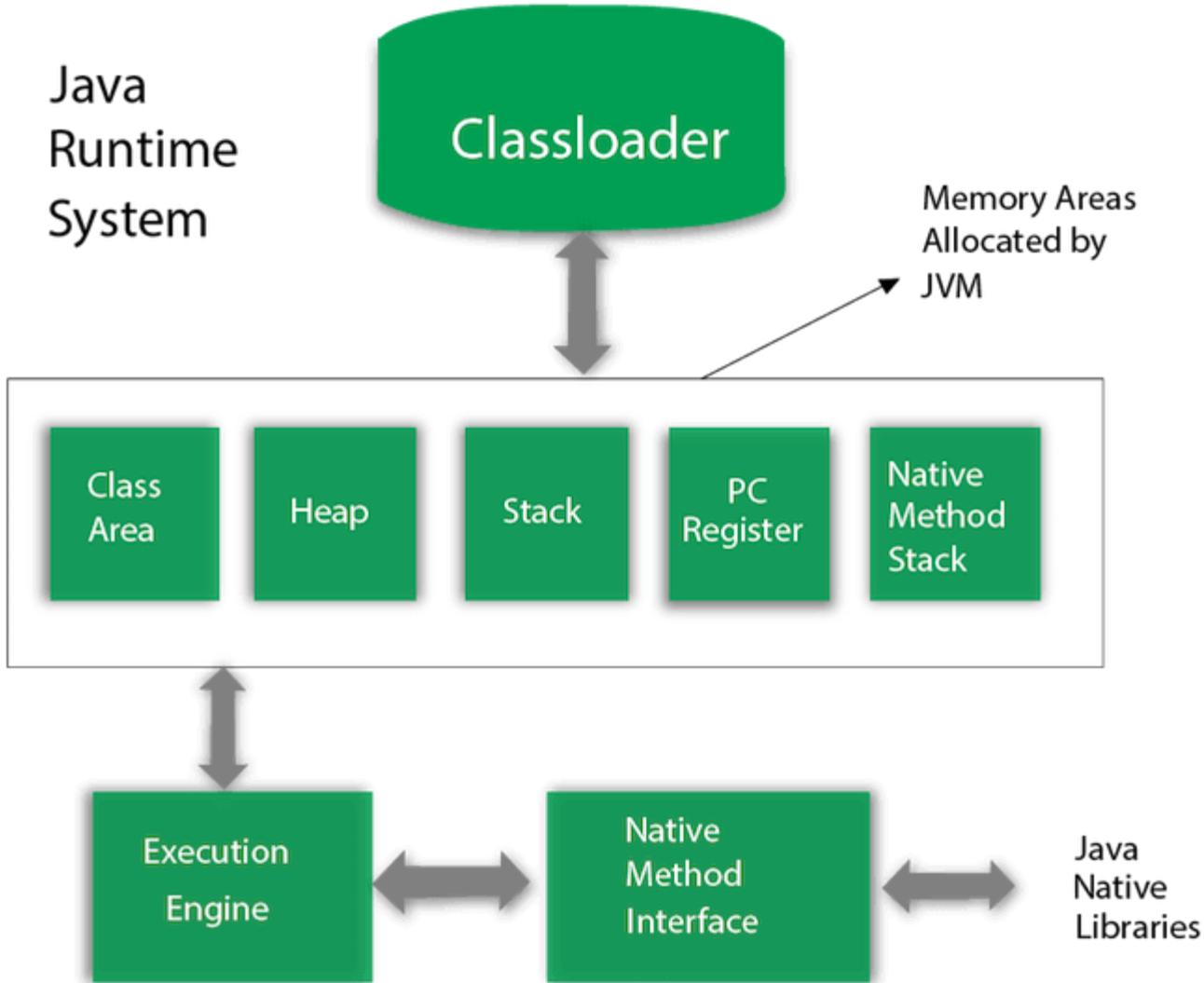
JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.



1) Classloader

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

Bootstrap ClassLoader: This is the first classloader which is the super class of Extension classloader. It loads the rt.jar file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes etc.

Extension ClassLoader: This is the child classloader of Bootstrap and parent classloader of System classloader. It loads the jar files located inside \$JAVA_HOME/jre/lib/ext directory.

System/Application ClassLoader: This is the child classloader of Extension classloader. It loads the classfiles from classpath. By default, classpath is set to current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

A virtual processor

Interpreter: Read bytecode stream then execute the instructions.

Just-In-Time(JIT) compiler: It is used to improve the performance.

JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.

Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

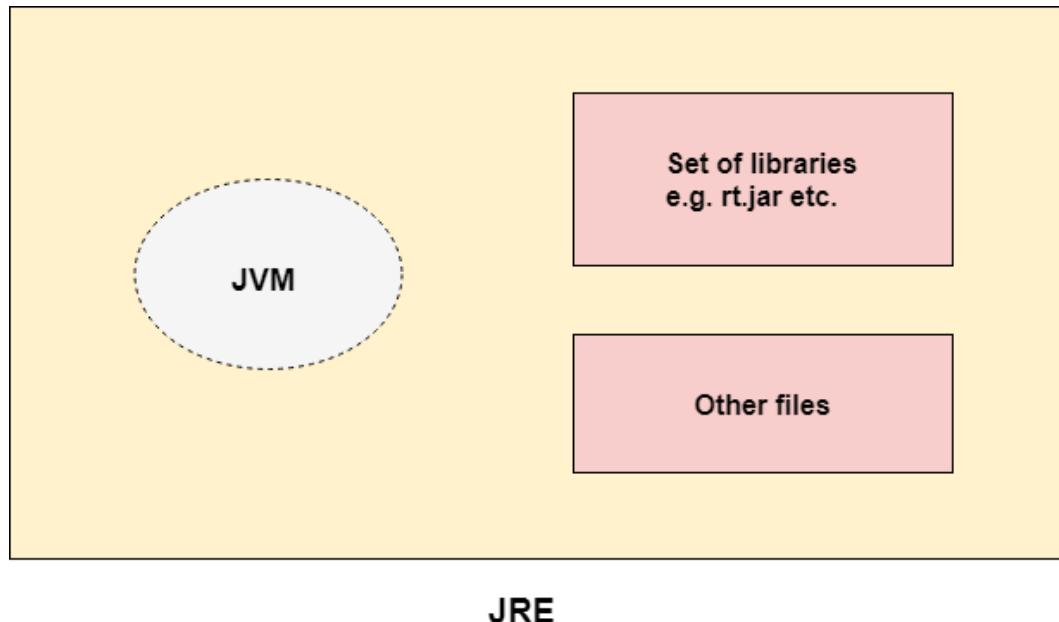
8) Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc.

Java uses JNI framework to send output to the Console or interact with OS libraries.

JRE

JRE is an acronym for **Java Runtime Environment**. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists.



JDK

JDK is an acronym for **Java Development Kit**. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.

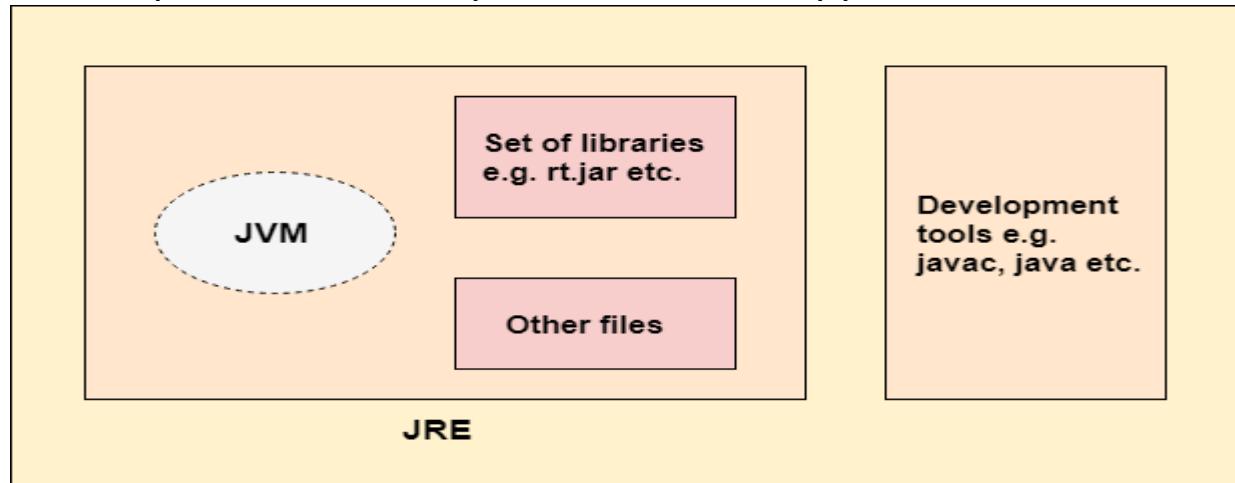
JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

Standard Edition Java Platform

Enterprise Edition Java Platform

Micro Edition Java Platform

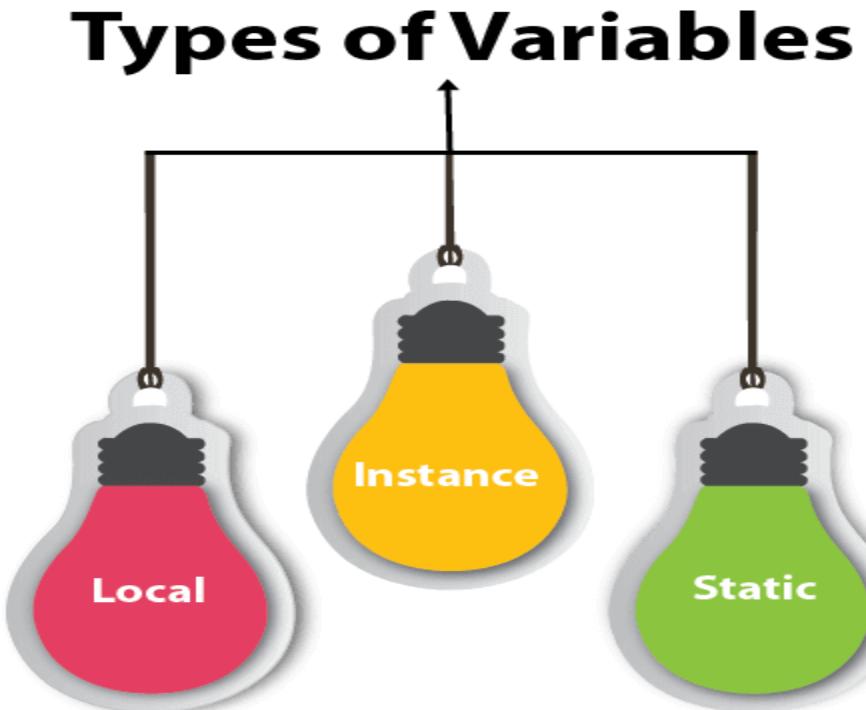
The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.



Java is a **statically-typed** programming language. It means, all variables must be declared before its use. That is why we need to declare variable's type and name.

Variable

A variable is the name of a reserved area allocated in memory. In other words, it is a name of the memory location. It is a combination of "vary + able" which means its value can be changed



Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Unicode System

Unicode is a universal international standard character encoding that is capable of representing most of the world's written languages.

Before Unicode, there were many language standards:

ASCII (American Standard Code for Information Interchange) for the United States.

ISO 8859-1 for Western European Language.

KOI-8 for Russian.

GB18030 and **BIG-5** for Chinese, and so on.

This caused two problems:

A particular code value corresponds to different letters in the various language standards.

The encodings for languages with large character sets have variable length.

Some common characters are encoded as single bytes, others require two or more bytes.

To solve these problems, a new language standard was developed i.e. Unicode System.

In unicode, character holds 2 byte, so Java also uses 2 byte for characters.

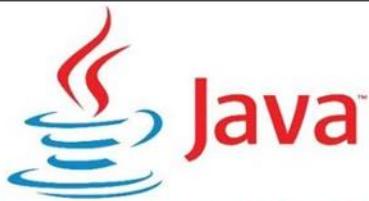
lowest value:\u0000

highest value:\uFFFF

abstract	continue	for
assert ***	default	goto *
boolean	do	if
break	double	implements
byte	else	import
case	enum ****	instanceof
catch	extends	int
char	final	interface
class	finally	long
const *	float	native

```
new                      switch
package                  synchronized
private                  this
protected                throw
public                   throws
return                   transient
short                    try
static                   void
strictfp**              volatile
super
```

Total 50 keywords, 2 keywords **const** and **goto** are reserved, even though they are not currently used. true, false, and null might seem like keywords, but they are actually literals



Java Naming Conventions

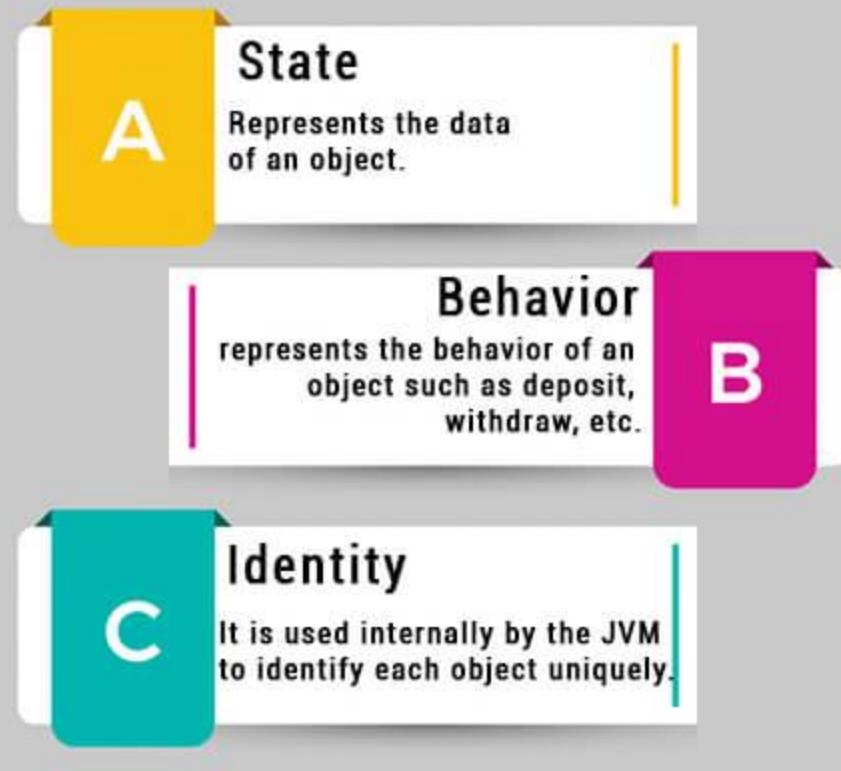
Name	Convention
Class Name	Should start with uppercase letter and be a noun e.g. <code>String</code> , <code>Color</code> , <code>Button</code> , <code>System</code> , <code>Thread</code> , <code>ArrayList</code> , <code>StringBuffer</code> etc.
Interface Name	Should start with uppercase letter and be an adjective e.g. <code>Runnable</code> , <code>Remote</code> , <code>List</code> , <code>ActionListener</code> etc.
Method Name	Should start with lowercase letter and be a verb e.g. <code>actionPerformed()</code> , <code>main()</code> , <code>print()</code> , <code>println()</code> etc.
Variable Name	Should start with lowercase letter e.g. <code>firstName</code> , <code>lastName</code> , <code>orderNumber</code> etc.
Package Name	Should be in lowercase letter e.g. <code>java.lang</code> , <code>java.sql</code> , <code>java.util</code> etc.
Constants Name	should be in uppercase letter. e.g. <code>RED</code> , <code>YELLOW</code> , <code>MAX_PRIORITY</code> etc.

OOP's Concepts

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

OBJECT	CLASS
Object is an instance of a class.	Class is a blueprint from which objects are created
Object is a real world entity such as chair, pen, table, laptop etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocate memory when it is created.
Object is created through new keyword. Employee ob = new Employee();	Class is declared using class keyword. class Employee{}
There are different ways to create object in java:- New keyword, newInstance() method, clone() method, And deserialization.	There is only one way to define a class, i.e., by using class keyword.

Characteristics of Object



A class in Java can contain:

- Fields
- Methods
- Constructors
- Blocks
- Nested class and interface

Inheritance

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism

If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

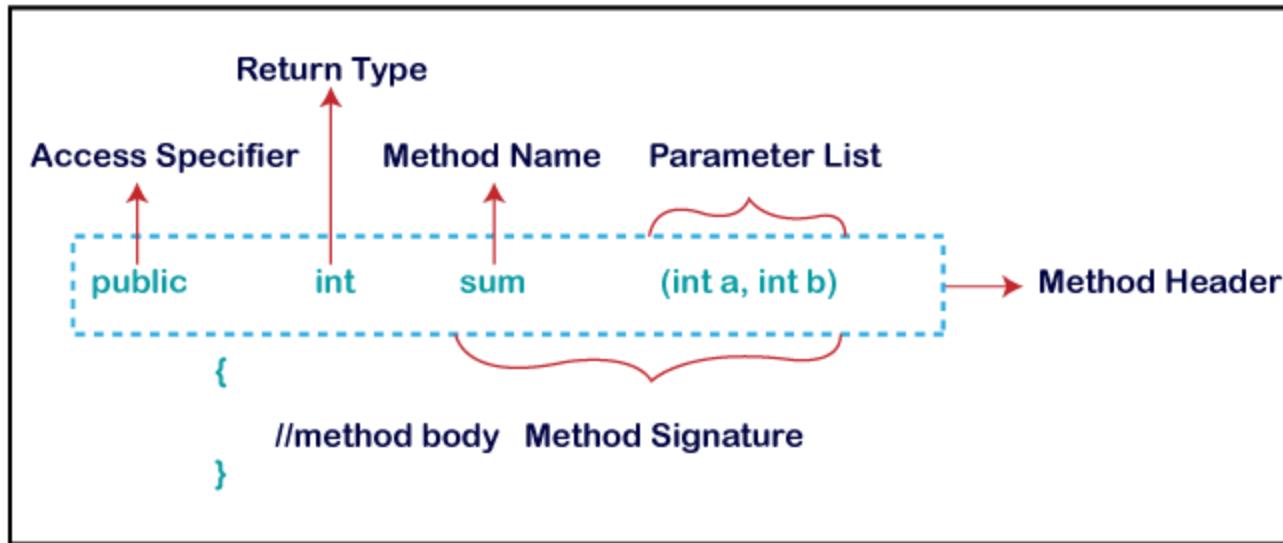
In Java, we use abstract class and interface to achieve abstraction

Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Method Declaration



Method Signature: Every method has a method signature. It is a part of the method declaration. It includes the method name and parameter list.

Constructors in Java

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the new() keyword, at least one constructor is called.

It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.

Rules for creating Java constructor

- Constructor name must be the same as its class name
- A Constructor must have no explicit return type
- A Java constructor cannot be abstract, static, final, and synchronized

There are **two types** of constructors in Java:

- **Default constructor (no-arg constructor)**

```
<class_name>()
{
}
```

Rule: If there is no constructor in a class, compiler automatically creates a default constructor

- **Parameterized constructor**

```
<class_name>(parameters list)
{
}
```

Constructor overloading in Java is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task. They are differentiated by the compiler by the number of parameters in the list and their types.

```
public class Demo {  
    Demo() {  
        ...  
    }  
    Demo(String s) {  
        ...  
    }  
    Demo(int i) {  
        ...  
    }  
    ...  
}
```

Beginnersbook.com

Three overloaded constructors - They must have different Parameters list

If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.

```
class Cat{  
    public void Sound(){  
        System.out.println("meow");  
    }  
  
    //overloading method  
    public void Sound(int num){  
        for(int i=0; i<2;i++){  
            System.out.println("meow");  
        }  
    }  
}
```

OVERLOADING

**Same method
name but
different
parameters**

Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as the class name.

The **static keyword in Java** is used for memory management mainly.

The static can be:

- Variable (also known as a class variable)
- Method (also known as a class method)
- Block
- Nested class

1) Java static variable

If you declare any variable as static, it is known as a static variable.

The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.

The static variable gets memory only once in the class area at the time of class loading.

2) Java static method

If you apply static keyword with any method, it is known as static method.

- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access static data member and can change the value of it.

Restrictions for the static method

There are two main restrictions for the static method. They are:

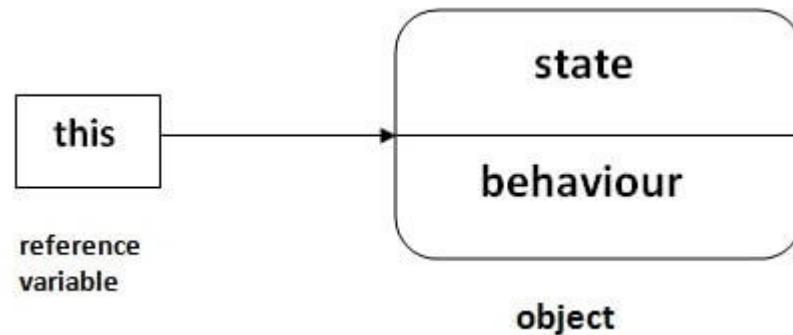
- The static method can not use non static data member or call non-static method directly.
- this and super cannot be used in static context.

3) Java static block

Is used to initialize the static data member.

It is executed before the main method at the time of classloading.

There can be a lot of usage of Java **this** keyword. In Java, this is a reference variable that refers to the current object.



Usage of Java this keyword

Here is given the 6 usage of java this keyword.

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor. (**Rule: Call to this() must be the first statement in constructor**)
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

Java Arrays

Normally, an array is a collection of similar type of elements which has contiguous memory location.

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location.

In Java, array is an object of a dynamically generated class. Java array inherits the Object class, and implements the Serializable as well as Cloneable interfaces.

Code Optimization: It makes the code optimized, we can retrieve or sort the data efficiently.

Random access: We can get any data located at an index position.

Single Dimensional Array in Java

Syntax to Declare an Array in Java

```
dataType[] arr; (or)  
dataType []arr; (or)  
dataType arr[];
```

Instantiation of an Array in Java

```
arrayRefVar=new datatype[size];
```

The Java Virtual Machine (JVM) throws an **ArrayIndexOutOfBoundsException** if length of the array is negative, equal to the array size or greater than the array size while traversing the array.

Multidimensional Array in Java

In such case, data is stored in row and column based index (also known as matrix form).

Syntax

```
dataType[][] arrayRefVar; (or)  
dataType [][]arrayRefVar; (or)  
dataType arrayRefVar[][]; (or)  
dataType []arrayRefVar[];
```

Example to instantiate Multidimensional Array in Java

```
int[][] arr=new int[3][3];//3 row and 3 column
```

Jagged Array in Java

If we are creating odd number of columns in a 2D array, it is known as a jagged array. In other words, it is an array of arrays with different number of columns.

```
//declaring a 2D array with odd columns
int arr[][] = new int[3][];
arr[0] = new int[3];
arr[1] = new int[4];
arr[2] = new int[2];
```

Java String

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string.

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



Same as

```
String s="javatpoint";
```

No.	Method	Description
1	char charAt(int index)	It returns char value for the particular index
2	int length()	It returns string length
3	static String format(String format, Object... args)	It returns a formatted string.
4	static String format(Locale l, String format, Object... args)	It returns formatted string with given locale.
5	String substring(int beginIndex)	It returns substring for given begin index.
6	String substring(int beginIndex, int endIndex)	It returns substring for given begin index and end index.
7	boolean contains(CharSequence s)	It returns true or false after matching the sequence of char value.
8	static String join(CharSequence delimiter, CharSequence... elements)	It returns a joined string.
9	static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)	It returns a joined string.
10	boolean equals(Object another)	It checks the equality of string with the given object.

11	<code>boolean isEmpty()</code>	It checks if string is empty.
12	<code>String concat(String str)</code>	It concatenates the specified string.
13	<code>String replace(char old, char new)</code>	It replaces all occurrences of the specified char value.
14	<code>String replace(CharSequence old, CharSequence new)</code>	It replaces all occurrences of the specified CharSequence.
15	<code>static String equalsIgnoreCase(String another)</code>	It compares another string. It doesn't check case.
16	<code>String[] split(String regex)</code>	It returns a split string matching regex.
17	<code>String[] split(String regex, int limit)</code>	It returns a split string matching regex and limit.
18	<code>String intern()</code>	It returns an interned string.
19	<code>int indexOf(int ch)</code>	It returns the specified char value index.
20	<code>int indexOf(int ch, int fromIndex)</code>	It returns the specified char value index starting with given index.

21	<code>int indexOf(String substring)</code>	It returns the specified substring index.
22	<code>int indexOf(String substring, int fromIndex)</code>	It returns the specified substring index starting with given index.
23	<code>String toLowerCase()</code>	It returns a string in lowercase.
24	<code>String toLowerCase(Locale l)</code>	It returns a string in lowercase using specified locale.
25	<code>String toUpperCase()</code>	It returns a string in uppercase.
26	<code>String toUpperCase(Locale l)</code>	It returns a string in uppercase using specified locale.
27	<code>String trim()</code>	It removes beginning and ending spaces of this string.
28	<code>static String valueOf(int value)</code>	It converts given type into string. It is an overloaded method.

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.

Inheritance represents the **IS-A** relationship which is also known as a parent-child relationship.

Why use inheritance in java

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

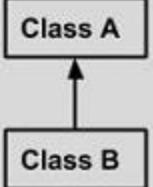
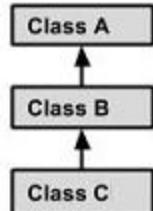
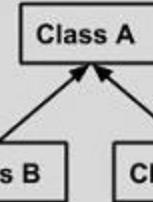
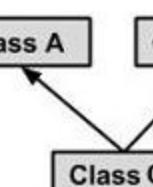
Terms used in Inheritance

Class: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

Sub Class/Child Class: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

Super Class/Parent Class: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

Reusability: As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

Single Inheritance		public class A { } public class B extends A { }
Multi Level Inheritance		public class A { } public class B extends A { } public class C extends B { }
Hierarchical Inheritance		public class A { } public class B extends A { } public class C extends A { }
Multiple Inheritance		public class A { } public class B { } public class C extends A,B { } } // Java does not support multiple inheritance

Method Overriding in Java

Understanding the problem without method overriding

Can we override the static method

Method overloading vs. method overriding

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

Usage of Java Method Overriding

Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.

Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

- The method must have the same name as in the parent class
- The method must have the same parameter as in the parent class.
- There must be an IS-A relationship (inheritance).

Method overloading	Method overriding
<ol style="list-style-type: none"> 1. The process of defining functions having the same name with different parameter list is called as overloading method. 2. It is a relationship between methods in the same class. 3. It is static binding—at the compile time. 4. It is the concept of compile time polymorphism or early binding. 5. It may or may not be observed during inheritance. 6. Return types do not affect overloading. 	<ol style="list-style-type: none"> 1. When methods of the subclass having same name as that of superclass, overrides the methods of the superclass then it is called as overriding methods. 2. It is a relationship between subclass method and a superclass method. 3. It is dynamic binding—at the runtime. 4. It is the concept of run-time polymorphism or late binding. 5. It is observed during inheritance. 6. Return types, method names and signatures, both overridden methods must be identical.

Super Keyword in Java

The super keyword in Java is a reference variable which is used to refer immediate parent class object.

Usage of Java super Keyword

- super can be used to refer immediate parent class instance variable.
- super can be used to invoke immediate parent class method.
- super() can be used to invoke immediate parent class constructor

Note: super() is added in each class constructor automatically by compiler if there is no super() or this().

Java Final Keyword

Final Variable



Stop value change

Final Method



Prevent Method Overriding

Final Class



Prevent Inheritance

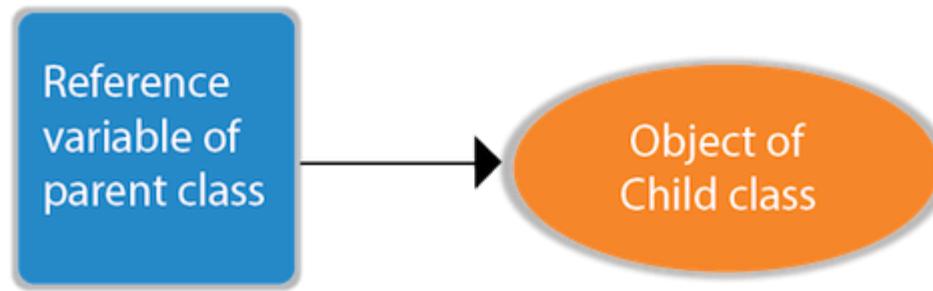
Runtime Polymorphism in Java

Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

Upcasting

If the reference variable of Parent class refers to the object of Child class, it is known as upcasting



```
class A{}  
class B extends A{}
```

```
A a=new B();//upcasting
```

Abstract class in Java

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

There are two ways to achieve abstraction in java

- Abstract class (0 to 100%)
- Interface (100%)

Abstract class in Java

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Points to Remember

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

```
abstract class Bike
{
    abstract void run();
}
class Honda4 extends Bike
{
void run()
{
System.out.println("running safely");
}
public static void main(String args[])
{
    Bike obj = new Honda4();
    obj.run();
}
```

Let us summarize the points on abstract class:

- An abstract class is a class that contains 0 or more abstract methods.
- An abstract class can contain instance variables and concrete methods in addition to abstract methods.
- Abstract class and the abstract methods should be declared by using the key word 'abstract'.
- All the abstract methods of the abstract class should be implemented (body) in its sub classes.
- If any abstract method is not implemented, then that sub class should be declared as 'abstract'. In this case, we cannot create an object to the sub class. We should create another sub class to this sub class and implement the remaining abstract method there.
- We cannot create an object to abstract class.
- But, we can create a reference of abstract class type.
- The reference of abstract class can be used to refer to objects of its subclasses.
- The reference of abstract class can not refer to individual methods of its subclasses.
- It is possible to derive an abstract class as a sub class from a concrete super class.
- We cannot declare a class as both abstract and final. For example,

Interface in Java

An interface in Java is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is a mechanism to achieve abstraction.

Why use Java interface?

There are mainly three reasons to use interface. They are given below.

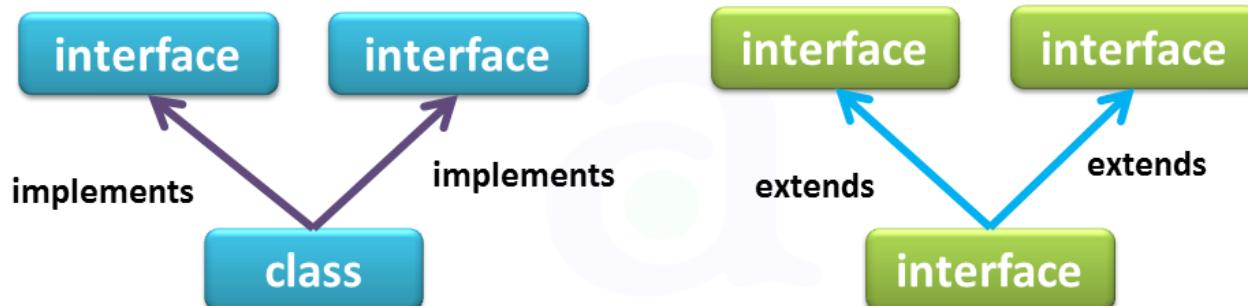
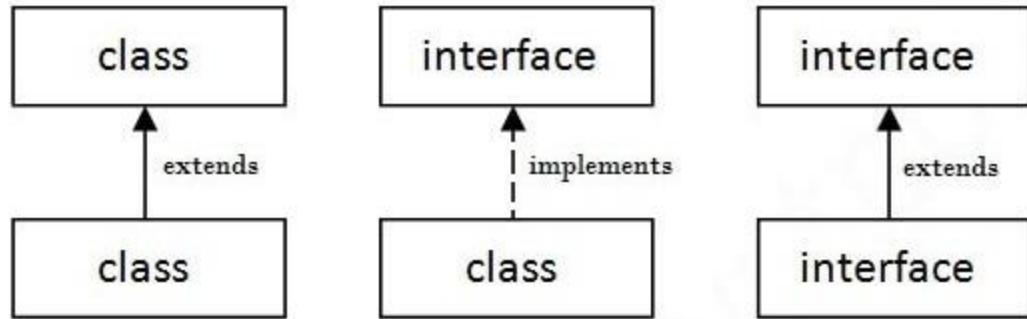
- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

Syntax:

```
interface <interface_name>
{
```

```
// declare constant fields
// declare methods that abstract
// by default.
}
```

The Java compiler adds public and abstract keywords before the interface method. Moreover, it adds public, static and final keywords before data members.



Multiple Inheritance in Java

Since Java 8, interface can have default and static methods.

An interface which has no member is known as a **marker or tagged interface**, for example, Serializable, Cloneable, Remote, etc. They are used to provide some essential information to the JVM so that JVM may perform some useful operation.

Summary

Now, let us summarize the following points on interfaces, before we proceed further.

- An interface is a specification of method prototypes. This means, only method names are written in the interface without method bodies.
- An interface will have 0 or more abstract methods which are all public and abstract by default.
- An interface can have variables which are public static and final by default. This means all the variables of the interface are constants.
- None of the methods in Interface can be private, protected or static.
- We cannot create an object to an interface, but we can create a reference of interface type.
- All the methods of interface should be implemented in its implementation classes. If any method is not implemented, then that implementation class should be declared as 'abstract'.
- Interface reference can refer to the objects of its implementation classes.
- When an interface is written, any third party vendor can provide implementation classes to it.
- An interface can extend another interface.
- An interface cannot implement another interface.
- It is possible to write a class within an interface.

Interface forces the implementation classes to implement all of its methods compulsory. Java compiler checks whether all the methods are implemented in the implementation classes or not.

A class can implement (not extend) multiple interfaces. For example, we can write:

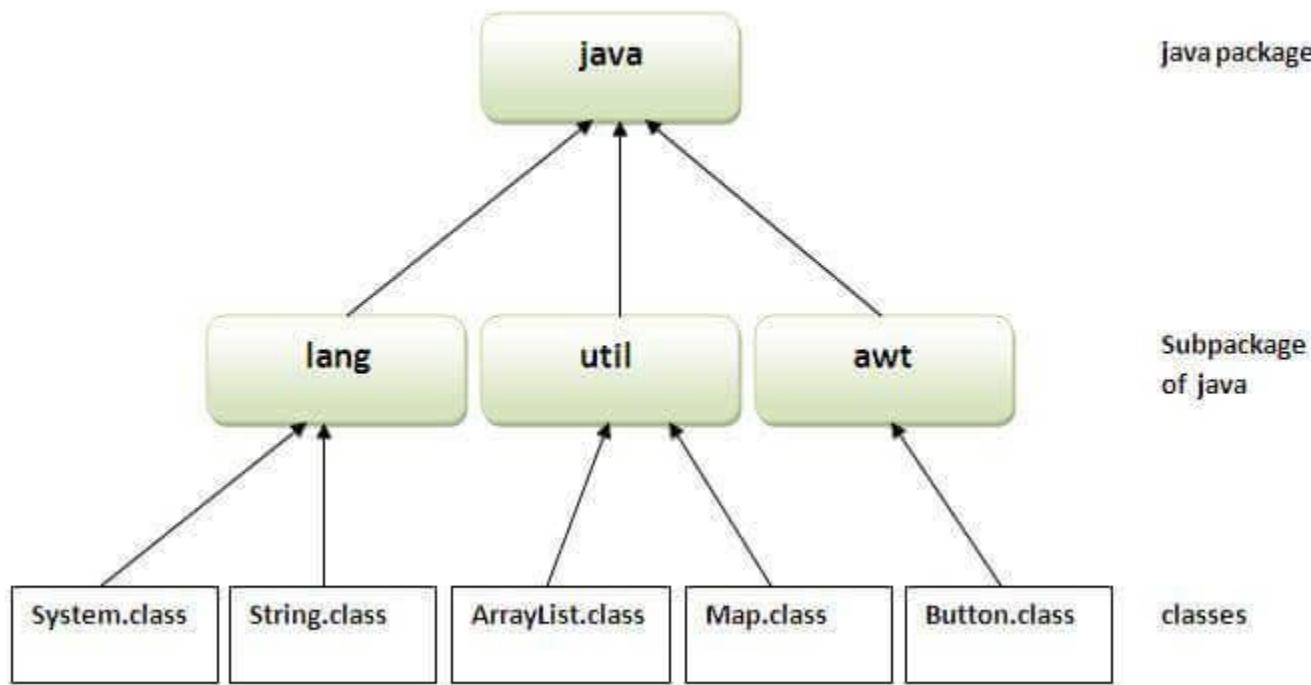
Abstract class	Interface
1) Abstract class can have abstract and non-abstract methods.	Interface can have only abstract methods. Since Java 8, it can have default and static methods also.
2) Abstract class doesn't support multiple inheritance.	Interface supports multiple inheritance.
3) Abstract class can have final, non-final, static and non-static variables.	Interface has only static and final variables.
4) Abstract class can provide the implementation of interface.	Interface can't provide the implementation of abstract class.
5) The abstract keyword is used to declare abstract class.	The interface keyword is used to declare interface.
6) An abstract class can extend another Java class and implement multiple Java interfaces.	An interface can extend another Java interface only.
7) An abstract class can be extended using keyword "extends".	An interface can be implemented using keyword "implements".
8) A Java abstract class can have class members like private, protected, etc.	Members of a Java interface are public by default.
9) Example: public abstract class Shape{ public abstract void draw(); }	Example: public interface Drawable{ void draw(); }

Java Package

A java package is a group of similar types of classes, interfaces and sub-packages

Advantage of Java Package

- 1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.
- 2) Java package provides access protection.
- 3) Java package removes naming collision.



```
package mypack;  
public class Simple  
{  
    public static void main(String args[])  
    {  
        System.out.println("Welcome to package");  
    }  
}
```

How to access package from another package?

There are three ways to access the package from outside the package.

- import package.*;
- import package.classname;
- fully qualified name.

Example of package that import the packagename.*

```
//save by A.java
package pack;
public class A{
    public void msg(){System.out.println("Hello");}
}
```

```
//save by B.java
package mypack;
import pack.*;

class B{
    public static void main(String args[]){
        A obj = new A();
        obj.msg();
    }
}
```

Output:Hello

Example of package by import package.classname

```
//save by A.java

package pack;
public class A{
    public void msg(){System.out.println("Hello");}
}
```

```
//save by B.java
package mypack;
import pack.A;

class B{
    public static void main(String args[]){
        A obj = new A();
        obj.msg();
    }
}
```

Output:Hello

Example of package by import fully qualified name

```
//save by A.java
package pack;
public class A{
    public void msg(){System.out.println("Hello");}
}
```

```
//save by B.java
package mypack;
class B{
    public static void main(String args[]){
        pack.A obj = new pack.A();//using fully qualified name
        obj.msg();
    }
}
```

Output:Hello

- The private access modifier is accessible only within the class.
- If you don't use any modifier, it is treated as default by default. The default modifier is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than private. But, it is more restrictive than protected, and public.
- The protected access modifier is accessible within package and outside the package but through inheritance only. The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class. It provides more accessibility than the default modifier.
- The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

I/O Streams

Java I/O Tutorial

Java I/O (Input and Output) is used to process the input and produce the output.

Java uses the concept of a stream to make I/O operation fast. The `java.io` package contains all the classes required for input and output operations.

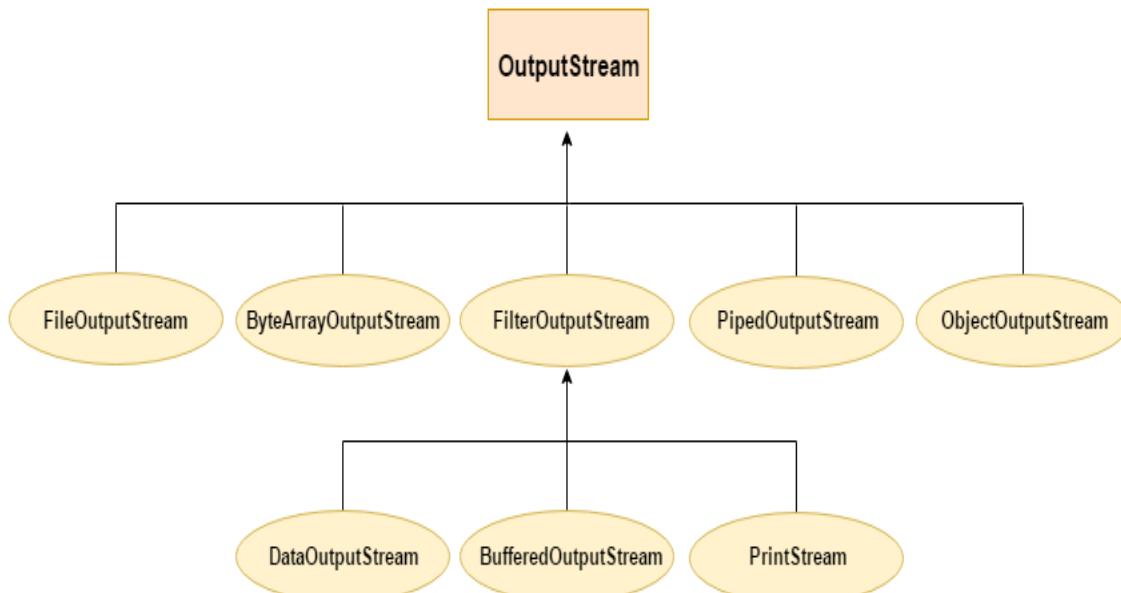
We can perform file handling in Java by Java I/O API.

Stream

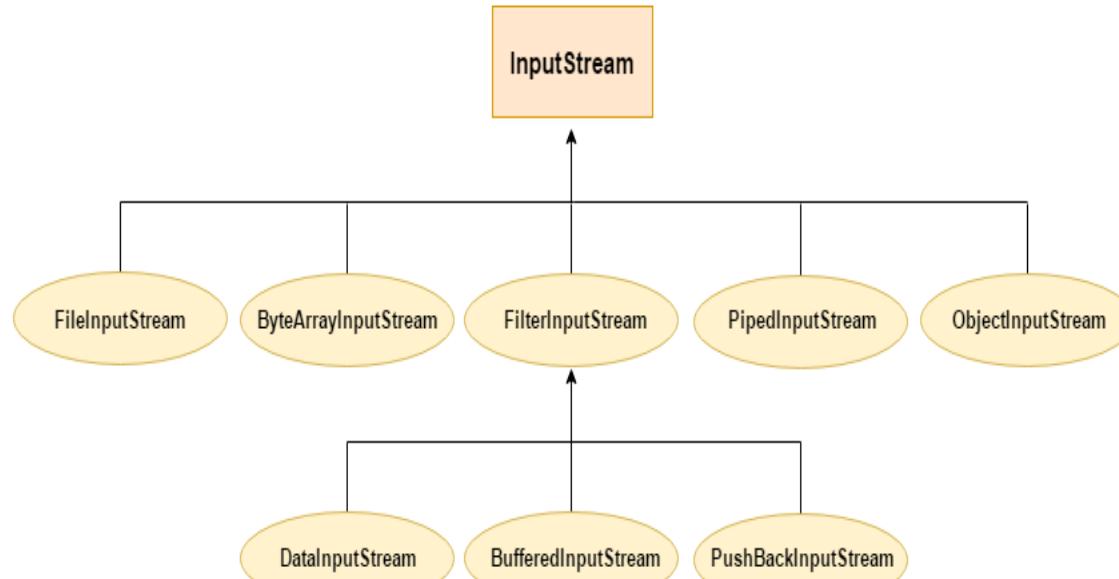
A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

- 1) `System.out`: standard output stream
- 2) `System.in`: standard input stream
- 3) `System.err`: standard error stream

OutputStream Hierarchy



InputStream Hierarchy



Java I/O is used to process the?

- A. input
- B. output
- C. Both A and B
- D. None of the above

Which of these class is used to read from byte array?

- A. `ArrayInputStream`
- B. `InputStream`
- C. `BufferedInputStream`
- D. `ByteArrayInputStream`

Which of these class is used to read from a file?

- A. **FileInputStream**
- B. InputStream
- C. BufferedInputStream
- D. BufferedFileInputStream

A stream is a sequence of data.In Java a stream is composed of?

- A. Bits
- B. Bytes**
- C. Both A & B
- D. None of the above

Which stream does Java application uses to read data from a source, it may be a file, an array, peripheral device or socket?

- A. **InputStream**
- B. OutputStream
- C. Input/OutputStream
- D. None of the above

_____ returns true if called on a file and returns false when called on a directory.

- A. IsFile()
- B. Isfile()
- C. **isFile()**
- D. isfile()

Which of these is used to perform all input & output operations in Java?

- A. Streams
- B. Variables
- C. classes
- D. Methods

Which of these is a type of stream in Java?

- A. Integer stream
- B. Short stream
- C. Byte stream
- D. Long stream

Which of these classes are used by Byte streams for input and output operation?

- A. **InputStream**
- B. **InputOutputStream**
- C. Reader
- D. All of the mentioned

Byte stream uses **InputStream** and **OutputStream** classes for input and output operation.

Which of these classes are used by character streams for input and output operations?

- A. **InputStream**
- B. **Writer**
- C. **ReadStream**
- D. **InputOutputStream**

Which of these class contains the methods print() & println()?

- A. System
- B. System.out
- C. BufferedOutputStream
- D. PrintStream

Which of these methods can be used to writing console output?

- A. print()
- B. println()
- C. write()
- D. All of the mentioned

Which of these class is used to read characters in a file?

- a) FileReader
- b) FileWriter
- c) FileInputStream
- d) InputStreamReader

1. **System** class is defined in

- A. java.util package
- B. java.lang package
- C. java.io package
- D. java.awt package
- E. None of these

In java, how many streams are created for us automatically?

- A. 2
- B. 3
- C. 4
- D. 5

Which method is used to **write a byte** to the current output stream?

- A. **public void write(int) throws IOException**
- B. public void write(byte[]) throws IOException
- C. public void flush() throws IOException
- D. public void close() throws IOException

Which method is used to **write an array of byte** to the current output stream?

- A. public void write(int) throws IOException
- B. **public void write(byte[]) throws IOException**
- C. public void flush() throws IOException
- D. public void close() throws IOException

Which of these class is used to read from byte array?

- A. InputStream.
- B. BufferedInputStream.
- C. ArrayInputStream.
- D. **ByteArrayInputStream**

Which exception is thrown by read() method?

- A. **IOException**
- B. InterruptedException
- C. SystemException
- D. SystemInputException

Which of these is used to read a string from the input stream?

- A. get()
- B. getLine()
- C. read()
- D. readLine()

Which of these class is used to read characters and strings in Java from console?

- A. **BufferedReader**
- B. StringReader
- C. BufferedReader
- D. InputStreamReader

Collection

- The **Collection** in Java is a framework that provides an architecture to store and manipulate the group of objects.
 - Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes ([ArrayList](#), [Vector](#), [LinkedList](#), [PriorityQueue](#), [HashSet](#), [LinkedHashSet](#), [TreeSet](#)).
-
- A Collection represents a single unit of objects, i.e., a group.

What is a framework in Java

It provides readymade architecture.

It represents a set of classes and interfaces.

It is optional.

What is Collection framework

The Collection framework represents a unified architecture for storing and manipulating a group of objects. It has:

Interfaces and its implementations, i.e., classes

Algorithm

Which of these is not a interface in the Collections Framework?

- A. Set
- B. List
- C. Group
- D. Collection

Which of these packages contain all the collection classes?

- A. java.net
- B. java.awt
- C. java.lang
- D. **java.util**

Which of these classes is not part of Java's collection framework?

- A. Maps
- B. Stack
- C. Array
- D. Queue

Which collection allows indexed access to its elements, but its methods are not synchronized?

- A. Vector
- B. TreeMap
- C. HashSet
- D. ArrayList

Collection _____

- A. inherits the Collections class
- B. **inherits the Iterable interface**
- C. implements the Serializable interface
- D. implements the Traversable interface

Which implementation of Iterator can traverse
a collection back and forth?

- A. Iterator
- B. **ListIterator**
- C. SetIterator
- D. MapIterator

The collection is a _____

- A. framework and interface
- B. framework and class
- C. only interface
- D. only class

List, Set and Queue _____ Collection.

- A. extends
- B. implements
- C. both of the above
- D. none of the above

What is Collection in Java?

- A. A group of objects
- B. A group of interfaces
- C. A group of data types
- D. A group of classes

Which of these methods deletes all the elements from invoking collection?

- A. clear()
- B. reset()
- C. delete()
- D. refresh()

Collection stores only ?

- A. **object type data**
- B. string type data
- C. primitive type data
- D. All of the above

Which type of exception is thrown if you try to add an element to an already full Collection ?

- A. ClassCastException
- B. NullPointerException
- C. **IllegalStateException**
- D. IllegalArgumentException

The Comparable interface contains which called?

- A. compare
- B. **compareTo**
- C. toCompare
- D. compareWith

Which of these interface declares core method that all collections will have?

- A. Set
- B. Comparator
- C. EventListner
- D. **Collection**

Which of these interface handle sequences?

- A. Set
- B. **List**
- C. Comparator
- D. Collection

Which of these is Basic interface that all other interface inherits?

- A. Set
- B. Array
- C. List
- D. **Collection**

Elements of which of the collection can be traversed using Enumeration?

- A. ArrayList
- B. **Vector**
- C. HashSet
- D. TreeMap

Which method is used by the contains() method of a list to search an element?

- A. **equals()**
- B. hashCode()
- C. compareTo()
- D. Both equals() & hashCode()

Which of the following collection is not of Iterable type?

- A. ArrayList
- B. Vector
- C. TreeSet
- D. **HashMap**

Each tree based collection assumes its elements to be of --- type?

- A. Sortable
- B. Comparator
- C. Serializable
- D. **Comparable**

Which among the following Sets maintains insertion order?

- A. HashSet
- B. TreeSet
- C. **LinkedHashSet**
- D. Both B & C

Which of these methods can randomize all elements in a list?

- A. rand()
- B. **shuffle()**
- C. randomize()
- D. ambiguous()

Which is faster and uses less memory?

- A. Iterator
- B. Enumeration
- C. ListIterator
- D. ListEnumeration

The default capacity of a ArrayList is:

- A. 1
- B. 10
- C. 12
- D. 16

Which of these methods can convert an object into a List?

- A. SetList()
- B. CopyList()
- C. **singletonList()**
- D. ConvertList()

Which of these method can be used to increase the capacity of ArrayList object manually?

- A. Capacity()
- B. Capacity()
- C. increasecapacity()
- D. **ensureCapacity()**

The class provides the capacity to read primitive data types from an input stream.

- A. PipelInputStream
- B. **DataInputStream**
- C. pushbackInputStream
- D. BufferedInputStream

Exception handling

Exception Handling

The **Exception Handling in Java** is one of the powerful *mechanism to handle the runtime errors* so that the normal flow of the application can be maintained.

Difference between Checked and Unchecked Exceptions

1) Checked Exception

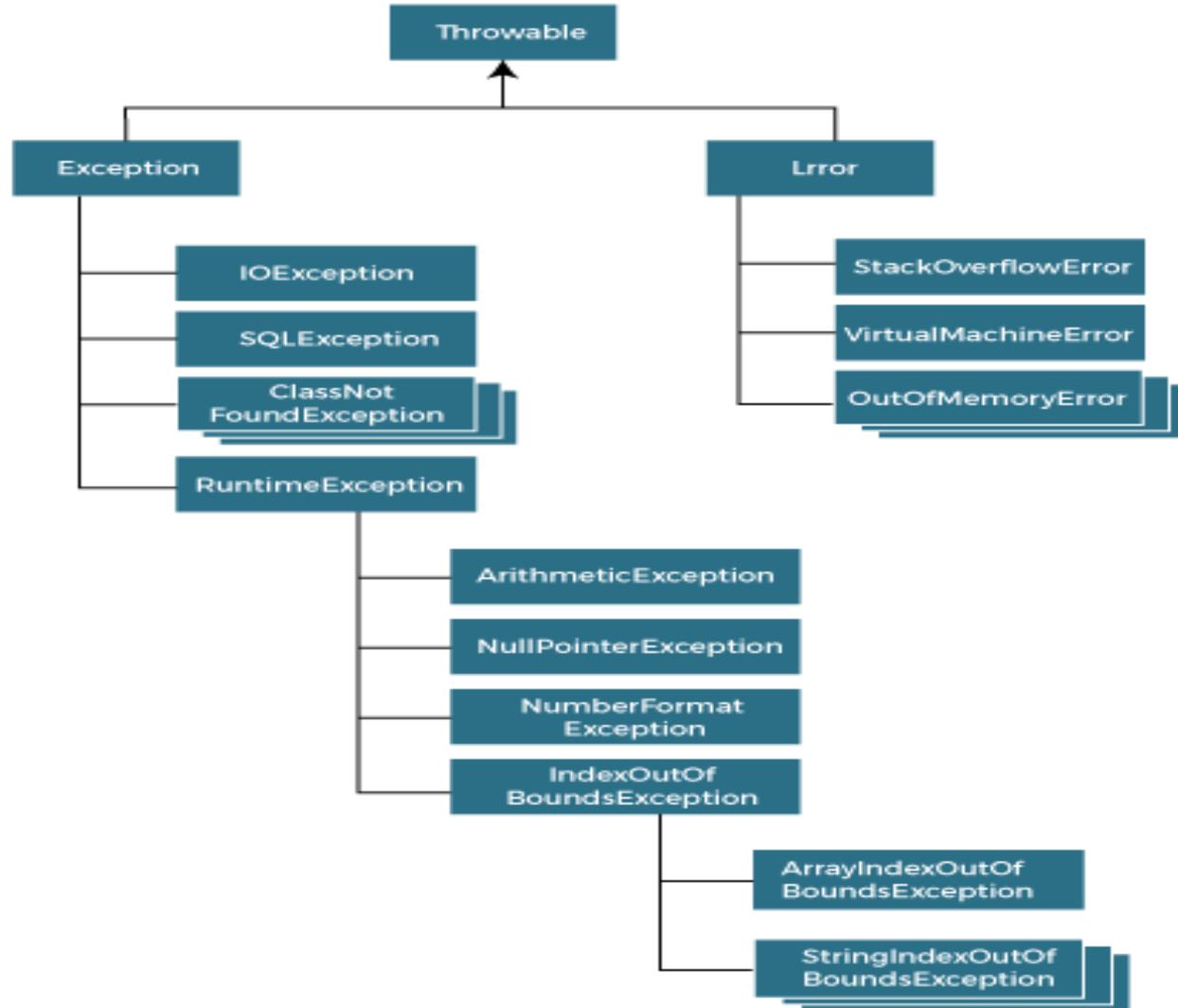
The classes that directly inherit the `Throwable` class except `RuntimeException` and `Error` are known as checked exceptions. For example, `IOException`, `SQLException`, etc. Checked exceptions are checked at compile-time.

2) Unchecked Exception

The classes that inherit the `RuntimeException` are known as unchecked exceptions. For example, `ArithmaticException`, `NullPointerException`, `ArrayIndexOutOfBoundsException`, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

3) Error

Error is irrecoverable. Some example of errors are `OutOfMemoryError`, `VirtualMachineError`, `AssertionError` etc.



Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

- At a time only one exception occurs and at a time only one catch block is executed.
- All catch blocks must be ordered from most specific to most general, i.e. catch for ArithmeticException must come before catch for Exception.

Rule: For each try block there can be zero or more catch blocks, but only one finally block.

Throw

Java throw keyword is used to explicitly throw an exception.

Checked exception cannot be propagated using throw only.

Throw is followed by an instance.

Throw is used within the method.

You cannot throw multiple exceptions.

Throws

Java throws keyword is used to declare an exception.

Checked exception can be propagated with throws.

Throws is followed by class.

Throws is used with the method signature.

You can declare multiple exceptions

Exception is a class/interface/abstract class/other?

- A. Class
- B. Interface
- C. Abstract class
- D. Other

Exception is found in which package in java

- A. java
- B. java.io
- C. java.util
- D. **java.lang**

When does Exceptions in Java arises in code sequence?

- A. Run Time
- B. Compilation Time
- C. Can Occur Any Time
- D. None of the mentioned

Which of these keywords is not a part of exception handling?

- A. try
- B. finally
- C. thrown
- D. catch

Which of these keywords must be used to monitor for exceptions?

- A. try
- B. catch
- C. finally
- D. throw

Which exception is thrown when divide by zero statement executes?

- A. **ArithmeticException**
- B. NullPointerException
- C. NumberFormatException
- D. None of these

What exception thrown by parseInt() method?

- A. NullPointerException
- B. ArithmeticException
- C. ClassNotFoundException
- D. **NumberFormatException**

Which of these exceptions will occur if we try to access the index of an array beyond its length?

- A. ArrayException
- B. ArithmeticException
- C. ArrayIndexException
- D. **ArrayIndexOutOfBoundsException**

What happen in case of multiple catch blocks?

- A. The superclass exception cannot caught first.
- B. The superclass exception must be caught first.
- C. Either super or subclass can be caught first.
- D. None of these

In which condition will the finally block will not be executed?

- A. When some Error occurs
- B. When Exception is raised
- C. When System.Exit(1) is called
- D. In all the cases

Which of these methods return description of an exception?

- A. getException()
- B. **getMessage()**
- C. obtainDescription()
- D. obtainException()

Which of these methods is used to print stack trace?

- A. getStackTrace()
- B. displayStackTrace()
- C. obtainStackTrace()
- D. **printStackTrace()**

The class at the top of exception class hierarchy is

- A. Object
- B. **Throwable**
- C. Exception
- D. ArithmeticException

In which of the following package Exception class exist?

- A. java.io
- B. **java.lang**
- C. java.util
- D. java.file

Exception generated in try block is caught in block.

- A. throw
- B. throws
- C. **catch**
- D. finally

Which keyword is used to explicitly throw an exception?

- A. try
- B. catch
- C. **throw**
- D. throwing

Wrapper classes in Java

The wrapper class in Java provides the mechanism to convert primitive into object and object into primitive.

autoboxing and unboxing feature convert primitives into objects and objects into primitives automatically. The automatic conversion of primitive into an object is known as autoboxing and vice-versa unboxing.

Primitive Type	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Autoboxing

The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing, for example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short.

Unboxing

The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing. It is the reverse process of autoboxing.

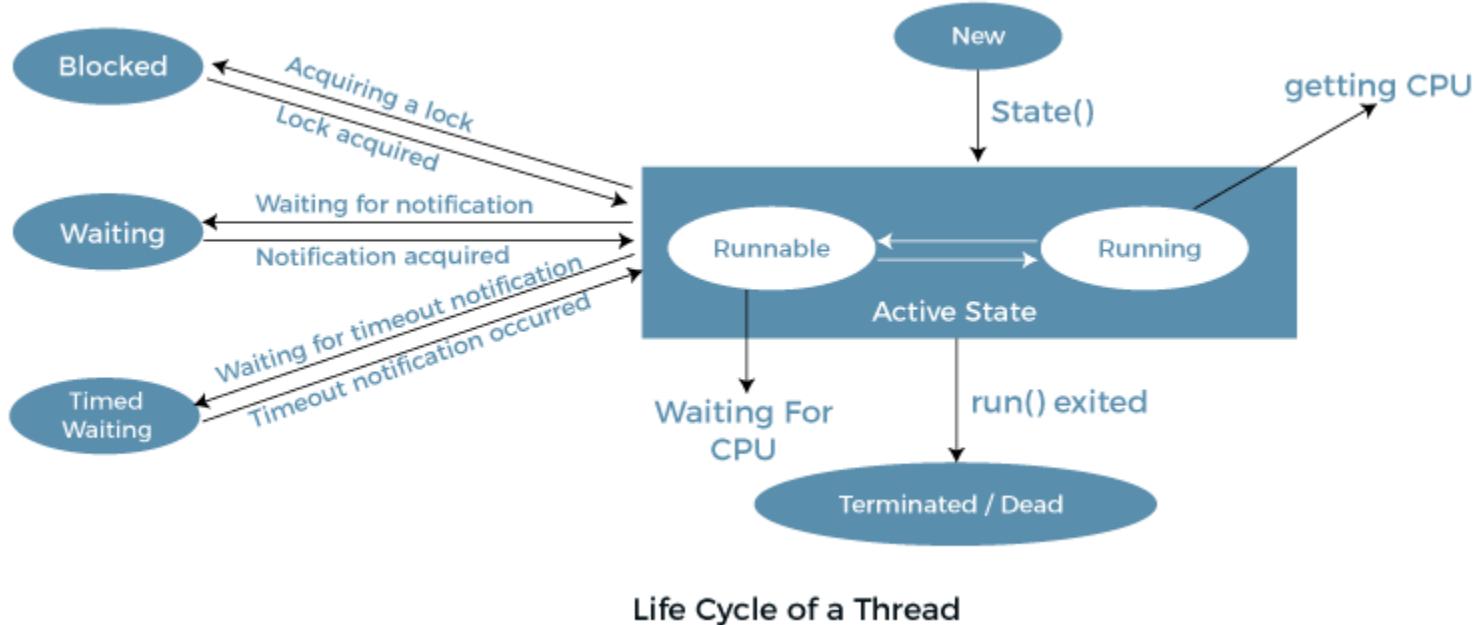
Multi-threading

Multithreading in Java

- Multithreading in Java is a process of executing multiple threads simultaneously.
- A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

Advantages of Java Multithreading

- 1) It doesn't block the user because threads are independent and you can perform multiple operations at the same time.
- 2) You can perform many operations together, so it saves time.
- 3) Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.



How to create a thread in Java

There are two ways to create a thread:

- By extending Thread class
- By implementing Runnable interface.

Thread class:

Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

Commonly used Constructors of Thread class:

Thread()

Thread(String name)

Thread(Runnable r)

Thread(Runnable r, String name)

```
class Multi extends Thread
{
public void run()
{
System.out.println("thread is running...");
}
public static void main(String args[])
{
Multi t1=new Multi();
t1.start();
}
}
```

```
class Multi3 implements Runnable
{
public void run()
{
System.out.println("thread is running...");
}

public static void main(String args[])
{
Multi3 m1=new Multi3();
Thread t1 =new Thread(m1); // Using the constructor Thread(Runnable r)
t1.start();
}
}
```


Commonly used methods of Thread class:

public void run(): is used to perform action for a thread.

public void start(): starts the execution of the thread. JVM calls the run() method on the thread.

public void sleep(long miliseconds): Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

public void join(): waits for a thread to die.

public void join(long miliseconds): waits for a thread to die for the specified milliseconds.

public int getPriority(): returns the priority of the thread.

public int setPriority(int priority): changes the priority of the thread.

public String getName(): returns the name of the thread.

public void setName(String name): changes the name of the thread.

public Thread currentThread(): returns the reference of currently executing thread.

public int getId(): returns the id of the thread.

public Thread.State getState(): returns the state of the thread.

public boolean isAlive(): tests if the thread is alive.

public void yield(): causes the currently executing thread object to temporarily pause and allow other threads to execute.

public void suspend(): is used to suspend the thread(deprecated).

public void resume(): is used to resume the suspended thread(deprecated).

public void stop(): is used to stop the thread(deprecated).

public boolean isDaemon(): tests if the thread is a daemon thread.

public void setDaemon(boolean b): marks the thread as daemon or user thread.

public void interrupt(): interrupts the thread.

public boolean isInterrupted(): tests if the thread has been interrupted.

public static boolean interrupted(): tests if the current thread has been interrupted.

What is multithreaded programming?

- A. It's a process in which two different processes run simultaneously
- B. It's a process in which two or more parts of same process run simultaneously
- C. It's a process in which many different process are able to access same information
- D. It's a process in which a single process can access information from many sources

Threads are

- A. **lightweight process**
- B. heavyweight process
- C. both
- D. none

A process can have

- A. only one thread
- B. **one or multiple thread**
- C. multiple sub-threads
- D. multiple sub-threads & thread

What is the name of the method used to start a thread execution?

- A. resume();
- B. run();
- C. start();**
- D. init();

What is the valid range of priority of a thread in Java multi-threading?

- A. 1 to 10
- B. 0 to 10
- C. 0 to 9
- D. 1 to 9

Which is the minimum priority of the Thread?

- A. Thread.LOW_PRIORITY
- B. Thread.MIN_PRIORITY
- C. Thread.NORM_PRIORITY
- D. Thread.MINIMUM_PRIORITY

What is the value of Thread.NORM_PRIORITY?

- A. 1
- B. 5
- C. 6
- D. 9

What is the default priority of the main thread?

- A. 1
- B. 5
- C. 10
- D. Random value within 1 to 10

Which of these are types of multitasking?

- A. Process based
- B. Thread based
- C. **Process and Thread based**
- D. None of the mentioned

Thread priority in Java is?

- A. **integer**
- B. float
- C. double
- D. long

We can create thread in java by

- A. implementing Thread
- B. **extending Thread**
- C. extending Runnable
- D. both b & c

If there occurs any exception in thread, then other threads

- A. stop executing
- B. gets impacted
- C. **doesn't gets impacted**
- D. daemon thread starts executing

The tasks or job that thread needs to perform
is written inside

- A. static block
- B. inner class
- C. Both A & B
- D. run()

Which of these method waits for the thread to terminate?

- A. join()
- B. stop()
- C. sleep()
- D. isAlive()

Which of these method is used to explicitly set the priority of a thread?

- A. set()
- B. make()
- C. **setPriority()**
- D. makePriority()

Which of the following method is not used to suspend the execution of a thread?

- A. sleep()
- B. wait()
- C. **yield()**
- D. join()

Explanation:- yield() method does not suspend the thread. It let the thread surrenders the execution control i.e. moves the thread to Ready to Run state from the Running state.

What will happen if two threads of the same priority are called to be processed simultaneously?

- A. Anyone will be executed first lexicographically
- B. Both of them will be executed simultaneously
- C. None of them will be executed
- D. It is dependent on the operating system

Which cannot directly cause a thread to stop executing?

- A. Calling notify() method on an object
- B. Calling the wait() method on an object
- C. Calling read() method on an InputStream object
- D. Calling the SetPriority() method on a Thread object

What will be the output of the following Java code?

```
class multithreaded_programing
{
    public static void main(String args[])
    {
        Thread t = Thread.currentThread();
        System.out.println(t);
    }
}
```

- A. Thread[main,0]
- B. Thread[main,5]
- C. Thread[5,main]
- D. Thread[main,5,main]

Which function of pre defined class Thread is used to check weather current thread being checked is still running?

- A. Join()
- B. Alive()
- C. **isAlive()**
- D. isRunning()

Which of these method is used to tell the calling thread to give up a monitor and go to sleep until some other thread enters the same monitor?

- A. **wait()**
- B. notify()
- C. notifyAll()
- D. sleep()

A Daemon thread is a ---?

- A. Controller thread
- B. Dangerous thread
- C. Low priority thread
- D. Assistant thread

What is true about threading?

- A. run() method creates new thread
- B. run() method calls start() method and runs the code
- C. run() method can be called directly without start() method being called
- D. start() method creates new thread and calls code written in run() method

Which method must be defined by a class implementing the java.lang.Runnable interface?

- A. void run()
- B. public void run()**
- C. public void start()
- D. void run(int priority)

Which of these keywords are used to implement synchronization?

- A. syn
- B. synch
- C. synchronize
- D. synchronized**

Which statement is true?

- A. The `notify()` method is defined in class `java.lang.Thread`.
- B. To call `wait()`, an object must own the lock on the thread.
- C. **The `notifyAll()` method must be called from a synchronized context.**
- D. The `notify()` method causes a thread to immediately release its locks.

Java Applet

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

Advantage of Applet

There are many advantages of applet. They are as follows:

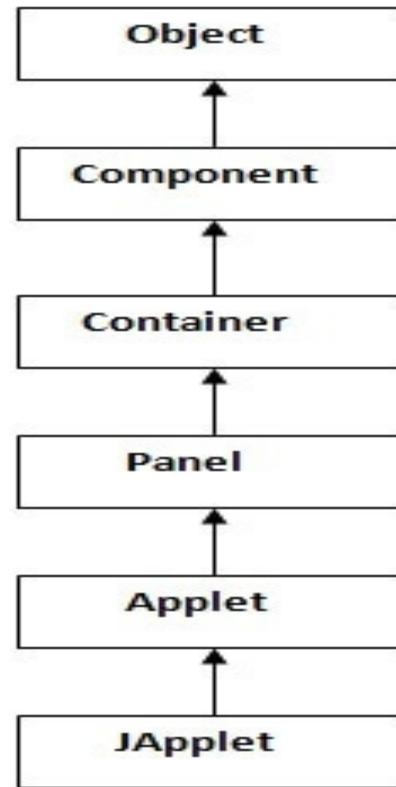
It works at client side so less response time.

Secured

It can be executed by browsers running under many platforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

Plugin is required at client browser to execute applet.



Lifecycle of Java Applet

- Applet is initialized.
- Applet is started.
- Applet is painted.
- Applet is stopped.
- Applet is destroyed.

```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("welcome to applet",150,150);
    }
}
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
```

Commonly used methods of Graphics class:

public abstract void drawString(String str, int x, int y): is used to draw the specified string.

public void drawRect(int x, int y, int width, int height): draws a rectangle with the specified width and height.

public abstract void fillRect(int x, int y, int width, int height): is used to fill rectangle with the default color and specified width and height.

public abstract void drawOval(int x, int y, int width, int height): is used to draw oval with the specified width and height.

public abstract void fillOval(int x, int y, int width, int height): is used to fill oval with the default color and specified width and height.

public abstract void drawLine(int x1, int y1, int x2, int y2): is used to draw line between the points(x1, y1) and (x2, y2).

public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer): is used draw the specified image.

public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle): is used draw a circular or elliptical arc.

public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle): is used to fill a circular or elliptical arc.

public abstract void setColor(Color c): is used to set the graphics current color to the specified color.

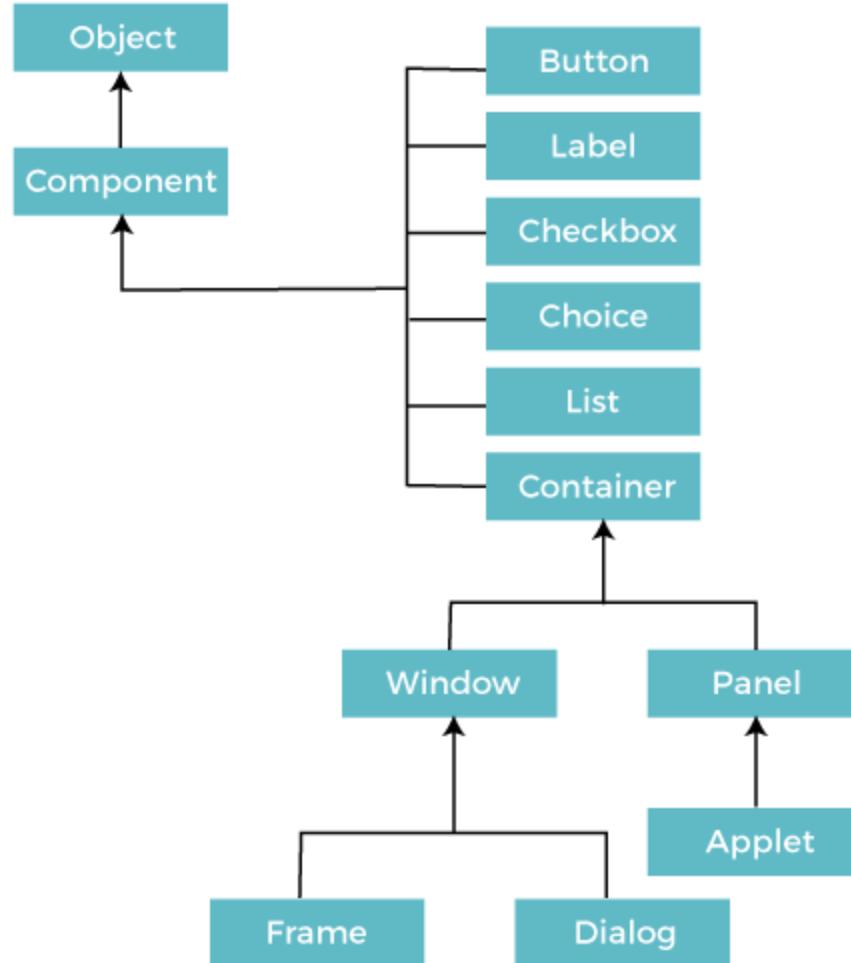
public abstract void setFont(Font font): is used to set the graphics current font to the specified font.

Java AWT

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The `java.awt` package provides classes for AWT API such as `TextField`, `Label`, `TextArea`, `RadioButton`, `CheckBox`, `Choice`, `List` etc.



There are **four types** of containers in Java AWT:

Window

Panel

Frame

Dialog

Event and Listener (Java Event Handling)

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

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

The LayoutManagers are used to arrange components in a particular manner. The Java LayoutManagers facilitates us to control the positioning and size of the components in GUI forms.

java.awt.BorderLayout
java.awtFlowLayout
java.awt.GridLayout
java.awt.CardLayout
java.awt.GridBagLayout
javax.swingBoxLayout
javax.swing.GroupLayout
javax.swing.ScrollPaneLayout
javax.swing.SpringLayout etc.

Which of these functions is called to display the output of an applet?

- A. **paint()**
- B. **display()**
- C. **PrintApplet()**
- D. **displayApplet()**

Which of these methods can be used to output a string in an applet?

- A. **display()**
- B. **print()**
- C. **drawString()**
- D. **transient()**

Which of these methods is a part of Abstract Window Toolkit (AWT) ?

- A. **paint()**
- B. **display()**
- C. **transient()**
- D. **drawString()**

Applets are designed to be embedded within an _____.

- A. SQL
- B. **HTML**
- C. Css
- D. Javascript

What will be output for the following code?

```
import java.awt.*;
import java.applet.*;
public class myapplet extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("A Simple Applet", 20, 20);
    }
}
```

- A. A Simple Applet
- B. Compilation Error
- C. A Simple Applet 20 20
- D. Runtime Error

1. Number of primitive data types in Java are?

- 6
- 7
- 8
- 9

2. What is the size of float and double in java?

- 32 and 64
- 32 and 32
- 64 and 64
- 64 and 32

3. Automatic type conversion is possible in which of the possible cases?

- Byte to int
- Int to long
- Long to int
- Short to int

4. Find the output of the following code.

```
int Integer = 24;  
char String  = 'I';  
System.out.print(Integer);  
System.out.print(string);
```

- Compile error
- Throws exception
- I
- 24 I

9. When an array is passed to a method, what does the method receive?

The reference of the array

A copy of the array

Length of the array

Copy of first element

10. Select the valid statement to declare and initialize an array.

int[] A = {}

int[] A = {1, 2, 3}

int[] A = (1, 2, 3)

int[][] A = {1,2,3}

12. Arrays in java are-

Object references

objects

Primitive data type

None

13. When is the object created with new keyword?

At run time

At compile time

Depends on the code

None

14. Identify the corrected definition of a package.

- A package is a collection of editing tools
- A package is a collection of classes
- A package is a collection of classes and interfaces
- A package is a collection of interfaces

15. Identify the correct restriction on static methods.

1. They must access only static data
2. They can only call other static methods.
3. They cannot refer to this or super.

- I and II
- II and III
- Only III
- I, II and III

16. Identify the keyword among the following that makes a variable belong to a class,rather than being defined for each instance of the class.

final

static

volatile

abstract

17. Identify what can directly access and change the value of the variable res.

```
Package commypackage;
Public class Solution{
    Private int res = 100;
}
```

Any class

Only Solution class

Any class that extends Solution

None

18. In which of the following is `toString()` method defined?

- `java.lang.Object`
- `java.lang.String`
- `java.lang.util`
- None

19. `compareTo()` returns

- True
- False
- An int value
- None

20. Identify the output of the following program.

```
String str = "abcde";
System.out.println(str.substring(1, 3));
```

 abc bc bcd cd

22. Identify the output of the following program.

```
Public class Test{
    Public static void main(String argos[]){
        String str1 = "one";
        String str2 = "two";
        System.out.println(str1.concat(str2));
    }
}
```

- one
- two
- onetwo
- twoone

24. To which of the following does the class string belong to.



java.lang



java.awt



java.applet



java.string

25. How many objects will be created in the following?

```
String a = new String("Interviewbit");
String b = new String("Interviewbit");
String c = "Interviewbit";
String d = "Interviewbit";
```



2



3



4



None

30. Identify the return type of a method that does not return any value.

int

void

double

None

32. Where does the system stores parameters and local variables whenever a method is invoked?

Heap

Stack

Array

Tree

33. Identify the modifier which cannot be used for constructor.

- public
- protected
- private
- static

34. What is the variables declared in a class for the use of all methods of the class called?

- Object
- Instance variables
- Reference variable
- None

36. When is the finalize() method called?

- Before garbage collection
- Before an object goes out of scope
- Before a variable goes out of scope
- None

42. What is Runnable?

- Abstract class
- Interface
- Class
- Method

43. Exception created by **try** block is caught in which block

- catch
- throw
- final
- none

Hide

Correct Answer

Answer- A) Exception created by **try** block is caught in catch block.

44. Which of the following exception is thrown when divided by zero statement is executed?

- NullPointerException
- NumberFormatException
- ArithmeticException
- None

45. Where is System class defined?

- java.lang.package
 - java.util.package
 - java.io.package
 - None
-

47. Which of the following statements are true about finalize() method?

- It can be called Zero or one times
- It can be called Zero or more times
- It can be called Exactly once
- It can be calledOne or more times

1. JDK stands for ____.

- A. Java development kit
- B. Java deployment kit
- C. JavaScript deployment kit
- D. None of these

2. JRE stands for ____.

- A. Java run ecosystem
- B. JDK runtime Environment
- C. Java Runtime Environment
- D. None of these

3. What makes the Java platform independent?

- A. Advanced programming language
- B. It uses bytecode for execution**
- C. Class compilation
- D. All of these

Which of these functions is called to display the output of an applet?

- a) display()
- b) paint()**
- c) displayApplet()
- d) PrintApplet()

Which of this keyword can be used in a subclass to call the constructor of superclass?

- a) super
- b) this
- c) extent
- d) extends

What is the process of defining a method in a subclass having same name & type signature as a method in its superclass?

- a) Method overloading
- b) Method overriding
- c) Method hiding
- d) None of the mentioned

Which of these keywords can be used to prevent Method overriding?

- a) static
- b) constant
- c) protected
- d) final

Which of these is correct way of calling a constructor having no parameters, of superclass A by subclass B?

- a) super(void);
- b) superclass.();
- c) super.A();
- d) super();

String in Java is a _____?

- A. class
- B. object
- C. reference
- D. array of character

Which of these operators can be used to concatenate two or more String objects?

- A. +
- B. +=
- C. &
- D. | |

Which of these constructors is used to create an empty String object?

- A. String(void)
- B. String(0)
- C. String()
- D. None of the mentioned

String is _____ in java.

- A. interface
- B. mutable
- C. method
- D. immutable

Which of these method of class String is used to extract more than one character at a time a String object?

- A. Getchars()
- B. **getChars()**
- C. getchars()
- D. GetChars()

Which of these method of class String is used to compare two String objects for their equality?

- A. **equals()**
- B. Equals()
- C. isequal()
- D. Isequal()

Which of this method of class String is used to obtain a length of String object?

- A. get()
- B. Sizeof()
- C. lengthof()
- D. length()

What will s2 contain after following lines of Java code?

```
String s1 = "one";
String s2 = s1.concat("two")
```

- A. one
- B. two
- C. onetwo
- D. twoone

Which of these method of class String is used to extract a single character from a String object?

- A. ChatAt()
- B. **charAt()**
- C. chatat()
- D. CHARAT()

String in java implements _____ interface(s).

- A. Serializable
- B. CharSequence
- C. Comparable
- D. All of the above**

Which of these methods of class String is used to check whether a given object starts with a particular string literal?

- A. ends()
- B. Starts()
- C. endsWith()
- D. **startsWith()**

Which of the following affirmations are incorrect?

- A. String is a class
- B. **Strings in java are changeable**
- C. Each string is an object of class String
- D. Java defines a fellow class of String, called StringBuffer, which enables string to be modified

Which of these data type value is returned by equals() method of String class?

- A. char
- B. int
- C. boolean
- D. all the above

Which of these method of class String is used to remove leading and trailing whitespaces?

- A. startsWith()
- B. trim()
- C. Trim()
- D. doTrim()

What will be the output of the following Java program?

```
class String_demo
{
    public static void main(String args[])
    {
        char chars[] = {'a', 'b', 'c'};
        String s = new String(chars);
        System.out.println(s);
    }
}
```

- A. a
- B. b
- C. c
- D. abc**

String can be created in java using:

- A. new keyword
- B. String literal
- C. both of the above**
- D. by extending CharSequence

What will be the output of the following Java program?

```
class String_demo
{
    public static void main(String args[])
    {
        char chars[] = {'a', 'b', 'c'};
        String s = new String(chars);
        String s1 = "abcd";
        int len1 = s1.length();
        int len2 = s.length();
        System.out.println(len1 + " " + len2);
    }
}
```

- A. 3 0
- B. 0 3
- C. 3 4
- D. 4 3

What will be the output of the following Java code?

```
class output
{
    public static void main(String args[])
    {
        String c = "Hello i love java";
        boolean var;
        var = c.startsWith("hello");
        System.out.println(var);
    }
}
```

- A. 0
- B. 1
- C. True
- D. False

What will be the output of the following Java code?

```
class output
{
    public static void main(String args[])
    {
        String s1 = "Hello";
        String s2 = new String(s1);
        String s3 = "HELLO";
        System.out.println(s1.equals(s2) + " " + s2.equals(s3));
    }
}
```

- A. true true
- B. false false
- C. true false
- D. false true

The output of the following fraction of code is

```
public class Test
{
    public static void main(String args[])
    {
        String s1 = new String("Hello");
        String s2 = new String("Hellow");
        System.out.println(s1 = s2);
    }
}
```

- A. Hello
- B. Hellow
- C. Compilation error
- D. Throws an exception