

Index

- **Introduction of Java**
- **Data Types**
- **Static Members**
- **String Handling**
- **Array**
- **Classes and Objects**
- **Constructor and garbage Collection**
- **Inheritance**
- **Interface**
- **Exception Handling**
- **Multithreading**
- **Wrapper Classes**
- **File Handling**
- **Applet Programming**

• Introduction of Java

Java is a **programming language** and a **platform**.

Java is a high level, secured and object-oriented programming language.

History of Java

Java history is interesting to know. The history of java starts from Green Team. Java team members (also known as **Green Team**).

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) Originally designed for small, embedded systems in electronic appliances like set-top boxes.

3) Firstly, it was called "**Greentalk**" by James Gosling.

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

In 1995, Oak was renamed as "**Java**" because it was already a trademark by Oak Technologies.

Originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

JDK 1.0 released in(January 23, 1996).

Features of Java

There is given many features of java. They are also known as java buzzwords. The Java Features given below are simple and easy to understand.

1. Simple
2. Object-Oriented
3. Platform independent
4. Secured
5. Robust

6. Architecture neutral
7. Portable
8. High Performance
9. Multithreaded
10. Distributed

Where it is used?

According to Sun, 3 billion devices run java. There are many devices where java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus etc.
2. Web Applications such as irctc.co.in, javatpoint.com etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games etc.

• Variable

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

```
int data=50;//Here data is variable
```

Types of Variable

There are three types of variables in java:

- local variable
- instance variable
- static variable

1) Local Variable

A variable which is declared inside the method is called local variable.

2) Instance Variable

A variable which is declared inside the class but outside the method, is called instance variable . It is not declared as static.

3) Static variable

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

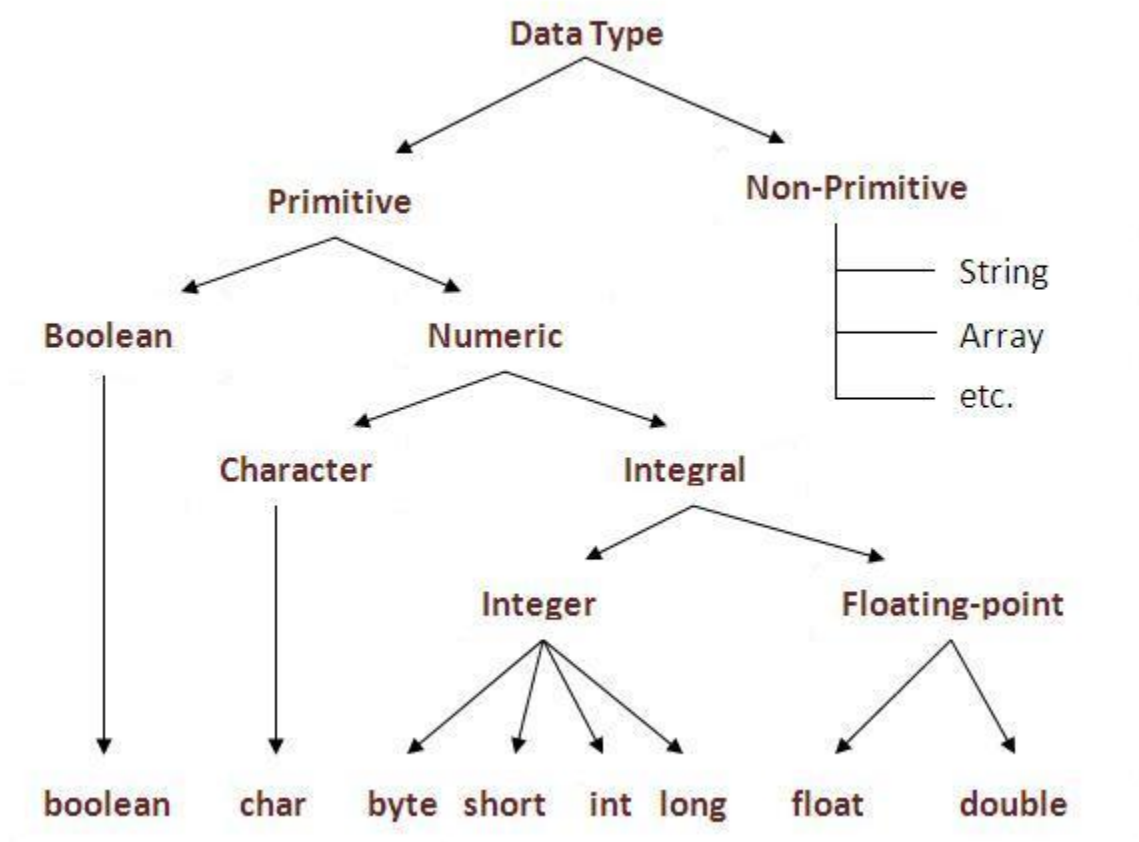
We will have detailed learning of these variables in next chapters.

```
class A{  
int data=50;//instance variable  
static int m=100;//static variable  
void method(){  
int n=90;//local variable  
}  
}//end of class
```

• Data Types in Java

Data types represent the different values to be stored in the variable. In java, there are two types of data types:

- Primitive data types
- Non-primitive data types



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

• Operators in java

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

There are many types of operators in java which are given below:

- Unary Operator,
- Arithmetic Operator,
- shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

Operators	Precedence
Unary	<code>++expr --expr +expr -expr ~ !</code>
Multiplicative	<code>* / %</code>
Additive	<code>+ -</code>

Shift	<< >> >>>
Relational	< > <= >=
Equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	
logical AND	&&
logical OR	
Ternary	? :
Assignment	= += -= *= /= %= &= ^= = <<= >>= >>>=

Java Unary Operator Example: ++ and --

```

class OperatorExample{
public static void main(String args[]){
int x=10;
System.out.println(x++); //10 (11)
System.out.println(++x); //12
System.out.println(x--); //12 (11)
System.out.println(--x); //10
}}

```

• Java If-else Statement

The Java *if statement* is used to test the condition. It checks boolean condition: *true* or *false*. There are various types of if statement in java.

- if statement
- if-else statement
- if-else-if ladder

Java IF Statement

The Java if statement tests the condition. It executes the *if block* if condition is true.

Syntax:

```
if(condition){  
    //code to be executed }  
}
```

Java IF-else Statement

The Java if-else statement also tests the condition. It executes the *if block* if condition is true otherwise *else block* is executed.

Syntax

```
if(condition){ //code if condition is true  
}  
else{  
    //code if condition is false }  
}
```

Java IF-else-if ladder Statement

The if-else-if ladder statement executes one condition from multiple statements.

Syntax:

```
if(condition1){ //code to be executed if condition1 is true  
}  
else if(condition2){  
    //code to be executed if condition2 is true }else {  
    if all the conditions are false }  
}
```


• Java Switch Statement

The Java *switch statement* executes one statement from multiple conditions. It is like if-else-if ladder statement.

Syntax:

```
switch(expression){  
case value1:  
    //code to be executed;  
    break; //optional  
case value2:  
    //code to be executed;  
    break; //optional  
.....  
default:  
    code to be executed if all cases are not matched;  
}
```

• Java For Loop

The Java *for loop* is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.

Java Simple For Loop

The simple for loop is same as C/C++. We can initialize variable, check condition and increment/decrement value.

Syntax:

```
for(initialization;condition;incr/decr){  
    //code to be executed  
}
```

• Java Array

Normally, array is a collection of similar type of elements that have contiguous memory location.

Java array is an object that contains elements of similar data type. It is a data structure where we store similar elements. We can store only a fixed set of elements in a java array.

Array in java is index based, first element of the array is stored at 0 index.

We can store only a fixed size of elements in the array.

Types of Array in java

There are two types of array.

- Single Dimensional Array
- Multidimensional Array

Syntax to Declare an Array in java

```
dataType arr[];
```

We can declare, instantiate and initialize the java array together by :

```
int a[]={33,3,4,5};//declaration, instantiation and initialization
```

Multidimensional array in java

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

Syntax to Declare Multidimensional Array in java

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

```
dataType arrayRefVar[][];
```

Example to instantiate Multidimensional Array in java

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

• Java static keyword

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

The static can be:

1. variable (also known as class variable)
2. method (also known as class method)
3. block

Java static variable

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

- The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees, college name of students etc.
- The static variable gets memory only once in class area at the time of class loading.

Examples

```
Class Example1{  
  
static int a=10;  
  
Public static void main(String args[]){  
  
System.out.println("Value= "+a);  
  
}}
```

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 object of a class.

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

Restrictions for static method

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

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

Java static block

- Is used to initialize the static data member.
- It is executed before main method at the time of classloading.

```
Class Example2{  
  
static  
  
{  
  
System.out.println("I am in static Block");  
  
}  
  
Public static void main(String args[]){  
  
System.out.println("Main Block");  
  
}}
```

• Object and Class in Java

In this Chapter, we will learn about java objects and classes. In object-oriented programming technique, we design a program using objects and classes.

Object in Java

An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical.

For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior.

Class in Java

Class is a standard inbuilt keyword of java language, it create the block that encapsulate the members variables, methods, constructors, blocks, etc.

Syntax to declare a class:

```
class <class_name>{  
    field;  
    variable;  
    method;  
}
```

The types of object as given belows.

Memory allocated object (creation of single object)

Whenever any object is create by new operator then that object will allocate memory and allocated memory will contain member variables according to the number of variables available in class, such allocate memory is known as instance. This concept is known as creation of object.

Multiple objects

Whenever any object is create by new operator then that object will allocate memory and allocated memory will contain member variables according to the number of variables available in class, such allocate memory is known as instance.

We can also create multiple objects of one class and store information in it through reference variable

• Constructor in Java

Constructor in java is a special type of method that is used to initialize the object.

Java constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object..

Rules for creating java constructor

There are basically two rules defined for the constructor.

1. Constructor name must be same as its class name
2. Constructor must have no explicit return type
3. Overloading of constructor is allow.

Types of java constructors

There are two types of constructors:

1. Default constructor (no-arg constructor)
2. Parameterized constructor

Java Default Constructor

A constructor that have no parameter is known as default constructor.

Syntax of default constructor:

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

Java parameterized constructor

A constructor that have parameters is known as parameterized constructor.

Why use parameterized constructor?

Parameterized constructor is used to provide different values to the distinct objects.

For-each loop in java

The for-each loop is used to traverse array or collection in java. It is easier to use than simple for loop because we don't need to increment value and use subscript notation.

It works on elements basis not index. It returns element one by one in the defined variable.

Syntax:

```
for(Type var:array){  
    //code to be executed  
}
```

this keyword in java

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.
3. this() can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.

this: to refer current class instance variable

The this keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

this: to invoke current class method

You may invoke the method of the current class by using the this keyword. If you

don't use the `this` keyword, compiler automatically adds `this` keyword while invoking the method. Let's see the example

this() : to invoke current class constructor

The `this()` constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.

Inheritance in Java

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

Extending the properties of one class into another class is known as inheritance.

The idea behind inheritance in java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.

Why use inheritance in java

- For Method Overriding.
- For Code Reusability.

Syntax of Java Inheritance

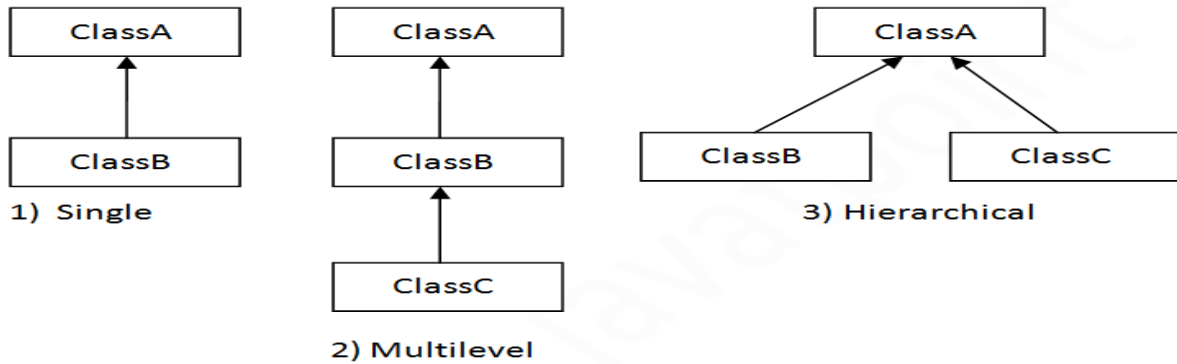
```
class Subclass-name extends Superclass-name
{
    //methods and fields
}
```

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called parent or super class and the new class is called child or subclass.

Types of inheritance in java

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.



super keyword in java

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

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of java super Keyword

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

1) super is used to refer immediate parent class instance variable.

We can use super keyword to access the data member or field of parent class. It is used if parent class and child class have same fields.

2) super can be used to invoke parent class method

The super keyword can also be used to invoke parent class method. It should be used if subclass contains the same method as parent class. In other words, it is used if method is overridden.

3) super is used to invoke parent class constructor.

The super keyword can also be used to invoke the parent class constructor. Let's see a simple example:

Final Keyword In Java

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

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 blank final variable or uninitialized final variable. Let's first learn the basics of final keyword.

1) Java final variable

If you make any variable as final, you cannot change the value of final variable(It will be constant).

2) Java final method

If you make any method as final, you cannot override it.

3) Java final class

If you make any class as final, you cannot extend it.

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only important things to the user and hides the internal details for example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Abstract Keyword

Abstract is a standard inbuilt keyword of java language. It provide two different features i.e abstract class and abstract method.

Abstract class in Java

A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods (method with body). It needs to be extended and its method implemented. It cannot be instantiated.

Example abstract class

```
abstract class A{}
```

Abstract method

A method that is declared as abstract and does not have implementation is known as abstract method.

Example abstract method

```
abstract void printStatus();//no body and abstract
```

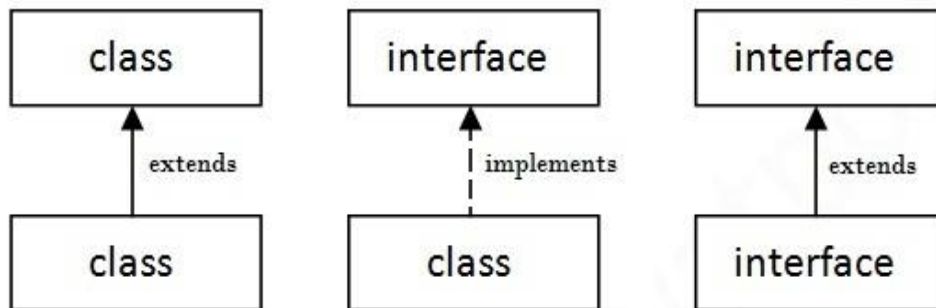
Interface in Java

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

There can be only abstract methods in the java interface not method body. It is used to achieve abstraction and multiple inheritance in Java. It cannot be instantiated just like abstract class.

Understanding relationship between classes and interfaces

As shown in the figure given below, a class extends another class, an interface extends another interface but a **class implements an interface**.



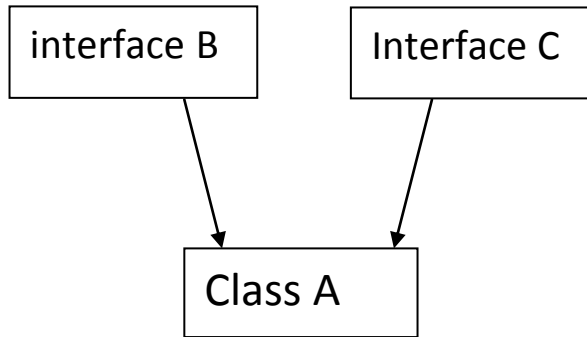
Syntax

class Subclass-name **implements** Interface-name

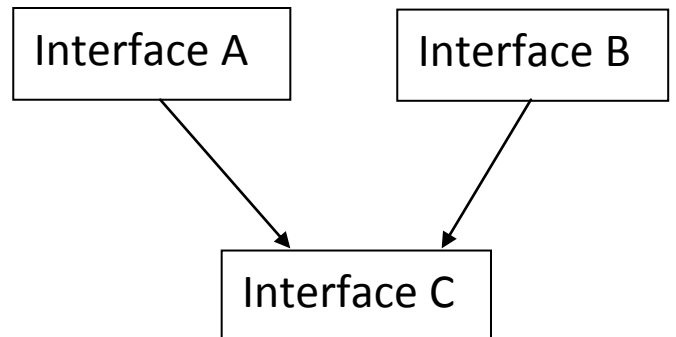
```
{  
    //methods and fields  
}
```

Multiple inheritance in Java by interface

If a class implements multiple interfaces, or an interface extends multiple interfaces i.e. known as multiple inheritance.

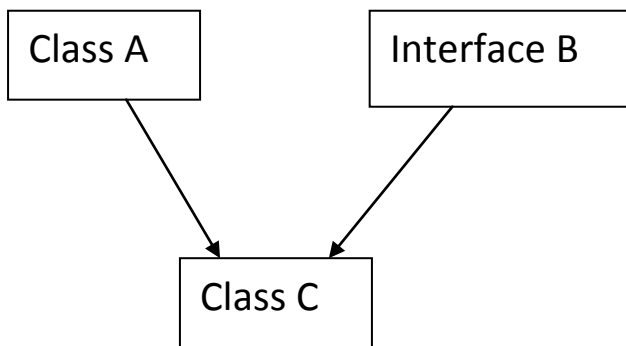


class A impliments B,C



interface C extend A

interface C extend B



Class C extends A impliments B

Java Package

A **java package** is a group of similar types of classes, 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.

Here, we will have the detailed learning of creating and using user-defined 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.

Simple example of java package

The **package keyword** is used to create a package in java.

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

How to compile java package

```
javac -d . Simple.java
```

The -d switch specifies the destination where to put the generated class file. You can use any directory name like /home (in case of Linux), d:/abc (in case of windows) etc. If you want to keep the package within the same directory, you can use . (dot).

How to run java package program

To Run: java mypack.Simple

How to access package from another package?

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

1. `import package.*;`
2. `import package.classname;`
3. fully qualified name.

1) Using `packagename.*`

If you use `package.*` then all the classes and interfaces of this package will be accessible but not subpackages.

The `import` keyword is used to make the classes and interface of another package accessible to the current package.

2) Using `packagename.classname`

If you import `package.classname` then only declared class of this package will be accessible.

3) Using fully qualified name

If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import. But you need to use fully qualified name every time when you are accessing the class or interface.

How to send the class file to another directory or drive?

There is a scenario, I want to put the class file of A.java source file in classes folder of c: drive. For example:

```
package mypack;
```

```
public class Simple{
```

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

To Compile and Run

```
e:\sources> javac -d c:\classes Simple.java
```

```
e:\sources> set classpath=c:\classes;.;
```

```
e:\sources> java mypack.Simple
```

• Multithreading in Java

Multithreading in java is a process of executing multiple threads simultaneously.

Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

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 the threads takes less time than process.

Java Multithreading is mostly used in games, animation etc.

Advantages of Java Multithreading

- 1) It **doesn't block the user** because threads are independent and you can perform multiple operations at same time.
- 2) You **can perform many operations together so it saves time**.

Multitasking

Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved by two ways:

- Process-based Multitasking(Multiprocessing)
- Thread-based Multitasking(Multithreading)

1) Process-based Multitasking (Multiprocessing)

- Each process have its own address in memory i.e. each process allocates separate memory area.
- Process is heavyweight.
- Cost of communication between the process is high.
- Switching from one process to another require some time for saving and loading registers, memory maps, updating lists etc.

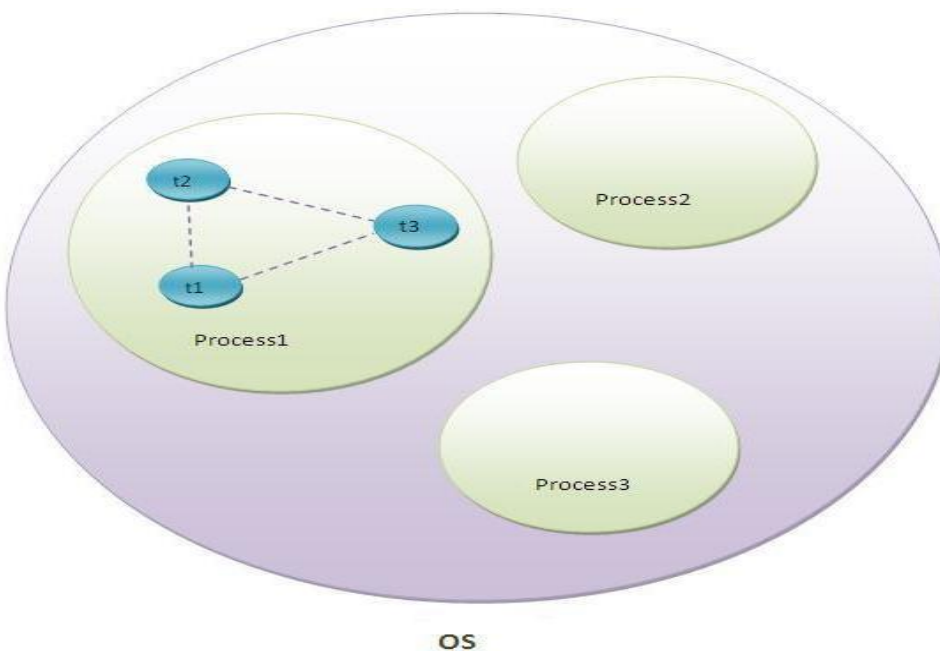
2) Thread-based Multitasking (Multithreading)

- Threads share the same address space.
- Thread is lightweight.
- Cost of communication between the thread is low.

What is Thread in java

A thread is a lightweight sub process, a smallest unit of processing. It is a separate path of execution.

Threads are independent, if there occurs exception in one thread, it doesn't affect other threads. It shares a common memory area.



As shown in the above figure, thread is executed inside the process. There is context-switching between the threads. There can be multiple processes inside the OS and one process can have multiple threads.

How to create thread

There are two ways to create a thread:

1. By extending Thread class
2. 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)

Commonly used methods of Thread class:

1. **public void run():** is used to perform action for a thread.
2. **public void start():** starts the execution of the thread.
3. **public void sleep(long milliseconds):** Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.
4. **public void join():** waits for a thread to die.
5. **public void join(long milliseconds):** waits for a thread to die for the specified milliseconds.
6. **public int getPriority():** returns the priority of the thread.
7. **public int setPriority(int priority):** changes the priority of the thread.
8. **public String getName():** returns the name of the thread.
9. **public void setName(String name):** changes the name of the thread.
10. **public Thread currentThread():** returns the reference of currently executing thread.
11. **public int getId():** returns the id of the thread.
12. **public boolean isAlive():** tests if the thread is alive.
13. **public void yield():** causes the currently executing thread object to temporarily pause and allow other threads to execute.

14. **public void suspend():** is used to suspend the thread(deprecated).
15. **public void resume():** is used to resume the suspended thread(deprecated).
16. **public void stop():** is used to stop the thread(deprecated).
17. **public boolean isDaemon():** tests if the thread is a daemon thread.
18. **public void setDaemon(boolean b):** marks the thread as daemon or user thread.
19. **public void interrupt():** interrupts the thread.
20. **public boolean isInterrupted():** tests if the thread has been interrupted.

Runnable interface:

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. Runnable interface have only one method named run().

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

Starting a thread:

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

- A new thread starts(with new callstack).
- The thread moves from New state to the Runnable state.
- When the thread gets a chance to execute, its target run() method will run.

- **Wrapper class in Java**

Wrapper class in java provides the mechanism *to convert primitive into object and object into primitive*.

Since J2SE 5.0, **autoboxing** and **unboxing** feature converts primitive into object and object into primitive automatically. The automatic conversion of primitive into object is known as autoboxing and vice-versa unboxing.

The eight classes of java.lang package are known as wrapper classes in java. The list of eight wrapper classes are given below:

Primitive Type	Wrapper class
Boolean	Boolean
Char	Character
Byte	Byte
Short	Short
Int	Integer
Long	Long
Float	Float
Double	Double

• Java I/O

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

Java uses the concept of 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.

In java, 3 streams are created for us automatically. All these streams are attached with console.

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream

Let's see the code to print **output and error** message to the console.

```
System.out.println("simple message");
```

```
System.err.println("error message");
```

Let's see the code to get **input** from console.

```
int i=System.in.read();//returns ASCII code of 1st character
```

```
System.out.println((char)i);//will print the character
```

Java FileOutputStream Class

Java FileOutputStream is an output stream used for writing data to a file.

If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to use FileWriter than FileOutputStream.

FileOutputStream class declaration

```
public class FileOutputStream extends OutputStream
```

Java FileInputStream Class

Java `FileInputStream` class obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc. You can also read character-stream data. But, for reading streams of characters, it is recommended to use `FileReader` class.

Java FileInputStream class declaration

```
public class FileInputStream extends InputStream
```

• Java AWT Tutorial

Java AWT (Abstract Window Toolkit) is used to develop GUI or window-based applications in java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

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

• 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.

Lifecycle of Java Applet

1. Applet is initialized.
2. Applet is started.
3. Applet is painted.
4. Applet is stopped.
5. Applet is destroyed.

java.applet.Applet class

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

1. **public void init():** is used to initialize the Applet. It is invoked only once.

2. **public void start():** is invoked after the init() method or browser is maximized. It is used to start the Applet.
3. **public void stop():** is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.
4. **public void destroy():** is used to destroy the Applet. It is invoked only once.

java.awt.Component class

The Component class provides 1 life cycle method of applet.

1. **public void paint(Graphics g):** is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

Simple example of Applet by appletviewer tool:

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by: appletviewer First.java. Now Html file is not required but it is for testing purpose only.

```
//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>
*/
```

To execute the applet by appletviewer tool, write in command prompt:

```
c:\>javac First.java
c:\>appletviewer First.java
```


• Displaying Graphics in Applet

java.awt.Graphics class provides many methods for graphics programming.

Commonly used methods of Graphics class:

1. **public abstract void drawString(String str, int x, int y):** is used to draw the specified string.
 2. **public void drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height.
 3. **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.
 4. **public abstract void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.
 5. **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.
 6. **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).
 7. **public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image.
 8. **public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used draw a circular or elliptical arc.
 9. **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.
 10. **public abstract void setColor(Color c):** is used to set the graphics current color to the specified color.
- **Color is a std class present in java.awt package and –**
 - **java.applet package.**

Format=RGB (Red,Green,Blue)

Each component value varies from 0 to 255.

```
Color c = new Color (255,255,255); //White color.
```