

## **What is the Java?**

---

It is an object oriented programming language similar to C++. This is the open source and can run on all platforms. It is concurrent language you can execute many statements instead of sequential executions also it is class based and object oriented programming language. Java allow us write once run anywhere i.e compiled code can execute on any operating system.

## **Why use the java or feature of java**

---

**Portable:** This is the platform independent language which application written in one platform and can easily portable to another platform.

**Object oriented:** everything is considered to be an object. Means in java we cannot create any program without inheritance and without class so we can say it is pure object oriented.

**Secured:** This code is converted in byte code after compilation which is not readable by a human and java does not use the explicit pointer so user cannot access the memory space and its address directly so it is secured language

**Dynamic:** Java allows us to allocate memory dynamically at run time to which memory wastage is reduced and improve application performance.

**Robust:** java has strong memory management system. It help in eliminating error and it check code during compile and runtime.

**Multithreaded:** java support multiple threads for execution, including a set of synchronization primitives

**Distributed:** this language provide a feature which help to create distributed application using RMI (Remote method invocation) a program can invoke method of another program across network and get the output.

**If we want to work with java we have to know how to create program in it and execute it.**

If we want to create the java program then we have the some following important steps.

**1) Installed the JDK:** before that we have to know what the JDK is

### JDK stands java development kit

**Definition:** It is a software development environment that offers collection of tools and libraries necessary for java application development. JDK contain compiler, JVM, Applet viewer etc

### What is the compiler?

---

Compiler is program or application software which is used for convert the one programming language to another programming. In java case of java it is used for convert the source code to byte code

### Working of compiler

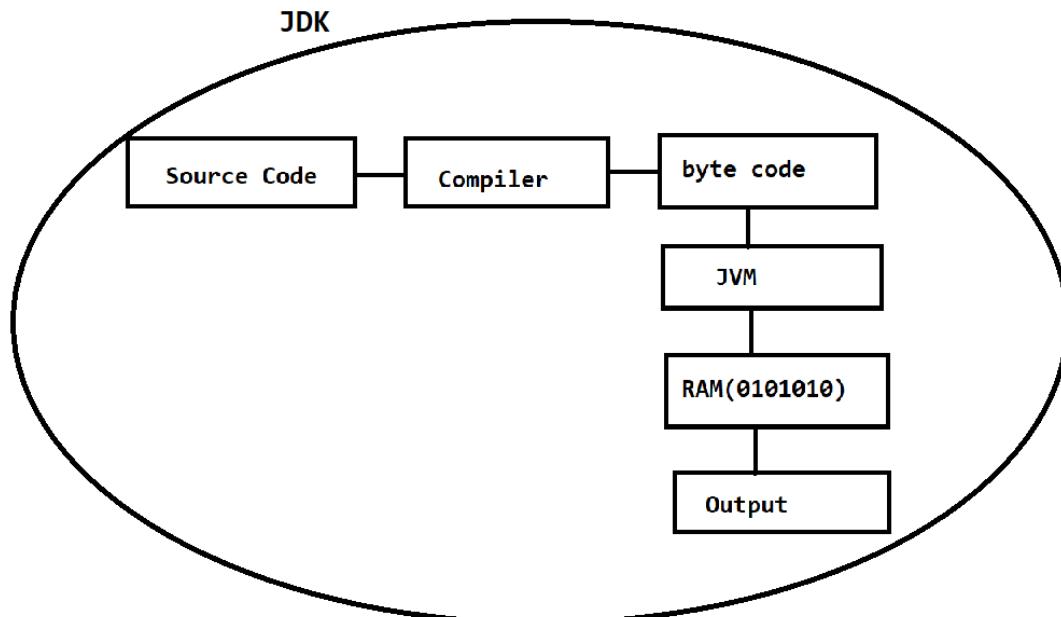
---



### What is the byte code?

---

Byte code is machine understandable code or it is format which easily converts in machine code with the help of JVM.



## What is the JVM?

JVM stands for Java Virtual Machine it is specification that provide runtime environment in which byte code can be executed and translate in machine code

## What are the roles of JVM or what it does?

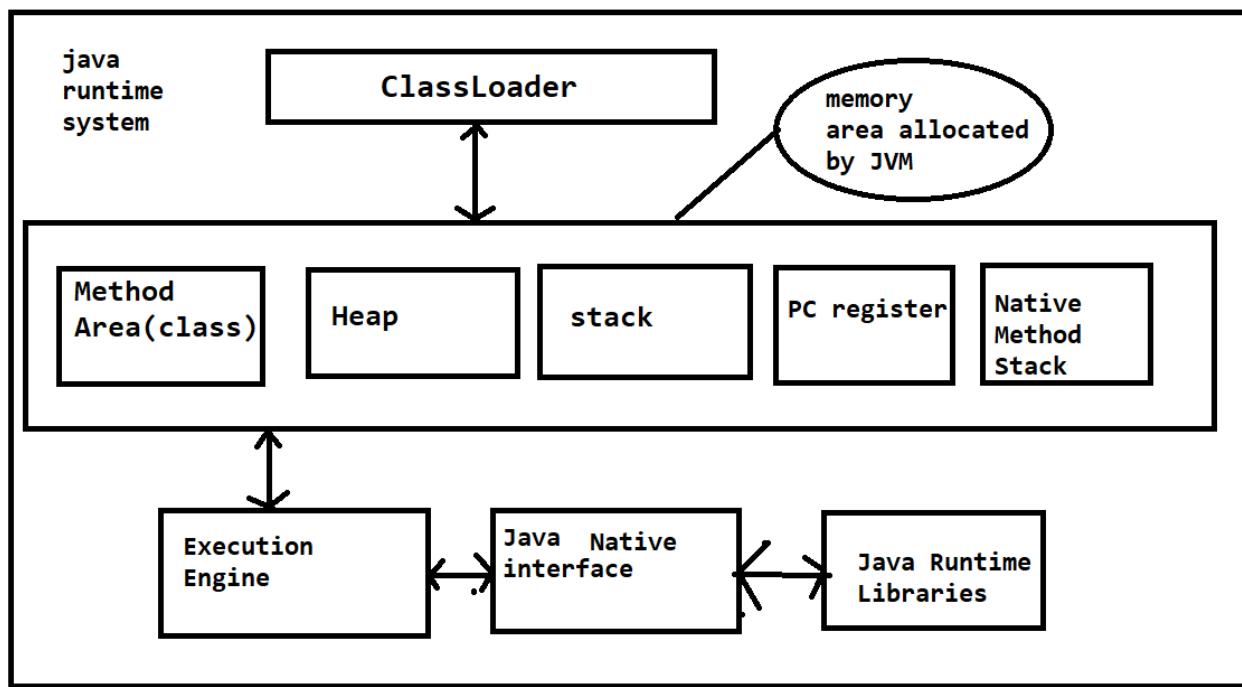
1) Load the byte code 2) verify the byte code 3) execute the byte code 4) provide the runtime environment for execution of code.

## JVM provide the definition for the

1) Memory area 2) class file format 3) register set 4) garbage collector 5) fatal error reporting file

## Explain the JVM Architecture

Following diagram shows the JVM Architecture



**Class Loader:** class loader is subsystem of JVM which load the class file whenever we run the java program as well as verify the byte code. There are major three types of class loader

**1) Bootstrap Class Loader:** this is the first class loader which is super of extension class loader. It loads the rt.jar which contains all class files in java standard edition like as java.lang package, java.net package, java.util package.

**2) Extension Class Loader:** this is the child of Bootstrap Classs Loader which is responsible for load the external library and its .class files in your application

**3) System/Application class loader:** it is used for load the class files from class path by default class path to current directory means jdk\bin folder

**Method Area or Class Area:** In the method area all class level information stored like as class name, immediate parent class name, methods of class, and variables of class including static variables. The only one method for per JVM and it is also shared resource.

**Heap Section:** information of all objects is stored in the heap area. There is also one heap area for per JVM and it is shared resource.

**Stack Area:** for every thread JVM create runtime stack which is stored here every block of this stack is stack frame or active record which store the method call. All local variables of this method are stored corresponding stack frame after thread termination stack frame will be destroyed by JVM it is not shared resource.

**PC Register:** It store the address of current instruction of thread and every thread having separate pc register

**Native Method Stack:** for every thread a separate native stack is created it store native method information

**Execution Engine:** Execution engine execute the .class file (byte code). It read the byte code line by line uses data and information in various memory areas and execute instructions

Execution engine divide in three parts

**a) Interpreter:** read the byte code line by line and then execute it. The disadvantage is that when one method is called multiple time every time interpreter get executed.

**b) JIT (Just in Time):** it is used for increase the efficiency of interpreter. It compile the entire byte code and change it to native code so whenever we call method multiple time no need to interpretation multiple time.

**c) Garbage collector:** it is used for destroy the unused objects.

**JNI (Java Native Interface):** it is an interface with the native method library and provides the native libraries (C, C++) required for execution and it enables JVM to call C/C++ libraries.

### **Why java developed the byte code**

---

Byte code is platform independent code means we can execute the byte code on any operating system so java is platform independent language and it is very strong feature of java

### **2) Create the sample application:**

If we want to create the any program in java we must be write minimum single class. Means we cannot run the java application without class

### **Generalize format of java application**

---

```
access specifier class classname
{ public static void main(String x[])
{
    write here your logics
}
}
```

**Example:**

```
public class First
{
    public static void main(String x[])
    {
        System.out.println("good morning");
    }
}
```

---

#### Description of above code

**public class First:** public is access specifier and class is keyword for class declaration and First is class name and user can give any name to his class.

**public static void main(String x[]):** this is the main method of java same like as main method in c or c++.

**System.out.println():** this is the output statement of java it is used for display the output on output screen like as printf()

**Meaning:** System is class out is static reference of PrintStream class System and PrintStream maintain the HAS-A relationship between them println () is overloaded method for display the output on output screen.

**Note:** we can create the java program by using notepad, eclipse, spring tool suite or interllJ etc

But here we use the notepad.

---

**Example:**

```
public class TenAug
{ public static void main(String x[])
    { System.out.println("Good Evening");
    }
}
```

**3) Save the Application:** if we want to save the application of java then save in bin folder where jdk installed and give class name and file name same with .java extension as per our example we have to give the file name as TenAug.java

**4) Compile the Application:** if we want to compile the application of java we have the some following steps.

**a) Open the command prompt**

Start menu → search → command prompt

**b) Go where java file save:** copy the path where java file save and paste on command prompt using the following command shown in screen short.

```
C:\Users\Admin>cd C:\Program Files\Java\jdk1.8.0_291\bin  
C:\Program Files\Java\jdk1.8.0_291\bin>
```

**c) type the command javac filename.java shown in following screen short.**

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac TenAug.java  
C:\Program Files\Java\jdk1.8.0_291\bin>
```

**Note: when we compile the file then java compiler create the new file automatically with extension of .class file and in this file contain your byte code.**

```
C:\Program Files\Java\jdk1.8.0_291\bin>dir TenAug.*  
Volume in drive C has no label.  
Volume Serial Number is D892-2B3C  
  
Directory of C:\Program Files\Java\jdk1.8.0_291\bin  
byte code  
20-10-2021 18:15      418 TenAug.class  
20-10-2021 18:12      121 TenAug.java  
          2 File(s)   539 bytes  
          0 Dir(s)  80,080,384,000 bytes free  
source code
```

**5) Run the application:** if we want to run the java application then we have the command name as java filename

E.g **java TenAug**

Output shown in following screen short

```
C:\Program Files\Java\jdk1.8.0_291\bin>java TenAug  
Good Evening
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>
```

### **Q. Why main method is static?**

Because Java main method present in class and java main method call by using JVM and JVM need to call the java main method using class name so it is static Suppose java not declare main method as static then JVM need to create the object of the class in which main method exist but there is problem in every class contain by default constructor called as implicit constructor but if user declare the parameterized constructor in its class and when JVM create the object of the class then JVM need to pass parameter to constructor but JVM cannot pass the

parameter to user constructor so java avoid to create the object of class in which main method is present by JVM so it is static because static method can call by using class name (refer java interview video series)

**Q. can we overload the main method in java?**

---

Yes we can overload the main method in java when we overload the main method in java then program compile get but not execute if we not overload the method public static void main(String x[]) because program execution done by using JVM and JVM knows the only one signature of main method called as public static void main(String args[])

```
class A
{
    public static void main(int x[])
    {
    }
}
```

If we write the main method mention as above then program get compile but not execute because compiler think user may be overload the main method but when we execute the java program then JVM is enable and JVM knows only one signature of main method called as public static void main(String x[])

So if we want to execute the java program property then your code should be

```
class A
{
    public static void main(int x[])
    {
    }
    public static void main(String x[])
    {
        System.out.println(" i can overload");
    }
}
```

Command Prompt

```
D:\anup sir>javac A.java

D:\anup sir>java A
i can overload

D:\anup sir>
```

**What is the meaning of string x[] present in main method ?**

String x[] indicate the command line argument in java

Command line argument means parameter pass in main method of string array called as command line arguments.

```
class A
{
    public static void main(String x[])
    {
    }
}
```

command line arguments

## Why use the command line arguments or what is the benefit of command line arguments?

---

It is used for accept the input at program run time and it is infinite string array present in main method means we can accept the n number of input through the command line argument but the first input of command line is at position of zero because it is array.

**Now we want to create application to accept the two numbers from keyboard and calculate its addition**

```
public class A
{
    public static void main(String x[])
    {
        int a,b,c;
        a=x[0]; //accept first input
        b=x[1]; // accept second input
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

If we think about the above code then we have two compile time errors

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac A.java
A.java:6: error: incompatible types: String cannot be converted to int
    a=x[0];
           ^
A.java:7: error: incompatible types: String cannot be converted to int
    b=x[1];
           ^
2 errors

C:\Program Files\Java\jdk1.8.0_291\bin>
```

## Why these errors occur?

---

Because we have the string value for input and we want to accept the integer value as input and string cannot store in integer so compiler generate the error to us in incompatible types.

```
public class A
{
    public static void main(String x[])
    {
        int a,b,c;
        a=x[0];           x[0] is type of string and try to store a variable and
        b=x[1];           a variable is type of integer so we have the
        c=a+b;            error incompatible
        System.out.printf("Addition is %d\n",c);
    }
}
```

If we want to solve this error in java we have to use the type casting technique

### **What is the type casting?**

---

Convert one type value to another type called as type casting.

Means as per our example we have to convert the string to integer

### **How to convert String to integer in java or String to other type**

---

If we want to convert the string to integer in java we have the following statement.

#### **Syntax:**

```
int variable = Integer.parseInt(String);
```

For float conversion we have to write like as

```
Float variable=Float.parseFloat(String);
```

#### **For double conversion**

---

```
Double variable=Double.parseDouble (String);
```

Etc

So if we correct the above code for resolve the above error then your code should be.

```
public class A
{
    public static void main(String x[])
    {
        int a,b,c;
        a=Integer.parseInt(x[0]);
        b=Integer.parseInt(x[1]);
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

we solve error  
successfully

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac A.java
C:\Program Files\Java\jdk1.8.0_291\bin>
```

When we try to run this program then we get the error at program run time shown in following screen short.

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac A.java

C:\Program Files\Java\jdk1.8.0_291\bin>java A
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
at A.main(A.java:6)
```

We get error `ArrayIndexOutOfBoundsException`

Because when we run the program and if we use the command line argument then we have to pass input on same line where we run the program

Following diagram shows the correct way of input in the command line argument

```
public class A
{
    public static void main(String x[])
    {
        int a,b,c;
        a=Integer.parseInt(x[0]);
        b=Integer.parseInt(x[1]);
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

Not found input for  
x[0] so JVM  
raise error  
ArrayIndexOutOfBoundsException  
0

x[0] x[1]

proper input  
as per program

not found  
input for x[1]  
so JVM  
raise error  
ArrayIndexOutOfBoundsException  
1

## What is the JRE?

JRE Stands for Java Runtime Environment it is software package which bundle the libraries (jar) and the java virtual machine and other components to run application written in the java. JRE is physical existence of JVM and JVM is logical entity or abstract machine which is present under the JRE. If we want to execute any java program we have to install the JRE.

### JRE contain the following bundles or packages

- 
1. DLL
  2. Java extension files
  3. Font files
  4. Core libraries

## What is the diff between JDK JRE and JVM?

---

JDK is software development kit whereas JRE is software bundle that allow java program to run where as JVM is an environment for executing byte code.

JDK contain tools for developing, debugging etc JRE contain class libraries and other supporting files whereas software development tool are not included in JVM and JVM work as tool in JRE

JDK comes with the installer on the other hand JRE contain environment to execute the code where JVM is tool present in JDK and JRE

JDK contain compiler, applet viewer etc tools

### **Why use JDK**

JDK contain tools required to write Java Programs and JRE to execute them

JDK include compiler, Java Application launcher and Applet viewer

Means we can use the JDK for develop the java source debug it and convert it in byte code means for compilation purpose.

### **Why Use JRE?**

It is used for import the packages in application like as `java.util,java.math` and run time libraries means JRE specially design for execute and test the java application

### **Why Use JVM**

JVM is platform for executing java source code internally provide the memory for java objects and release the memory of objects and JVM comes with JIT which help to minimize the code interpretation and provide the clean code for converting in machine code.

### **Is JVM is platform dependent or independent**

JVM is platform dependent means every operating system having different JVM as per the operating system.

### **Why JVM is platform dependent**

Because JVM separate for every operating system and every operating system having different machine code generation algorithms and technique so Java design the JVM as per the operating system machine code generation technique so JVM is platform dependent

### **How to Accept the Input on new line using Java**

---

If we want to accept the input on new line using java we have the Scanner class.

#### **Steps to works with Scanner class**

---

##### **1) Add the java.util package in application**

###### **What is the package?**

---

Package is collection of classes and interfaces in java it is like as header file in c

If we want to add the package in program then we have the import keyword

It is like as #include in c

###### **If we want to add the package in java we have the four ways**

**a) Wild card import**

**b) Single type import**

**c) Inline package import**

**d) Static package import**

**Note: inline and static package we will discuss in package chapter in detail.**

**Wild card import:** this is the package import where we can import the all member from package in application it is denoted by using \*

**Syntax:** import packagename.\*;

**e.g import java.util.\*;** means we can import or add all member from java.util package in application.

**Single type import:** single type import means we can add the specific member from package in application.

**Syntax:** import packagename.classname;

**Example:** import java.util. Scanner;

**2) Create the object of Scanner class:** so if we want to use the Scanner class we have to create its object

**Syntax:** Scanner ref = new Scanner (System. in);

**3) Use its methods or function for accept the input:** Scanner class provides the some inbuilt method to us for accept the input

**int nextInt():** this method is used for accept the input of type integer

**float nextFloat():** this method is used for accept the input of type float

**double nextDouble():** this method is used for accept the input of type double.

**long nextLong():** this method is used for accept the input of type long

**String nextLine():** this method is used for accept the input of type string.

etc

### **Example**

---

We want to accept the name id and per of student and display using Scanner

```
import java.util.*; //step1
public class StudentA
{ public static void main(String x[])
  { Scanner xyz = new Scanner(System.in);//step2
    String name;
    int id;
    float per;
    System.out.println("Enter the name id and percentage of student");
    name=xyz.nextLine(); //step3
    id =xyz.nextInt();
    per=xyz.nextFloat();
```

```
System.out.printf("Name is %s\n",name);
System.out.printf("Id is %d\n",id);
System.out.printf("Percentage is %f\n",per);
}
}
```

## Data types in java

---

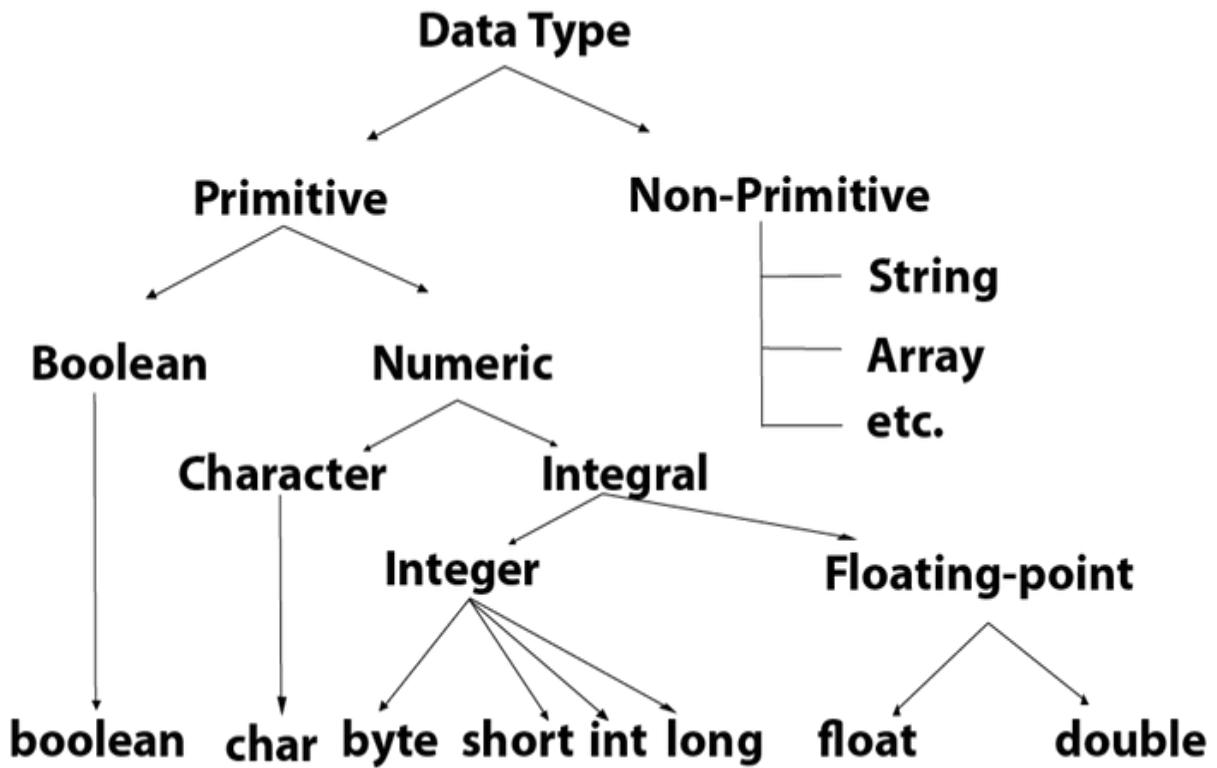
1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include Classes , Interfaces , and Arrays

### Primitive Data Type

---

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.

- 1) boolean data type
- 2) byte data type
- 3) char data type
- 4) short data type
- 5) int data type
- 6) long data type
- 7) float data type
- 8) double data type



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

### Boolean Data Type

The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions.

The Boolean data type specifies one bit of information, but its "size" can't be defined precisely.

Example: **boolean one = false**

### **Byte Data Type**

The byte data type is an example of primitive data type. It is an 8-bit signed two's complement integer. Its value-range lies between -128 to 127 (inclusive). Its minimum value is -128 and maximum value is 127. Its default value is 0. The byte data type is used to save memory in large arrays where the memory savings is most required. It saves space because a byte is 4 times smaller than an integer. It can also be used in place of "int" data type.

**byte a = 10, byte b = -20**

### **Short Data Type**

The short data type is a 16-bit signed two's complement integer. Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.

The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

Example:

**short s = 10000, short r = -5000**

### **Int Data Type**

The int data type is a 32-bit signed two's complement integer. Its value-range lies between - 2,147,483,648 (- $2^{31}$ ) to 2,147,483,647 ( $2^{31} - 1$ ) (inclusive). Its minimum value is - 2,147,483,648 and maximum value is 2,147,483,647. Its default value is 0.

The int data type is generally used as a default data type for integral values unless if there is no problem about memory.

**Example:**

`int a = 100000, int b = -200000`

### **Long Data Type**

The long data type is a 64-bit two's complement integer. Its value-range lies between -9,223,372,036,854,775,808 (- $2^{63}$ ) to 9,223,372,036,854,775,807 ( $2^{63} - 1$ ) (inclusive). Its minimum value is - 9,223,372,036,854,775,808 and maximum value is 9,223,372,036,854,775,807. Its default value is 0. The long data type is used when you need a range of values more than those provided by int.

**Example:**

`long a = 100000L, long b = -200000L`

### **Float Data Type**

The float data type is a single-precision 32-bit IEEE 754 floating point. Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating point numbers. The float data type should never be used for precise values, such as currency. Its default value is 0.0F.

**Example:**

`float f1 = 234.5f`

### **Double Data Type**

The double data type is a double-precision 64-bit IEEE 754 floating point. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency. Its default value is 0.0d.

**Example: double d1 = 12.3**

### **Char Data Type**

The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535 inclusive).The char data type is used to store characters.

**Example: char letterA = 'A'**

### **Operator Precedence Chart in Java**

---

<b>Operator</b>	<b>Precedence</b>
<b>Bracket</b>	( )
<b>Prefix increment ,decrement and unary</b>	++ -- + - ~ !
<b>Multiplicative</b>	/ * %
<b>Addititive</b>	+ -
<b>Postfix increment and decrement</b>	++ --
<b>Shift</b>	<< >> >>>
<b>Relational</b>	< > <= >= instance of
<b>Equality</b>	== !=
<b>Bitwise AND</b>	&
<b>Bitwise exclusive OR</b>	^
<b>Bitwise inclusive OR</b>	
<b>Bitwise AND</b>	&&
<b>Bitwise OR</b>	
<b>Ternary</b>	? :
<b>Assignment</b>	= += -= *= /= %= &= ^=

### MCQ Question

---

#### Question1 :

```
public class P {  
    public static void main(String[] args) {  
  
        int a = 10, b = 5, c = 1, result;  
        result = a++ + c - ++b;  
        // result = a-(2)-(6)  
        // result = 10-2=8  
        // result = 8-6  
        // result = 2  
        System.out.println(result);  
    }  
}
```

**Output:** 2

#### Description of above code:

If we think about the above code and think about the priority of operator then `++` operator having higher priority than `-` (binary minus) operator so `++` operator get executed before binary minus `(-)` so your expression look like as after execution of `++ result=a-(++c)-(++b)` means `result=10-(2)-(6)` after solve `++` operator we have the two binary minus operator present in equation so from left to right operator get executed means your equation get executed like as `result=10-2-6` solve first `result=8-6` solve later so final result is `result=2`.

#### Question2:

---

```
public class Second {  
    public static void main(String x1[]) {  
        int x = 0, y = 0, z = 0;  
        x = (++x + y--) * z++;  
        System.out.println("X is " + x);  
    }  
}
```

}

If we think about above code then its execution like as

Initially x=0,y=0;

x = (++x + y--) \* z++;

++x means first it gets incremented later gets initialised //++x=1

y-- means it gets initialised first later gets decremented //y-- =-1

z++ means first it gets incremented later gets initialised //z++ =1

x=(1+ -1) \* 1

x=0

### **Question 3:**

class Numbers{

```
    public static void main(String args[]){
        int a=20, b=10;
        if((a < b) && (b++ < 25)){
            System.out.println("This is any language logic");
        }
        System.out.println(b);
    }
}
```

### **Description of above code**

---

If we think about the code initially value of a=10 and b=20 and when if statement get executed then (a<b) get executed the value of a is 20 and the value of b is 10 so the first condition get failed and we use the && operator in if condition so the rule of && is when first condition get failed then second condition not check by && so b++ not executed so the value of b is 10

### **Question 4**

---

```
class MyClass
{
```

```
public static void main(String s[])
{ int i = 4;
  int j = 21;
  int k = ++i * 7 + 2 - j--;
  System.out.println("k = " + k+"\t J=" +j);
}
```

**Output: K= 16 J=20**

### **Explanation of above code**

---

If we think about the above code we have the expression `int k=++i *7+2-j—`  
Here we initialized the value `i=4` and `j=21` in this express we use the `++i` this is the pre increment and pre increment having higher priority than `*`, `+`, `-` and post increment so `++i` get executed very first so after executing `++i` your expression like as `int k =5*7+2-j—` after so the `++i` we have the `*` (mutliplicaton) is higher priority operator so `*` multiplication executed after multiplication our code is like as `int k=35+2-j—`after solve multiplication we have the `+` and `-` operator so `+` and `-` having same priority from left to right so `+` get executed first after `+` execution our equation is `int k=37-j—` after solve `+` we have the two operator `-` and `j-` Here `j-` is post increment so post increment having less priority than arithmetic operator so minus (`-`) operator solve first so your equation like as `int k= 37-21` so `k=16` and after initialized `k` value `j` will decrement so the value of `j` is `20`

### **Example**

---

```
class IncDec
{ public static void main(String s[])
  { int a = 1;
    int b = 2;
    int c;
    int d;
```

```
c = ++b;  
d = a++;  
c++;  
System.out.println ("a = " + a);  
System.out.println ("b = " + b);  
System.out.println ("c = " + c);  
System.out.println ("d = " + d);  
}  
}
```

### **Output**

```
a = 2  
b = 3  
c = 4  
d = 1
```

### **Description of above code**

---

In above code a is initialized as 1 and b is initialized as 2 c and d is not initialized if we think about the statement `c = ++b` then it is pre incremented means first increase the value of b means  $c=3$  and if we think about the statement `d = a++` here `a++` is post increment means first value of a is initialized in d means  $d=1$  and then `a++` executed means 1 increase by 1 from 1 to 2 so  $a=2$  and again `c++` means c is also increase by 1 means c was 3 so is c is increment by 1 so  $c=4$

When we print the a it will print 2 when we print the b it will print 3 and when we print c it will print 4 and d is 1

### **Example**

---

```
package org.techhub;  
public class Demo  
{  
    public static void main(String[] args)
```

```
{ int i, j, k, l = 0;  
    k = l++;  
    j = ++k;  
    i = j++;  
    System.out.println("l is "+i);  
}  
}
```

### Output

---

I is 1

### Code Description

---

If we think about the above code then we have the four variables i, j, k, l=0

If we think about k=l++ here we have the post increment so the initial value of l is 0 this value initialize in k first and after that l++ execute means after execution of this statement k=0 and l=1 and if we think about the j=++k statement here ++k is pre increment statement means first ++k execute and then value initialized in j means k =1 first and then value initialized in j means j is also 1 and if we think about the i=j++ here j++ is post increment means first value of j is initialized in i and then j++ execute means the value of i=1 and j=2 so output is 1

### Example

---

```
package org.techhub;  
public class Demo {  
    public static void main(String[] args) {  
        int dailywage = 4;  
        int number_of_days = 5;  
        int salary;  
        salary = number_of_days++ * --dailywage * dailywage++ * number_of_days--;  
        salary -= dailywage;  
        System.out.println(dailywage + " " + number_of_days + " " + salary);  
    }  
}
```

```
}
```

## Output

---

```
4 5 -4
```

### Description of above code

---

If we think about the above code we have the three variables dailaywage=4  
number\_of\_days=5 and salary

If we think about the statement salary=number\_of\_days++ \* --dailywage  
\*dailywage++ \* number\_of\_days- - in this expression top most priority operator  
is - -dailywage and - - dailywage execute very first after executing - - dailywage  
your expression look like as

salary= number\_of\_days++ \* dailywage \* dailywage++ if we put the values in this  
expression then expression look like as salary=5 \* 3 \* 3 so we have the two  
operator in expression ++ and \* but ++ in the form of post increment and post  
increment having less priority than \* so first multiplication get executed and  
result of multiplication stored in salary so salary=45 after solve the multiplication  
++ operator get executed number\_of\_days++ before increment

number\_of\_days=5 so it will increase by 1 and number\_of\_days=6 and after that  
dailywage++ execute so initially dailywage =4 but previous it is decrease by 1 with  
pre decrement operator so dailwage=3 at the time of multiplication but now  
dailywage++ execute so it will from 3 to 4 means now current value of  
dailywage=4 and after that number\_of\_days- - execute so number of days  
previous number\_of\_days=6 here again number\_of\_days decrease by 1 so it will  
again 5

and if we think about the statement salary = - dailywage so dailywage was 4  
previous and when we initialize it in salary using negative sign then it will stored

as -4 in salary and if we think about this statement `System.out.println(dailywage + " " + number_of_days + " " + salary);`

So we get the output 4 5 -4

### Example

---

```
package org.techhub;
class Demo
{ public static void main(String s[])
    { int i = 34.0;
        int j = 7;

        int k = i % j;
        System.out.println("k = " + k );
    }
}
```

### Output: compile time error

If we think about the above code we try to store int i=34.0 in integer and 34.0 is double value so we cannot store in integer directly so it will generate the compile time error at run time.

### Example

---

```
package org.techhub;
class Demo
{
    public static void main(String s[])
    {
        int x = 42;
        double y = 42.25;
        System.out.println("x mod 10 = " + x % 10 );
        System.out.println("y mod 10 = " + y % 10 );
    }
}
```

```
}
```

```
}
```

### Output

---

```
x mod 10 = 2
y mod 10 = 2.25
```

### Code description

---

If we think about the above code we try to apply the % mod operator on double value but after jdk 1.7 version of java % (mod) can apply on floating value so the result is mention above

### Example

---

Which of the following statements is true?

- 1. When  $a = 5$  the value of  $a \% 2$  is same as  $a - 4$
- 2. When  $a = 3$  the value of  $a * 3 * 3$  is greater than  $(a + 10) * 3$
- 3. When  $a = 7$  the value of  $a * 7 * 3$  is greater than  $(a * a + 7 * a + 3)$

Output: 1 and 3

### Code Description

---

Statement 1:  $a = 5$ .  $a \% 2 = 5 \% 2 = 1$  and  $a - 4 = 5 - 4 = 1$ . Both values are equal. So statement 1 is true.

Statement 2:  $a = 3$ .  $a * 3 * 3 = 27$  and  $(a + 10) * 3 = (3 + 10) * 3 = 13 * 3 = 39$ .

But 27 is not greater than 39. So statement 2 is false.

Statement 3:  $a = 7$ .  $a * 7 * 3 = 147$  and  $(a * a + 7 * a + 3) = (7 * 7 + 7 * 7 + 3) = (49 + 49 + 3) = 101$ . 147 is greater than 101. So statement 3 is true.

**Extra Tip:** Operator Precedence (highest to lowest)

\* / % left to right (associativity)

+ - left to right (associativity)

### Example

---

```
package org.techhub;
public class Demo {
    public static void main(String[] args) {
        System.out.println(10 * 5 + 100 * (25 * 11) / (25 * 10) * 10 - 5 + 7 % 2);
        int zx = (10 * 5 + 100 * (25 * 11));
        int yz = ((25 * 10) * 10 - 5 + 7 % 2);
        System.out.println(zx / yz);
    }
}
```

### Output

---

1146  
11

### Example

---

```
package org.techhub;
class Demo{
    public static void main(String args[])
    {
        int var1 = 42;
        int var2 = ~var1;
        System.out.print(var1 + " " + var2);
    }
}
```

### Output

---

42 -43

## Code Description

---

Unary not operator, `~`, inverts all of the bits of its operand. 42 in binary is 00101010 in using `~` operator on var1 and assigning it to var2 we get inverted value of 42 i.e 11010101 which is -43 in decimal.

## Example

---

```
package org.techhub;
class Demo {
    public static void main(String args[]) {
        int a = 3;
        int b = 6;
        int c = a | b;
        int d = a & b;
        System.out.println(c + "\t" + d);
    }
}
```

Output

---

7      2

And operator produces 1 bit if both operand are 1. Or operator produces 1 bit if any bit of the two operands is 1.

## Example

---

```
package org.techhub;
class Demo {
    public static void main(String args[])
    {
        byte x = 64;
        int i;
```

```
byte y;
i = x << 2;
y = (byte) (x << 2);
System.out.print(i + "\t" + y);
}
}
```

Output

---

256 0

Example

---

```
package org.techhub;
class Demo {
    public static void main(String args[])
    {
        int x;
        x = 10;
        x = x >> 1;
        System.out.println(x);
    }
}
```

Output

---

5

Example

---

```
package org.techhub;
class Demo {
    public static void main(String args[])
    {
        int a = 1;
```

```
int b = 2;
int c = 3;
a |= 4;
b >>= 1;
c <<= 1;
a ^= c;
System.out.println(a + " " + b + " " + c);
}
}
```

Output

---

```
3 1 6
```

Example

---

```
public class UnaryOperators {
    public static void main(String[] args) {
        int i = 5;
        System.out.print(i++);
        System.out.print(++i);
        System.out.print(i--);
        System.out.print(--i);
    }
}
```

Output

---

```
5775
```

Example

---

```
public class ShiftOperators {
    public static void main(String[] args) {
        System.out.print(8<<2);
        System.out.print(",");
    }
}
```

```
        System.out.print(8>>1);
    }
}
```

Output:

---

**32, 4**

**Example**

---

```
public class BitwiseNotOperator{
    public static void main(String[] args) {
        int i = 50;
        System.out.print(~i);
        System.out.print(",");
        System.out.print(~--i);
        System.out.print(",");
        System.out.print(~++i);
    }
}
```

Output

---

**-51,-50,-51**

**Example**

---

```
public class RelationalOperators {
    public static void main(String[] args) {
        int a = 4;
        int b = 5;
        System.out.print(a>b);
        System.out.print(",");
        System.out.print(a<b);
    }
}
```

**Output**

---

**false , true**

**Example**

---

```
public class LogicalOperators {  
    public static void main(String[] args) {  
        int a = 5;  
        int b = 10;  
        boolean flag = a>b;  
        System.out.println(!flag);  
    }  
}
```

**Output**

---

**True**

**Example**

---

```
public class TernaryOperators {  
    public static void main(String[] args) {  
        int a = 10;  
        String result = a > 5 ? "Hello 1" : "Hello 2";  
        System.out.println(result);  
    }  
}
```

**Output**

---

**Hello 1**

**Example**

---

```
class Ques  
{  
    public static void main(String args[])
```

```
{  
    byte b = 12;  
    int y = b;  
    b = b + 10;  
    System.out.println(b);  
}  
}
```

### **Output**

---

The program will lead to compile time error as explicit casting is required in the line,  $b = b + 10$ .

### **Explanation**

---

When you compile the above program, a compile time error occurs in the line,  $b = b + 10$  as explicit casting is required to assign an int value to a byte type. In the preceding program the byte type variable is declared and initialized to the value 12. Then the value of byte variable is assigned to the int variable, y as assigning byte type value to an int variable is possible. However the compilation error of loss of precision will occur while incrementing the byte type value with 10, i.e. int value. Therefore A, C, and D are incorrect options

### **Example**

---

**Which of the following are valid declarations of the main () method?**

- A. static main(String args[]){ }
- B. public static String main(String args[]) {... }
- C. public static void main(String args[]) {....}
- D. final static void main(String args[]) {....}

### **Output:**

---

The correct option is C.

**Explanation:** The following is a valid declaration of the static method: public static void main(String args[]) { //implementation of the main method }

Therefore, the correct option is C since the return type of the main method is void and is declared as static

**Example**

---

**Ram as a developer was asked to create a program using switch...case within for loop. Ram created the following program:**

```
class Ram {  
    public static void main(String args[]){  
        int z=3;  
        for(int i=0; i<2;i++)  
        {  
            z++;  
            switch(z)  
            {  
                case 3:  
                    System.out.print(z=z+1 + " ");  
                case 5:  
                    System.out.print(z=z+2 + " ");  
                    break;  
                default :  
                    System.out.print(z=z+8 + " ");  
            }  
        case 6:  
            System.out.print( z=z+4 + " ");  
        }  
    }  
}
```

```
    z--;
}
}
}
```

### Output

---

12 16 24 28

**Explanation:** Option D is correct Initially the variable z is initialized with 3, which becomes 4 inside the for loop therefore, the cases before default becomes false and default statement prints 12 then statement following default is executed because break is not used after default and therefore 16 will be printed. Then, the value of z decreases by 1 and becomes 15. Now, for loop executes once again and z becomes 16 and same process continued and z becomes 24 in default case and 28 in next case. Option B is incorrect because for loop is executed two times. Option A is incorrect because value of z is 4 and none of the case statement is 4 and same is the case in option C.

### Example

---

Imagine, you as a student provided with the following program during a class test

```
class Student {
    public static void main(String args[]){
        int z=6, k;
        for(int i=0; i<2;i++) {
            z++;
            switch(z) {
                case 3: System.out.print(z=z+1 + " ");
                case 5: System.out.print(z=z+2 + " ");
                break;
                default : {
```

```
for (int x=10; x>3; x++) {  
    System.out.print(x=k+x + " ");  
}  
}  
}  
case 6: System.out.print( z=z+4 + " ");  
}  
z--;  
}  
}  
}
```

**What would be the output of the preceding program?**

- A. Program will display 8 10 as an output
- B. Program will not compile successfully
- C. Program runs infinity endless
- D. Program will display 8 10 10as an output

**Output:** Option B is correct.

**Explanation:** Option B is correct. Program will not compile because k is not initialized. Option C will be correct when k is initialized but this time it is incorrect. Options A and D are incorrect because the program will not compile successfully.

### **Example**

---

**Ram as a student was provided with the following code snippet**

```
public void Ram(float c) {  
    switch (c) {  
        case 5:  
        case 7:  
        case 2:  
        default:  
        case 9.5:
```

```
}
```

```
}
```

After viewing the code snippet Ram was asked to notice the problems in preceding code snippet on the basis of the rules regarding switch...case statement. Following are the options from which Ram has to choose the correct answer.

- A. There is no problem in the code snippet
- B. Switch cannot evaluate float value
- C. The default statement cannot be used between case statements
- D. All cases must be in increasing order

### **Output**

---

Option B is correct.

**Explanation:** Option B is correct because switch...case statement can evaluate to a char, byte, short, int, or enum. Therefore Option A is incorrect because float value is being used as a case, which is not allowed. Option C is incorrect because default can be used anywhere in switch...case block. Option D is incorrect because it is not necessary that cases must be in increasing order.

### **Example**

---

Sheela as a faculty given following options to her students and asked them to choose the correct options:

- A. A switch statement can only evaluate to float and double values
- B. A switch...case block must have break statements after every case
- C. Switch case must be similar to switch expression type
- D. A switch...case can be nested like nested if...else

**Option D is correct.**

**Explanation:** Option D is correct because it is a fact.

Option A is incorrect because a switch...case cannot evaluate float and double.

Option B is incorrect because there is no need to have break statement after every case. Option C is incorrect, for example, your case expression is of char type but you used 65 as case label then internally that 65 is recognized as char A. Therefore, case expression and case label can be varied but must take attention before using them

### **Example**

---

Shyam during an interview was provided with following code and asked to review the program

```
public class Sam
{
    public static void main(String args[])
    {
        int x=0, i=0;
        for (int y=0; y>=i; ++y,i++) {
            System.out.println(y);
            System.out.println(i);
        }
    }
}
```

**After reviewing the code he was asked to predict the correct options among the following:**

- A. Program will print 0 0 for first time

- B. Program results in an endless loop
- C. Program will not compile because declaration is not allowed inside the for loop
- D. Program will successfully compile and print 0 0 on execution and then terminates

**Option B is correct.**

**Explanation:** Option B is correct because the value of y will always be equals to i and therefore loop will never terminates so, option A is incorrect.

Option C is incorrect because declaration can be done inside the for loop.

Option D is incorrect because program will successfully compile but when executes it becomes an endless loop.

**Example**

---

Rani during an interview was shown the following program:

```
class Rani {  
    public static void main(String args[]) {  
        int x = 0;int y=9;  
        for ( ; x<y; ) { x++; y++;} // (a)  
        for (x; x==y; --x) continue; // (b)  
        for (x=0; x<5; ) { x++; } // (c)  
        for ( ; ; ); // (d)  
    }  
}
```

What would be the output of the preceding program from the following options?

- A. Program will successfully compile and executes but does not print any value
- B. Program will successfully compile and becomes endless because of loop d
- C. Program will not compile because loop b is syntactically incorrect

D. Program will not compile because loop a has only expression part but missing initialization and Increment/decrement part

**Option C is the correct answer.**

**Explanation:** Correct answer is option C because the loop is not initialized. Therefore, options A and B are incorrect. Option D is incorrect because syntax of loop is correct.

**Example**

---

Shyam was given the following code snippet during an interview and asked to choose all correct decisional and loop statements:

```
int y=9;  
for ( ;true ; ) { break;} // 1  
if(y==9) { break; } // 2  
switch(y) {default: break;} // 3  
do ( ){ // code } while(expression); // 4  
while ( ) { //code } // 5
```

**Options:**

- A. Statement 1 and 3 are correct
- B. Statement 1, 2, and 3 are correct
- C. Statement 1, 3, and 5 are correct
- D. Statement 1,4, and 5 are correct
- E. Statement 1, 2, and 4 are correct

**Correct option is A.**

**Explanation:** Correct option is A. Reason for all other options can be verified on the basis of following facts:

The break statement can only be used in looping constructs and if need to use with if then if must be inside a loop the do... while do not have expression part with do i.e. do ( ).The expression is used in while block. The while block Cannot be used without specifying expression.

**Example**

---

Shyam was given the following program by his teacher

```
class Sam {  
    public static void main(String args[])  
{  
    int y=2;int i;  
    for (i=0; i <= 3; i++) {  
        if (i == 2) {  
            break;  
        }  
        else  
        {  
            y++;  
        }  
    }  
    System.out.println(i + ", " + y);  
}  
}
```

**What would be the output of the preceding code?**

- A. Program will display 2 , 2 as an output
- B. Program will display 2 , 3 as an output
- C. Program will display 2 , 4 as an output
- D. Program will display 1 , 2 as an output

**Option C is the correct.**

**Explanation:** Option C is the correct output. Inside the for loop condition `i==2` is checked and when it Evaluates to false the value of `y` will increment by 1. This process continues until `i` is not equal to 2. When the condition `i==2` evaluates to true then the loop terminates and the vales of `i` and `y` are printed.

Rest all of the options are incorrect as they are representing incorrect output.

### **Example**

---

Sheela after attending a lecture on for statement was shown the following for statements to choose the correct for statement:

- A. for( int j=2; j\*j==4, j<4; j++)
- B. for (int j=3; j/2==1; j++)
- C. for (int j=3, long k=0; j>k; j++)
- D. int k, j; for (j=3, k=2; k==j-1; k++, j--)

**Options B and D are correct for statements.**

**Explanation:** Options B and D are correct for statements.

Option A is incorrect because multiple conditions cannot be used in for statement. Option C is incorrect because different type of initialization variable cannot be declared inside the for loop rather they can be declared outside the for loop.

### **Example**

---

**Shyam works in a xyz company and he designed the following program:**

```
class Shyam {  
    public static void main (String args[]) {  
        int x=2; int y=6;  
        if( x!=y || (y*x)!=x) {  
            System.out.println(" Not equal");  
        }  
        else{  
            System.out.println(" Equal");  
        }  
    }  
}
```

What happens when he compile and run the preceding program?

- A. Program will display Equal
- B. Program will display Not Equal
- C. Program will not compile successfully because if statement is not correct
- D. Program will compile but not executes

**Option B is the correct answer.**

**Explanation:** Option B is correct because the first expression in if statement evaluates to true therefore Second expression is not checked and the statement in else block is displayed. Option A is incorrect because if statement evaluates to false. Options C and D are incorrect because the program compiles and executes successfully

### **Example**

---

**Reems and Sam while preparing for Java certification created the following program:**

```
public class Rose{  
    public static void main(String[] args)  
{  
        char x = 'a';  
        switch(x)  
        {  
            case 66: System.out.println( "B" + " ");break;  
            case 72: System.out.println( "H"+ " ");break;  
            case 97: System.out.println("a"+ " ");  
            case 89: System.out.println( "Y" + " ");break;  
            default: System.out.println( "default");break;  
        }  
    }  
}
```

```
}
```

```
}
```

**What would be the output of this program? Choose the correct option from the following options:**

- A. Program will display a Y default
- B. Program will display a Y
- C. Program will not compile successfully because break cannot be used with default case.
- D. Program will display A

**Option B is the correct answer.**

**Explanation:**

Option B is correct because ASCII equivalent of small a is 97 and because break is not used so, statement following this case is also displayed. Option A is incorrect because break statement is used in case 89 (ASCII equivalent of Y).

Option C is incorrect because break can be used with default statement.

Option D is incorrect because ASCII equivalent of capital A is 65, which is not a case in this program.

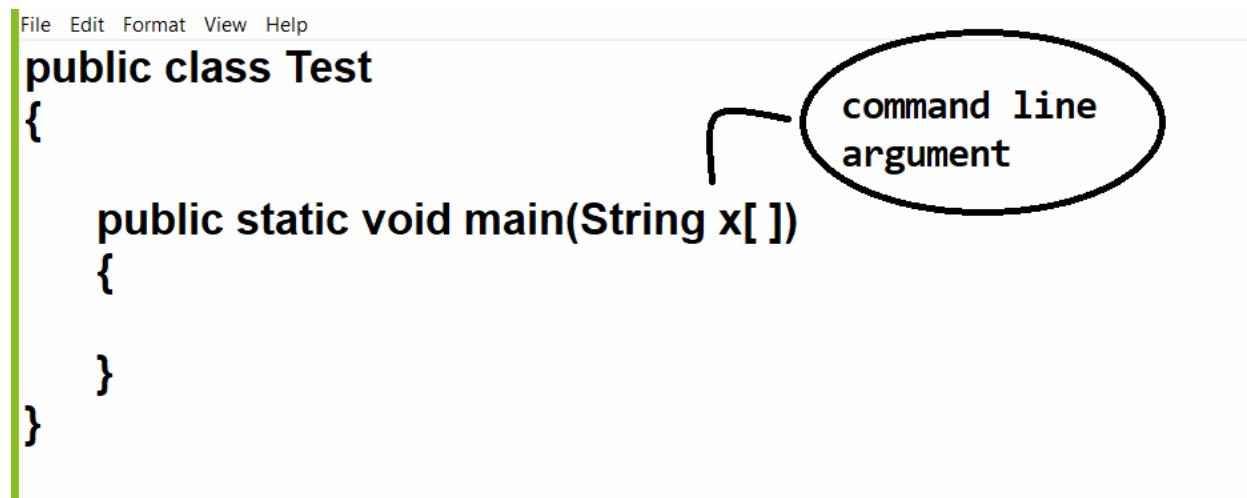
## Command Line Arguments

---

### Q. what is the command line arguments?

Command line argument is parameter passed

In main function of string array called as command line argument.



Command line argument is infinite string array means we can accept the n number of input through the command line argument but the first input of command line argument is start from 0<sup>th</sup> location

### Example:

---

We want to accept the two values from keyboard and calculate its addition.

If we use the command line argument then first input on 0<sup>th</sup> index ,second input on 1th index up to n-1 shown following example

```
public class Test
{
    public static void main(String x[ ])
    {
        int a,b,c;
        a=x[0]; //first input
        b=x[1];//second input
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

Command Prompt

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac Test.java
Test.java:6: error: incompatible types: String cannot be converted to int
        a=x[0]; //first input
                  ^
Test.java:7: error: incompatible types: String cannot be converted to int
        b=x[1];//second input
                  ^
2 errors
```

Above code generate the two error to us String cannot be converted to integer

### **Q.Why ?**

---

Because we want to accept the input of type integer but we have the String for input and String cannot store in integer directly so compiler generate the incompatible type error to us

### **Q. How to Solve This Type Of Error in Java ?**

---

If we want to solve this type of error in java we have the type casting technique.

## **Q. What Is The Type Casting?**

---

Type Casting means convert the one type of data in to the another type for single line of code called as Type Casting.

Means As per our Example we required to convert the String to integer

## **Q. How To Convert String To Integer In Java?**

---

If we want to convert the string to integer in java we have the Following Statement

**Syntax:** `int variablename =Integer.parseInt(String);`

Here Integer is classname and parseInt() is function which is used for convert the String to integer in java.

e.g `int a =Integer.parseInt(x[0]);`

if we want to convert the string to float we have the statement

**syntax:** `float variable =Float.parseFloat(String);`

if we want to convert the string to double we have the statement

**Syntax:** `double variable=Double.parseDouble(String);`

**Following example demonstrate the conversion between string to integer for Addition Example**

```
public class Test
{
    public static void main(String x[ ])
    {
        int a,b,c;
        a=Integer.parseInt(x[0]); //first input
        b=Integer.parseInt(x[1]);//second input
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac Test.java
C:\Program Files\Java\jdk1.8.0_291\bin>java Test
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
at Test.main(Test.java:6)
```

If we think about above code then program get compile successfully

But program generate the error at runtime

### **Q. why ?**

Because if we use the Command Line Argument then we need to accept the input on same line where we run the program

Shown below

```
public class Test
{
    public static void main(String x[ ])
    {
        int a,b,c;
        a=Integer.parseInt(x[0]); //first input
        b=Integer.parseInt(x[1]);//second input
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

Command Prompt

C:\Program Files\Java\jdk1.8.0\_291\bin>javac Test.java

C:\Program Files\Java\jdk1.8.0\_291\bin>java Test 100 200  
x[0] x[1]

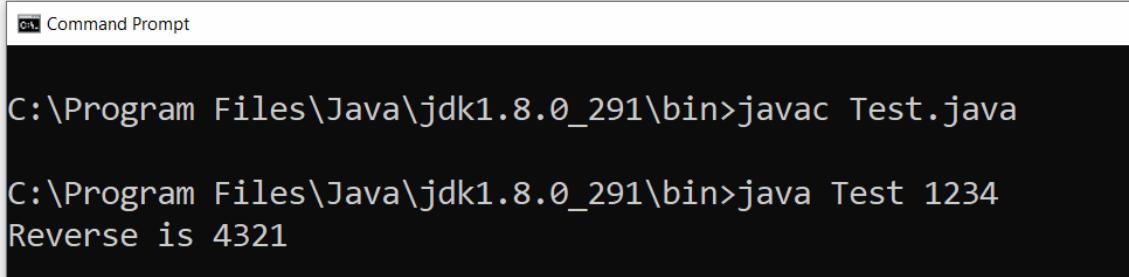
If we use the command then we must be give all input in single line where we execute the program

## Now We Will See The Some Example Using Command Line Arguments

### Example1

WAP to input the number and reverse it using command line argument?

```
public class Test
{
    public static void main(String x[ ])
    {
        int no = Integer.parseInt(x[0]);
        int r=0,rem;
        while(no!=0)
        {
            rem = no % 10;
            no = no /10;
            r = r *10 + rem;
        }
        System.out.printf("Reverse is %d\n",r);
    }
}
```



The screenshot shows a Windows Command Prompt window titled "Command Prompt". It displays the following text:  
C:\Program Files\Java\jdk1.8.0\_291\bin>javac Test.java  
  
C:\Program Files\Java\jdk1.8.0\_291\bin>java Test 1234  
Reverse is 4321

---

## Example2

---

WAP Input The Two Values Consider First As Base And Second As Index  
And Calculate The Power of Number

e.g 5 4

4

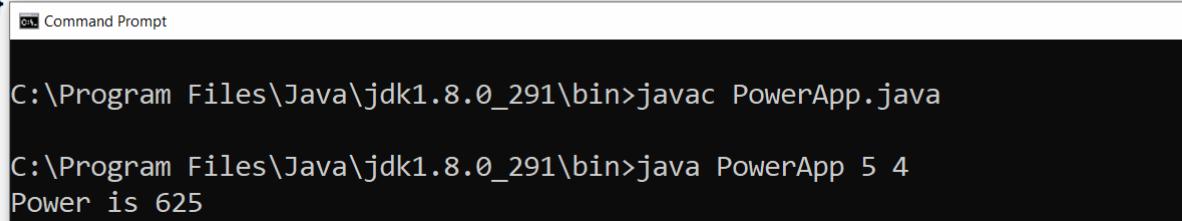
$$5 = 5 \times 5 \times 5 \times 5 = 625$$

```
public class PowerApp
{
    public static void main(String x[])
    {
        int base,index,p=1;

        base =Integer.parseInt(x[0]);

        index=Integer.parseInt(x[1]);

        for(int i=1; i<=index; i++)
        {
            p = p * base;
        }
        System.out.printf("Power is %d\n",p);
    }
}
```



```
Command Prompt
C:\Program Files\Java\jdk1.8.0_291\bin>javac PowerApp.java
C:\Program Files\Java\jdk1.8.0_291\bin>java PowerApp 5 4
Power is 625
```

---

### Example 3

---

WAP to input the number and calculate its Factorial using Command Line Argument?

```
public class FactorialApp
{
    public static void main(String x[])
    {
        int no=Integer.parseInt(x[0]);

        int f=1,i;

        for(i=1;i<=no;i++)
        {
            f = f * i;
        }
        System.out.printf("Factorial of number is %d\n",f);
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac FactorialApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java FactorialApp 5
Factorial of number is 120
```

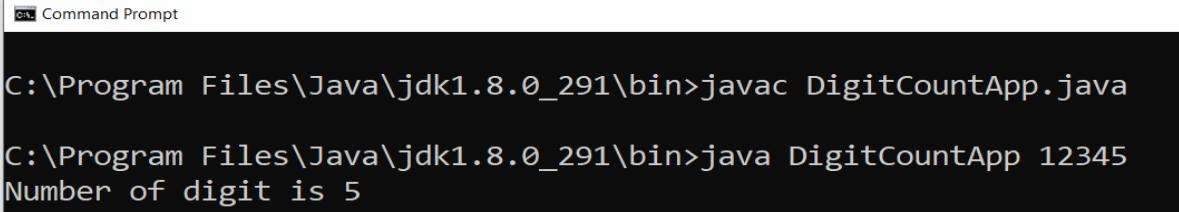
---

#### Example 4

---

WAP to input the number and count its digit using command line argument ?

```
public class DigitCountApp
{
    public static void main(String x[])
    {
        int no =Integer.parseInt(x[0]);
        int count=0;
        while(no!=0)
        {
            no = no /10;
            ++count;
        }
        System.out.printf("Number of digit is %d\n",count);
    }
}
```



```
Command Prompt
C:\Program Files\Java\jdk1.8.0_291\bin>javac DigitCountApp.java
C:\Program Files\Java\jdk1.8.0_291\bin>java DigitCountApp 12345
Number of digit is 5
```

---

## Example 5

---

WAP to input the number using command line argument and check number is perfect or not?

Description: Perfect number means number is equal with sum of all divisor

e.g 6 is perfect number because the divisor 6 1 2 and 3 and if we calculate the addition of these divisor  $1+2+3 = 6$

Means 6 is equal with its divisor addition called so 6 is perfect number.

```
public class PerfectApp
{ public static void main(String x[])
{
    int no=Integer.parseInt(x[0]);
    int sum=0;
    for(int i=1; i<no;i++)
    {
        if(no%i==0)
        { sum = sum + i;
        }
    }
    if(sum==no)
    { System.out.printf("Number is perfect");
    }
    else
    {
        System.out.printf("Number is not perfect");
    }
}
```

Command Prompt

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac PerfectApp.java
C:\Program Files\Java\jdk1.8.0_291\bin>java PerfectApp 6
Number is perfect
C:\Program Files\Java\jdk1.8.0_291\bin>
```

## **Scanner class**

---

If we use the command line argument then we must accept the input on same line. We cannot accept the Input on New line.

If we want to accept the input on new line we have the Scanner class

If we want to use the Scanner class we have the Following Steps.

### **Steps to Use Scanner Class**

---

- i) Add the java.util package in application

#### **Q. What is the Package ?**

---

Package is collection of classes and interfaces in java it is like as header file in c

If we want to use the any package in java we need to import it in application.

If we want to import package in java we have the import keyword

Syntax:

---

**import packagename.\*;**

**e.g import java.util.\*;**

Or

**import packagename.classname;**

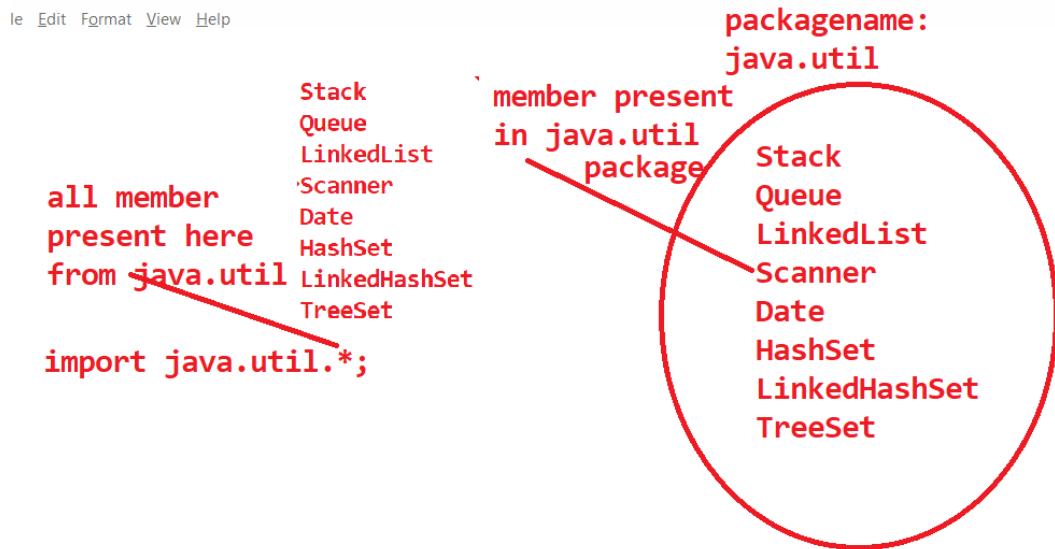
**e.g import java.util.Scanner;**

As per the standard of java there are major four ways to import the package in application

a) **Wild card import** : it is denoted by \* if we use the \* means we can import the all member from package in application.  
Syntax: import packagename.\*;  
e.g import java.util.\*;  
above statement indicate we can import the all member from java.util package in application.

Following diagram shows meaning above statement

---



Note: The Major limitation of wild card import is there is possibility in different package may be contain the same name member and if we import the packages in program in which member name is same and if we create the object of class whose name same in different packages those are imported in application so compiler may be get confused and if we want to solve this problem we have the single type import.

b) **Single type import:** single type import means we can import the specific member from package in application.

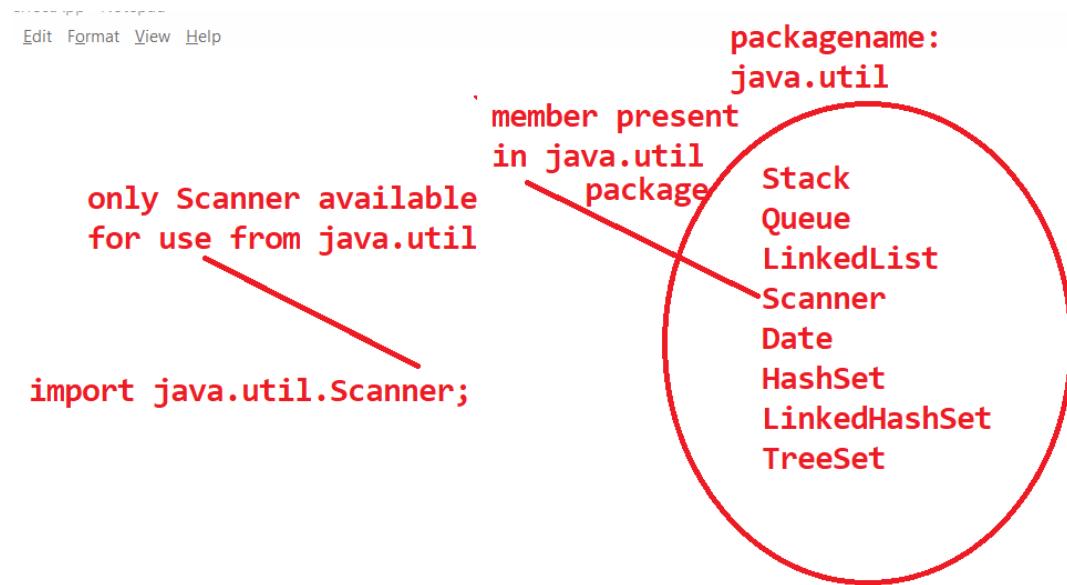
Syntax: import packagename.classname;

e.g import java.util.Scanner;

above statement indicate we can use the single member from java.util package in application.

Following diagram shows meaning of above statement

---



Note: The Major limitation of single type import is cannot use the same name member from same package in application

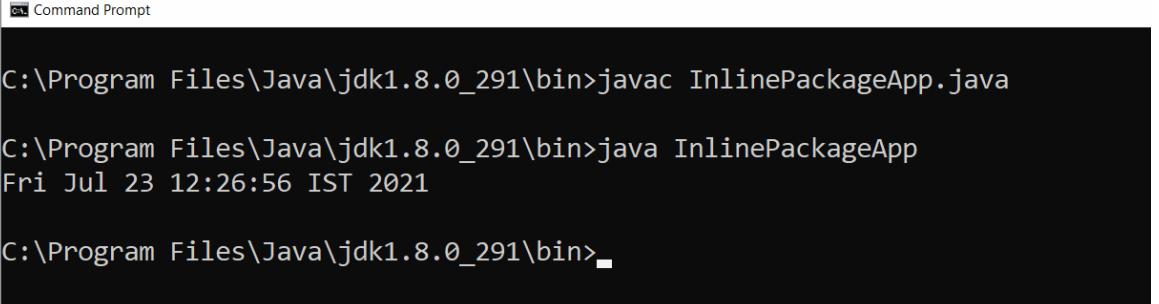
In this case we have one more approach name as inline package import.

c) Inline package import: inline package import means package import where we can import the member in application but without using import keyword.

Syntax:

```
packagename.classname ref=new packagename.classname();  
e.g java.util.Date d = new java.util.Date();
```

```
public class InlinePackageApp  
{  
    public static void main(String x[])  
    {  
        java.util.Date d = new java.util.Date();  
        java.sql.Date d1 = new java.sql.Date(5);  
        System.out.println(d);  
    }  
}
```



```
Command Prompt  
C:\Program Files\Java\jdk1.8.0_291\bin>javac InlinePackageApp.java  
C:\Program Files\Java\jdk1.8.0_291\bin>java InlinePackageApp  
Fri Jul 23 12:26:56 IST 2021  
C:\Program Files\Java\jdk1.8.0_291\bin>
```

d) Static package import:

Note: we will discuss classes and object chapter

---

ii) Create the object of Scanner class

**Syntax:** Scanner ref = new Scanner(System.in);

iii) Use its method/function of Scanner for accept the input

Scanner class provide the some inbuilt method/function to us to accept the input from keyboard .Scanner class provide the separate method to

Us for separate type of input .

## Methods or functions of Scanner class

---

`int nextInt():` this method is used for accept the input of type integer

`float nextFloat():` this method is used for accept the input of type float

`double nextDouble():` this method is used for accept the input of type double.

`long nextLong():` this method is used for accept the input of type long

`String nextLine():` this method is used for accept the input of type string

`short nextShort():` this method is used for accept the input of type short.

Etc

### Example 1

---

WAP To input the name id and salary of employee and display it using Scanner class.

```
import java.util.*;//step1
public class EmployeeApp
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in); //step2
        System.out.println("enter the name id and salary of employee");
        String name=xyz.nextLine(); //step3
        int id=xyz.nextInt();
        int sal=xyz.nextInt();
        System.out.printf("Name is %s\n",name);
        System.out.printf("Id is %d\n",id);
        System.out.printf("Salary is %d\n",sal);
    }
}
```



The screenshot shows a Command Prompt window. The user runs 'javac EmployeeApp.java'. When prompted, they enter 'ram' for the name, '1000' for the ID, and '10000' for the salary. The program then outputs 'Name is ram', 'Id is 1000', and 'Salary is 10000'.

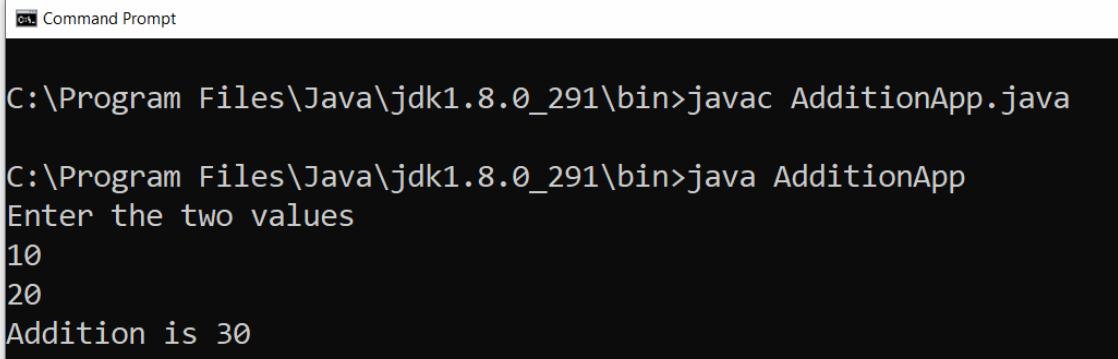
---

## Example 2

---

**WAP To input the two values and calculate its Multiplication using Scanner**

```
import java.util.*; //step1
public class AdditionApp
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in); //step2
        System.out.println("Enter the two values");
        int a=xyz.nextInt(); //step3
        int b=xyz.nextInt();
        int c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```



```
Command Prompt
C:\Program Files\Java\jdk1.8.0_291\bin>javac AdditionApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java AdditionApp
Enter the two values
10
20
Addition is 30
```

## Arrays

Array is a collection of similar data type and which is used for store the more than one values in single variables.

### How to declare the array in java

If we want to declare the array in java we have the two steps

**1) Declaration of array:** in declaration we specify the type of array not its size.

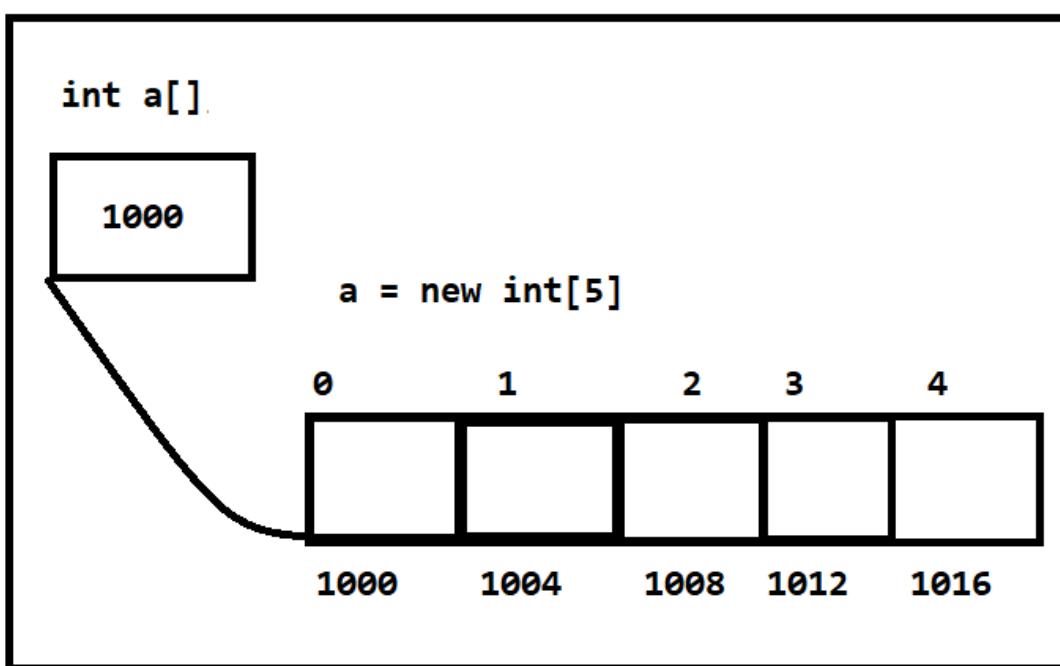
**Syntax:** datatype variableName []; //declaration

e.g int a[] ;//at the time declaration array variable initialize to null

**2) Memory allocation of array:** memory allocation means we decide the actual size of array at run time and in java array allocated by using new keyword and in java array consider as reference type or object type.

**Syntax:** variable name = new datatype[size];

e.g a = new int[5];



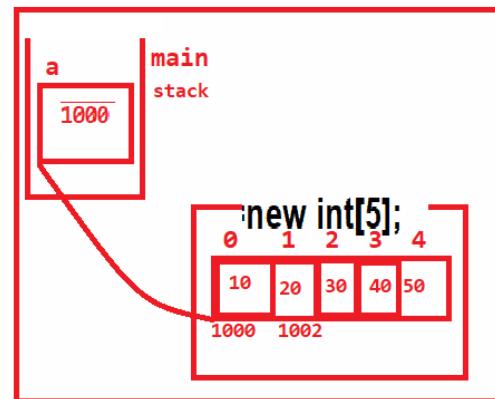
### Example

```

import java.util.*;//step1
public class ArrApp
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in); //step2
        int a[] = new int[5];
        System.out.println("Enter the five values in array\n");
        for(int i=0; i<a.length; i++)
        {
            a[i] = xyz.nextInt(); //step3
        }
        System.out.println("display the array values");
        for(int i=0; i<a.length; i++)
        {
            System.out.printf("a[%d] --->%d\n", i, a[i]);
        }
    }
}

```

Talking: GIRI'S TECH HUB



### Output

```

C:\Program Files\Java\jdk1.8.0_291\bin>java ArrApp
Enter the five values in array

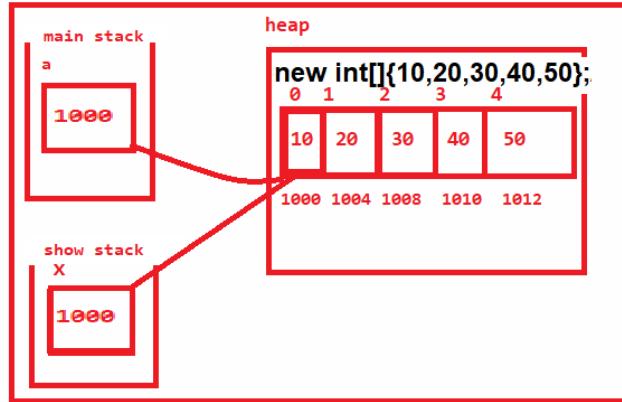
10
20
30
40
50
display the array values
a[0] --->10
a[1] --->20
a[2] --->30
a[3] --->40
a[4] --->50

```

## Example (what will be the output)

```
import java.util.*;//step1
public class ArrApp
{
    public static void main(String x[])
    {
        int a[]={10,20,30,40,50}//array initialization
        ArrApp ar = new ArrApp();
        ar.show(a);
        System.out.println("Display the array\n");
        for(int i=0; i<a.length; i++)
        { System.out.printf("a[%d] -->%d\n",i,a[i]);
        }
    }
    public void show(int x[])
    {
        for(int i=0; i<x.length; i++)
        {
            x[i]=x[i]+10;
        }
    }
}
```

Talking:



Example

Display the array

```
a[0] -->20
a[1] -->30
a[2] -->40
a[3] -->50
a[4] -->60
```

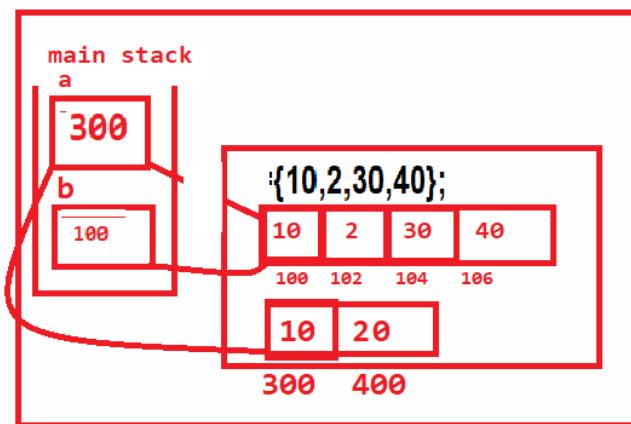
### Description of above code

In above code we declare the array name as **int a[]={10,20,30,40,50}** and we pass the array to the show function when we pass array to the function then array pass its base address to the function means as per our example show function contain

x[] points to the address where a array points so if we perform any change in x[] it will reflect on array a[] so we increase the value of array by ten using x[] array in show function it will effect on the memory where array a[] points so if we perform any change in array x[] it will reflect on array a[]

### Example (what will be the output of given code)

```
public class ArrApp
{
    public static void main(String x[])
    {
        int a[]={10,2,30,40};
        int b[];
        b=a;
        a=new int[]{10,20};
        for(int i=0; i<a.length;i++)
        {
            System.out.printf("%d\n",b[i]);
        }
    }
}
```



### Output

```
C:\Program Files\Java\jdk1.8.0_291\bin>java ArrApp
10
2
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>
```

We can declare the array in java like as datatype [] variablename;

E.g. int [] a; //valid declaration

### Description of above code

If we think about the statement `int a[]={10,2,30,40}` means we create array in memory and its address stored in reference `a[]` and we have the one more statement `int b[]` means we create one more array name as `b` and we have the statement `b=a` means we initialize the address of `a` in `b` means `b` array points where array `a[]` points and we have statement `a=new int[]{10,20}` means we have the new array and reference `a` points to new array means a release the memory of previous created array and points to new array i.e 10 and 20 but `b` array points to created previously array and we have the following statement

```
for(int i=0;i<a.length;i++){
```

```
    System.out.printf("%d\n",b[i]);  
}
```

here we travel the loop according to size of `a` means 2 time because new array size is 2 and `a` points to the array and we print the values of `b` array so it will print the first two values where `b` array points means `b` array points to first array i.e 10 2 is our final output .

### **What is the diff between array declaration using `a[]` and `[]a` ?**

---

If we give array sub script at left hand side then all variables in that line consider as by default array e.g `int [] a,b,c,d;` here `a` `b` `c` and `d` all are the arrays and we give the bracket at right hand side then specific variable consider as array e.g `int a[],b,c[],d;` here `a` and `c` is array but `b` and `d` is normal variable.

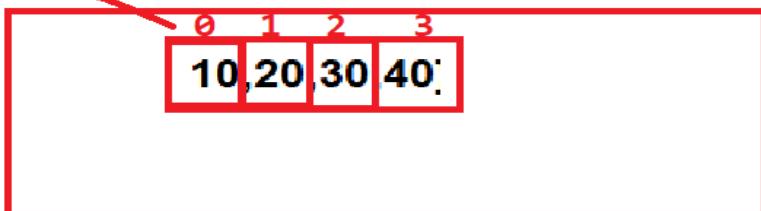
### **Example**

---

```
import java.util.*;//step1
public class ArrApp
{ static int i;
  public static void main(String x[])
  {
    int []a={10,20,30,40};
    20 int b=a[++i]; a[1]
    32int c=a[i+1]+2;
    21 int d=a[i++]+1; 20+1

    73 int e= b+c+d;
      20+32+21=73
    System.out.printf("D is %d\n",e);

  }
}
```



```
C:\Program Files\Java\jdk1.8.0_291\bin>javac ArrApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java ArrApp
D is 73

C:\Program Files\Java\jdk1.8.0_291\bin>
```

## Example

Consider the following various array declarations:

`int [] ar1, arr2[];`

`int[][] arr3;`

`int[] arr4[], arr5[];`

Which of the following options are true?

`arr2 = arr3;`

`arr2 = arr4;`

```
arr1 = arr2;  
arr4 = arr1;
```

### Output

The correct answer is A and B.

**Explanation:** There is a difference between int[] i; and int i[]; although in both the cases i is an array of integer values. Therefore the correct option is A and B as they are the declarations of the integer. The basic difference is that if you declare multiple variables in the same statement such as int [] i, j; and int i[], j; then it Implies that i and j are not of the same type.

### Example

---

Imagine you need to handle the records of multiple students and declaring a separate variable and then assigning the values will become a tedious task. Therefore, you write the following program to implement the concept of arrays which has simplified your task:

### Source code

---

```
public class Ques  
{ public static void main(String args[]) {  
    String[][][] arr = {  
        {  
            { "Suchita", "Vikash" , "Deepak"},  
            { "Charu", null, "Shikha"}  
        },  
        {  
            {"Shalini"}, {null}  
        },  
        {  
    }  
}
```

```
        {"Hemal"}  
    },  
    {  
        { "Santosh", "Manish"}, {}  
    }  
};  
System.out.println(arr[0][1][1]);  
}  
}
```

**What will be the output after compilation and execution of the preceding program?**

- A. The program will throw the runtime exception.
- B. The program will throw ArrayIndexOutOfBoundsException.
- C. The program will display null.
- D. The program will compile successfully but it will not display anything.

#### **Output**

---

**The correct option is C.**

**Explanation:** The program deals with three dimensional array and will display null as following is the structure of elements assigned to the arr array.

arr[0][0][0] = Suchita  
arr[0][0][1] = Vikash  
arr[0][0][2] = Deepak  
arr[0][1][0] = Charu  
arr[0][1][1] = null  
arr[0][1][2] = Shikha  
arr[1][0][0] = Shalini  
arr[1][1][0] = null  
arr[2][0][0] = Hemal

arr[3][0][0] = Santosh

arr[3][0][1] = Manish

Therefore, if you try to print the value of the arr[0][1][1] element of the arr array, then the null value will be displayed

### **Example**

---

Sam works in Xyz Company as Java programmer and he designed the following program:

### **Source Code**

---

```
class Rose{  
    public void sam() {  
        int y[] = {4, 2, 8};  
        for (int x=2; x<1+3*2-4; x++){  
            System.out.print(x+" ");  
            for (int j:y) {  
                j=j*x-4;  
                System.out.print(j+" ");  
            }  
        }  
    }  
    public static void main(String[] args) {  
        Rose r = new Rose();  
        r.sam();  
    }  
}
```

**What would be the output of this program? Choose the correct option from the following:**

- A. The program displays 2 4 2 8
- B. The program displays 2 4 0 12
- C. The program displays 2 4 4 16
- D. The program displays 3 4 0 12

**Correct option is B.**

**Explanation:** Correct option is B. Firstly the value of the variable x is 2, which is less than the 3 (Value specified in expression) therefore control will transfer into for each loop and prints the array variable with Modifications according to the expression. Array element is fetched into j. First array element is 4 and the Expression is  $j*x-4$  i.e.  $4*2-4=4$ . In the same way, other array elements are extracted and displayed with modifications.

### **Example**

---

**Sam as a developer in Dkinfotech created the following program**

```
class Rose {  
    static int j;  
    public int arr() {  
        int y[] = { 5 , 7, 8 , 6};  
        j = y[2]; return j;  
    }  
    public static void main (String args[]) {  
        Rose r=new Rose();  
        int x = r.arr( );  
        System.out.println(x);  
        switch(x) {  
            case 0: System.out.print(0 + " ");break;  
            case 2: System.out.print(2 + " ");break;  
            case 8: System.out.print(8 + " ");  
        }  
    }  
}
```

```
case 5: System.out.print(5 + " ");break;  
default: System.out.print("Default");  
}  
}  
}
```

**What would be the output when Sam compile and execute this program?**

- A. Program will not compile successfully
- B. Program will display 8 8 5
- C. Program will display 8 5
- D. Program will display 8

**Option B is the correct answer.**

**Explanation:** Option B is the correct answer. Firstly the control will transfer to arr () method and retrieves the array element at array [2] index and returns that value to calling routine. In the calling routine the retrieved value is first displayed and then used in switch statement. Therefore, the result is 8 8 and 5 is displayed because the case 8 does not have break and 5 is the statement written in next case. Therefore, options C and D are incorrect. Option A is incorrect because program will successfully compile.

## **Two Dimensional Arrays**

---

Two dimensional array is used for create the matrix but it is not a matrix internally.

### **Syntax:**

---

datatype variablename[][];

or

datatype [][]variablename;

or

datatype []variablename[];

e.g int []a,b[],c[];

//here a is single dimension b is three dimension and c is two dimension

### **Memory allocation**

---

variablename= new datatype[size][size];

### **Example**

---

```
import java.util.*;
public class MatrixApplication
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in);
        int a[][]=new int[3][3];
        System.out.println("Enter the values in matrix");
        for(int i=0; i<a.length; i++)
        {
            for(int j=0; j<a[i].length; j++)
            {   a[i][j]=xyz.nextInt();
            }
        }
        System.out.println("display the matrix");
        for(int i=0; i<a.length; i++)
        {
            for(int j=0; j<a[i].length; j++)
            { System.out.printf("%d\t",a[i][j]);
            }
        System.out.printf("\n");
        }
    }
}
```

}

## Jagged Array

---

Jagged Array is facility in java where we can create the matrix every row having different column list called as jagged array.

### Example we want to create the matrix like as

---

```
1 2 3  
4 5 6 7  
8 9
```

#### Syntax :

```
datatype variablename[][]=new datatype[rowsize][];  
variablename[rowindex]=new datatype[colszie];
```

### Example

---

```
import java.util.*;  
public class MatrixApplication{  
    public static void main(String x[]){  
        Scanner xyz = new Scanner(System.in);  
        int a[][]=new int[3][];  
        a[0]=new int[3];  
        a[1]=new int[4];  
        a[2]=new int[2];  
        System.out.println("Enter the values in matrix");  
        for(int i=0; i<a.length; i++){  
            for(int j=0; j<a[i].length; j++)  
                a[i][j]=xyz.nextInt();
```

```
    }
}
System.out.println("display the matrix");
for(int i=0; i<a.length; i++)
{   for(int j=0; j<a[i].length; j++)
    { System.out.printf("%d\t",a[i][j]);
    }
System.out.printf("\n");
}
}
}
```

## **Classes and objects**

---

Class is a combination of instance variable, class variable, method, constructor and instance initializer and nested classes or class is combination of state and behavior state means data or variable and behavior means function or method.

## **Why use the class or what is the benefit of class**

---

**1) Ability to store different data type:** class is complex data structure we can hold the different type of data in it.

Example:

```
class Product
{
    private int id;
    private String name;
    private float price;
    private String dealerName;
    private String compName;
    private int pprice;
    private int spprice;

    public void setDetail(String name,int id,float price,String dealerName,String
    compName,int pprice,int spprice)
    {
        this.name=name;
        this.id=id;
        this.price=price;
        this.dealerName=dealerName;
        this.compName=compName;
        this.pprice=pprice;
        this.spprice=spprice;
    }
    public void showDetail()
    {
```

```
System.out.println("Name is "+name)
System.out.println("Id is "+id);
System.out.println("Dealer Name "+dealerName);
System.out.println("Company Name "+compName);
System.out.println("Purchase Price "+pprice);
System.out.println("Selling price "+spprice);
}
}
```

**2) Provide the reusability:** means we can declare the class only once and can use it more than one time.

### **How we can reuse the class more than one time**

By creating object of class

### **What is the object?**

Object is block of memory where class data store or object is instance of class or object is run time entity. Means when we create the object of class then JVM create the block in heap section of memory and store the all class data non static data in it.

### **How we can create the object of class in java**

**classname ref= new classname();**

**e.g Product p = new Product();**

Here p is reference of Product class and new Product () is real object

### **What is the diff between reference and object?**

**Reference is variable which hold the address of object and object is block of memory where class data store.**

## **Following diagram shows the memory structure of object and reference**

```

class Product
{
    private int id; private String name;
    private float price; private String dealerName;
    private String compName; private int pprice;
    private int spprice;

    public void setDetail(String name,int id,float price,String dealerName,String compName,int pprice,int spprice)
    {
        this.name=name;
        this.id=id;
        this.price=price;
        this.dealerName=dealerName;
        this.compName=compName;
        this.pprice=pprice;
        this.spprice=spprice;
    }

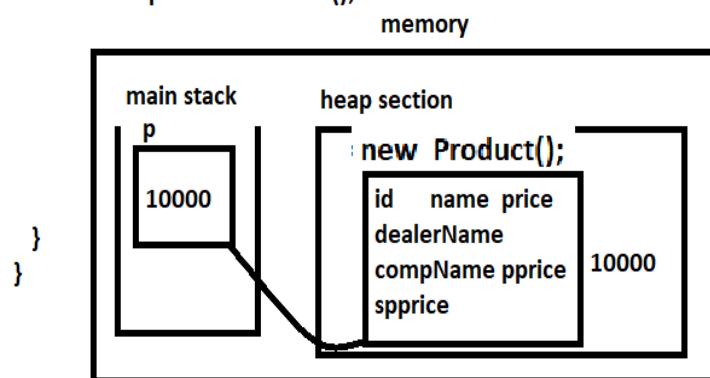
    public void showDetail()
    {
        System.out.println("Name is "+name);
        System.out.println("Id is "+id);
        System.out.println("Dealer Name "+dealerName);
        System.out.println("Company Name "+compName);
        System.out.println("Purchase Price "+pprice);
        System.out.println("Selling price "+spprice);
    }
}

```

```

public class ProductApplication
{
    public static void main(String x[])
    {
        Product p = new Product();
    }
}

```



Object created in heap section of memory and object contain the all data from class which declared as non static and reference can hold in the address of object and reference is stored in stack if we create its object in function or method

### **Why use the reference with object**

If we want to reuse the same object more than one time then we can reuse the reference with object.

**3) Provide encapsulation:** encapsulation means to hide the implementation detail from end user at implementation level or logical

level means encapsulation can achieve by declaring class variable as private and access via public or default function or setter or getter functions.

### **Can give real time scenario where we can implement the encapsulation**

Suppose consider we are developing application for shop keeper and in our application contain the three different types of login

#### **1. Admin or owner**

2. Sales Counter

3. Customer

So here we want to give the access of data of product according to is login type

**Admin:** we want to give the complete access of data to the admin like as product name, dealer name, purchase price, selling price, company name etc

**Sales Counter:** we want to give the partial access of data to the sales counter like as product name, selling price, companyname and discount amount etc

We want to hide the purchase price, dealer Name and some more important detail from sales counter

**Customer:** we want to provide only product name and company name access to the customer.

In this case we declare the all data or variables related with product as private and we design the function name as validateUser (String loginType) and in this function we write the logic for cross verification of user login and we provide the access of the data as per the logintype means if user is admin then we display all details and if user is salescounter then we provide the some detail above mention and if user is customer then we provide the detail mention above.

### **Following Example contain implementation of above scenario**

class Product

```
{ private int id;
private String name;
private String dealerName;
private String compName;
private int pprice;
private int spprice;

public void setDetail(String name,int id,String dealerName,String
compName,int pprice,int spprice)
{
this.name=name;
this.id=id;
this.dealerName=dealerName;
this.compName=compName;
this.pprice=pprice;
this.spprice=spprice;
}
public void validateUser(String loginType)
{
if(loginType.equals("admin"))
{
System.out.println(id+"\t"+name+"\t"+dealerName+"\t"+compName+"\t"+
pprice+"\t"+spprice);
}
else if(loginType.equals("salescounter"))
{ System.out.println(id+"\t"+name+"\t"+compName+"\t"+spprice);
}
else if(loginType.equals("customer"))
{ System.out.println(id+"\t"+name+"\t"+compName);
}
else{
System.out.println("invalid login");
}
}
```

```
}

}

public class ProductApplication
{
public static void main(String x[])
{
Product p = new Product();
p.setDetail("buscuits",1,"Parle Distributor","parle-G",3,5);
p.validateUser("customer");
}
}
```

If we think about the above code we have the function name as setDetails() it contain different type of parameters and the major limitation we need to maintain the sequence of parameter when we pass the parameter in it so it is very difficult to manage in real time when we have the n number of parameters so if we want to solve this problem we have the concept of POJO class

### **What is the POJO (Plain old java objects)?**

POJO is concept in java where we declare the class with setter and getter methods means we declare the variable of class as private and we create the functions name preceding with setter and getter methods .The major goal of POJO class is to store the data in objects it work like as container object in java.

### **How To Create Java Application using Eclipse**

#### **1) Create the Java Project**

File → New → Java Project → Give Project Name → Click on next button  
→click on finish button

#### **2) Create the package**

Right click on src → select new → select the package → give the package name and click on finish

**3) create the class (right click on package → new → class and give classname and finish**

**As per our above example we have the create POJO class name as Product**

package org.techhub;

public class Product {

    private int id;

    private String name;

    private String dealerName;

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getDealerName() {

        return dealerName;

    }

    public void setDealerName(String dealerName) {

        this.dealerName = dealerName;

    }

    public String getCompName() {

        return compName;

    }

    public void setCompName(String compName) {

        this.compName = compName;

GIRI'S TECH HUB PVT.LTD PUNE – 9175444433, 9049361265

```
}

public int getPprice() {
    return pprice;
}

public void setPprice(int pprice) {
    this.pprice = pprice;
}

public int getSpprice() {
    return spprice;
}

public void setSpprice(int spprice) {
    this.spprice = spprice;
}

private String compName;
private int pprice;
private int spprice;

}
```

Create the class name as Shop and pass the Product class reference in Shop class

When we pass the Product class reference in Shop class means we use the all information about the Product in Shop class.

### Example

```
public class Shop {

    Product prod;
    public void setProductInfo(Product prod)
    {
        this.prod=prod;
    }
}
```

```
public void validateUser(String loginType)
{
if(loginType.equals("admin"))
{
System.out.println("Product Id is "+prod.getId());
System.out.println("Product Name is "+prod.getName());
System.out.println("Product Purchase Price is "+prod.getPprice());
System.out.println("Product Selling Price is "+prod.getSpprice());
System.out.println("Product Dealer Name "+prod.getDealerName());
}
else if(loginType.equals("customer"))
{
System.out.println("Product Id is "+prod.getId());
System.out.println("Product Name is "+prod.getName());
System.out.println("Product Purchase price is "+prod.getSpprice());
System.out.println("Product Company name is "+prod.getCompName());
}
else {
System.out.println("Invalid user");
}
}
```

Create the class with main method and create the object of Shop class and Create the object of Product class and store data in it using setter methods and pass the product reference to Shop class and use its data in shop class.

```
package org.techhub;
public class ProductApplication {
public static void main(String[] args) {
Shop s = new Shop();
Product p = new Product();
p.setId(1);
p.setCompName("Parle-G");
```

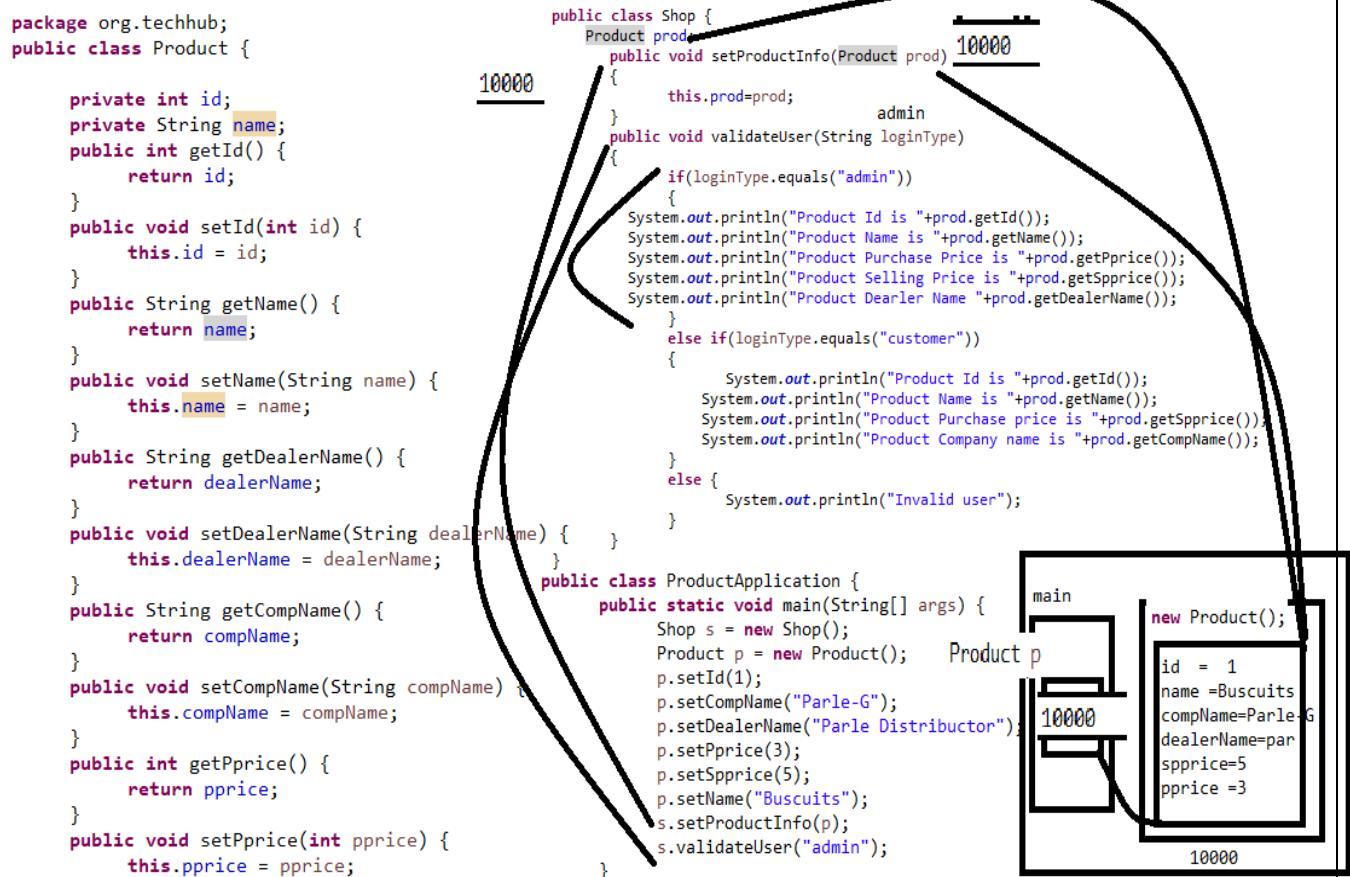
```

p.setDealerName("Parle Distributor");
p.setPprice(3);
p.setSpprice(5);
p.setName("Buscuits");
s.setProductInfo(p);
s.validateUser("admin");
}

}

```

Below diagram shows the working of above code



**WAP to create the class name as CalPer with two functions**

```
void acceptMarks(int s1,int s2,int s3,int s4,int s5,int s6)
void showPer()
```

### **Example**

---

```
package org.techhub;
class CalPer
{
    int s1,s2,s3,s4,s5,s6;
void acceptMarks(int s1,int s2,int s3,int s4,int s5,int s6)
{
    this.s1=s1;
    this.s2=s2;
    this.s3=s3;
    this.s4=s4;
    this.s5=s5;
    this.s6=s6;
}
void showPer()
{
    int agg=s1+s2+s3+s4+s5+s6;
    int per=agg/6;
    System.out.println("Percentage is "+per);
}
}

public class MarksApplication {
public static void main(String[] args) {
    CalPer cp = new CalPer();
    cp.acceptMarks(60, 60, 60, 60, 60, 60);
    cp.showPer();
}
}
```

If we think about the above code in acceptMarks () function contain six parameter of same type it is not good approach to pass parameter to function so better way we can pass array as parameter in function.

Following Statement shows the meaning of above statement  
Example

---

```
class CalPer
{
    int marks[];
    int agg=0,per;
    void acceptMarks(int marks[])
    {
        this.marks=marks;
    }
    void showPer()
    {
        for(int i=0; i<marks.length;i++)
        {
            agg=agg+marks[i];
        }
        per=agg/marks.length;
    }
}
public class MarksApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        CalPer cp = new CalPer();
        int a[]={60, 60, 60, 60, 60, 60};
        cp.acceptMarks(a);
        cp.showPer();
    }
}
```

Following Diagram shows the working of above code

---

```

class CalPer
{
    int marks[];
    int agg=0,per;
    void acceptMarks(int marks[])
    {
        this.marks=marks;
    }
    void showPer()
    {
        for(int i=0; i<marks.length,i++)
        {
            agg=agg+marks[i];
        }
        per=agg/marks.length;
    }
}
public class MarksApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        CalPer cp = new CalPer();
        int a[]={60, 60, 60, 60, 60, 60, 60};
        cp.acceptMarks(a);
        cp.showPer();
    }
}

```

1000

```

int a[]={60, 60, 60, 60, 60, 60, 60};
          0   1   2   3   4   5

```

1000	60	60	60	60	60	60
1000	1002	1004	1006	1008	1010	1012

## Method with Variable arguments

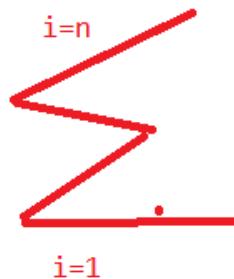
Method with variable argument is facility in java where we can pass the infinite parameter to the function

### Example

---

```
class Sum
{
    void setValue(int a,int b)
    {
        }
}
```

Note: if we think about setValue() function this function accept only two parameter means we can calculate the sum of only two numbers but if we want to implement the given formula



i=n  
if we want to implement this type of sum  
then above code get failed  
so here we required infinite parameter list  
to the function  
so if we want to pass infinite parameter to the  
function we have the method with variable  
argument concept in java  
i=1

### How to use the method with variable arguments

---

If we want to work with method variable arguments then we have the following syntax

Syntax:

```
returntype functionname (datatype ...variablename)
{write here your logics
}
```

Here ... indicate the variable arguments to the function

... is internally dynamic array in java means we can expand or shrink its size at run time as per your need.

**Following Example shows the above concept**

---

```
package org.techhub;
class Sum
{ int s=0;
void setValue(int ...x)
{   for(int i=0;i<x.length;i++)
{   s=s+x[i];
}
System.out.println("Sum is "+s);
}
}

public class CalSumApp
{ public static void main(String[] args) {
Sum s = new Sum();
s.setValue(60,60,60,60,60,60);
}
}
```

**Following diagram shows the working of above code**

---

```

x[0] x[1] x[2] x[3] x[4] x[5]
  60  60  60  60  60  60
1 package org.techhub;
2 class Sum
3 { int s=0;
4 void setValue(int ...x)
5 {
6     for(int i=0;i<x.length;i++)
7     {
8         s=s+x[i];
9     }
10    System.out.println("Sum is "+s);
11}
12}
13 public class CalSumApp
14 {
15 public static void main(String[] args) {
16     Sum s = new Sum();
17
18     s.setValue(60,60,60,60,60,60);
19
20 }
21
22 }
23

```

If we want to work with method with variable arguments we have the some important points

- 1) ... must be left hand side of variable not right hand side of variable
- 2) In single function we can pass only one variable argument we cannot pass multiple variable or ... argument in function
- 3) If function contains some simple argument and some variable arguments then variable arguments must be the last parameter in function

### **Example**

---

```

package org.techhub;
class Sum
{ int s=0;
void setValue(String name,int ...x)
{
System.out.println("Name is "+name);
for(int i=0;i<x.length;i++)
{
s=s+x[i];

```

```
}

System.out.println("Sum is "+s);
}
}

public class CalSumApp
{   public static void main(String[] args) {
Sum s = new Sum ();
s.setValue ("Ram", 60, 60, 60, 60, 60, 60);
}
}
```

### **MCQ Questions on class and method with variable arguments**

#### **Question1 : what will be the output of given code?**

```
public class VarargsExample
{ public static void displayNames(String... names)
{   for (String mynames:names)
{ System.out.print(mynames + " ");
}
}

public static void main(String args[])
{   displayNames("Alex","Richard","John");
}
}
```

**What will be the output after compiling and executing the preceding program?**

- A. The program leads to compilation error.

- B. The program compiles successfully and displays “Alex Richard John” as output.
- C. The program compiles successfully and leads to runtime exception.
- D. The program compiles successfully but does not display anything as output.

**The correct answer is B.**

**Explanation:** The preceding program displays “Alex Richard John” as output as it demonstrates the variable argument list concept. Therefore, option B is correct

### **Question2: what will be the output of given code?**

---

```
class Ques2 {  
    int eval(int[]...vars)  
    {   int sum=0, b, c;  
        for(b = 0; b<vars.length; b++) {  
            for(c=0;c<vars[b].length; c++) {  
                sum += vars[b][c];  
            }  
        }  
        return(sum);  
    }  
    public static void main(String args[])  
    {   Ques2 varargs = new Ques2();  
        int sum =0;  
        sum = varargs.eval(new int[]{10,20,30,40}, new int[]{40,50,60});  
        System.out.println("The sum of the numbers is:" + sum);  
    }  
}
```

**What will happen during compilation and execution of your program?**

- A. The program will compile and display “The sum of the numbers is: 250” as output.
- B. The program will compile and display 25 as output.

C. The program will not compile due to invalid declaration of integer variable arguments.

D. The program will generate the runtime exception.

**The correct option is A.**

**Explanation:** The code will compile and execute successfully displaying “The sum of the numbers is: 250” as output because the eval (int []...vars) method is declared with a variable argument list as its parameter. In the code, a two dimensional array is declared to evaluate the sum and the eval method is invoked from the main method.

**Question3 ( what will be the output of given code)**

---

```
class Ques3{  
    public static void main(String args[]) {  
        int x = 201;  
        myMethod(x++);  
        System.out.println(x);  
    }  
    static void myMethod(int x)  
    {  
        x %= 10;  
        System.out.println(x);  
    } }
```

**What will be output of the above program after compilation and execution?**

A. The program will compile successfully and execute displaying 1 and 202 as output.

B. The program will compile successfully and execute displaying 2 and 202 as output.

C. The program will compile successfully and execute displaying 1 and 201 as output.

D. The program will compile successfully and execute displaying 1 and 1 as output.

**The correct option is A.**

**Explanation:** After compiling and executing the above program 1 and 202 will be displayed as output. In the given program, the Ques12 class contains two static methods, the main () and myMethod (). Each of the static method defines a local variable x, having same name. When the program executes the myMethod () method is invoked and the value, 201 is passed as an argument to the myMethod () method which is assigned to its local variable, x. Then the compound operator performs the modulus operation and the resultant value (1) is displayed. Finally the value of the local variable within the main () method, i.e. 202 (after increment) is displayed.

#### **Question 4**

---

```
public class Ques4
{ public String name;
}
```

Now you realized that to make the name variable as read only for the other classes. Which of the following options are correct to mark the name variable as read only?

- A. You can mark the name variable as private.
- B. You can mark the name variable as private and provide the public method getName() which will return its value
- C. You can mark the name variable as protected.
- D. You can mark the name variable as static and provide the public static method getName() which will return its value

**The correct option is B.**

**Explanation:** In Java, the standard way to provide the read only access to the variables is to mark the variable as private and provide a public method returning its value.

**Question5: Which of the following statements are true based on the use of modifiers?**

---

- A. Local variables can be declared either static or transient.
- B. The visibility of the local variables cannot be specified.
- C. By default the variable is accessible within the same class and subclass of the super class.
- D. The visibility of the local variables is default.

**The correct option is B.**

**Explanation:** The local variables cannot be marked as transient, volatile, and static, Correct option is B and The local variable does not have any accessibility as they are accessible only from the block in which they are declared.

**Question6: Which of the following are valid declarations of the main () method?**

---

- A. static main(String args[]){ }
- B. public static String main(String args[]){...}
- C. public static void main(String args[]) {....}
- D. final static void main(String args[]) {....}

**The correct option is C**

**Explanation:** The following is a valid declaration of the static method:

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

**Question7: Which of the following is the correct higher to lower order of restrictiveness for access specifies?**

---

- A. public> default(within the package)> protected> private
- B. private> default(within the package)> protected> public
- C. private> protected> default(within the package)> public
- D. protected> default(within the package)> private> public

**The correct option is B.**

**Explanation:** The private class members can be accessed only within the class in which they are declared and therefore the private access specifier is highly restrictive. Moreover, the members with default accessibility are accessible within the class in which they are declared and by the classes belonging to the same package. In addition, the protected members are also accessible from subclasses and therefore the protected access specifier is less restrictive as compared to the default accessibility.

**Question 8: Imagine you want to clear your concept of nested classes and so you create a program containing nested and static classes. Consider that you have created the following program?**

---

```
public class Ques8
{
    public static void main(String args[])
    {   TestOuter o = new TestOuter();
        TestOuter.TestInner i = o.new TestInner();
        TestOuter.TestStaticInner inner = new TestOuter.TestStaticInner();
    }
}
class TestOuter {
    static int num1 = 100;
    TestOuter() {
        System.out.print("Welcome to the outer class" + " ");
    }
    class TestInner
    {
        TestInner() {
            System.out.print(TestOuter.num1 + " ");
        }
    }
    static class TestStaticInner {
        static int staticnum = 200;
```

```
TestStaticInner() {  
    System.out.print(staticnum + " ");  
}  
}  
}
```

**What will be the output after you compile and execute the preceding program?**

- A. The program compiles successfully and displays “Welcome to the outer class 100 200” as output.
- B. The program compiles successfully and displays “Welcome to the outer class 200 100” as output.
- C. The program compiles successfully and displays “Welcome to the outer class 100” as output.
- D. The program compiles successfully and displays “Welcome to the outer class 200” as output.

**The correct option is A.**

**Explanation:** The first statement in the main method creates an instance of the TestOuter class and then the second statement instantiates the TestInner class. The instantiation of the TestInner class is done by using the outer class instance as the TestInner class is a non-static class. Finally an instance of the TestStaticInner class is created without using the instance of the TestOuter class as the TestStaticInner class is a static class. As a result the correct option is A.

**Question9: Imagine that you are a Java programmer in the ABC Company and create the following program?**

---

```
public class Ques9  
{ public void myMethod1()  
{ static int num1=100;  
final int num2=200;
```

```
System.out.println("The value of first variable is " + num1);
System.out.println("The value of second variable is " + num2);
}
public void myMethod2()
{   int arr[] = new int[2];
System.out.println(arr[arr.length-1]);
}
public static void main(String args[])
{
new Ques9().myMethod1();
new Ques9().myMethod2();
}
```

**What will be the output after you compile and execute the preceding program?**

- A. The program will lead to compilation errors as static variables cannot be declared within methods.
- B. The program will compile successfully and display “The value of first variable is 100” and “The value of second variable is 200”, as output.
- C. The program will compile successfully and lead to the `ArrayIndexOutOfBoundsException` exception during runtime.
- D. The program will lead to compilation errors as the object arr is not initialized.

**The correct option is A.**

**Explanation:** The static variables cannot be declared within a method and therefore the correct option is A. However the final variables can be declared within a method and all array elements in an integer array are by default initialized to 0. As a result the correct option is A.

**Question10: what will be the output of given code?**

---

```
public class Ques10
{ private static int num1 = 100;
private int num2 = 200;
public static void myMethod1()
```

```
{  num1 = 300;
num2 = 400;
System.out.println(num1 + "," + num2);
}
public static void myMethod2()
{
num1 = 300;
Ques10.num2 = 400;
}
public void myMethod3()
{
num1 = 300;
num2 = 400;
}
public void myMethod4()
{
Ques10.num1 = 300;
num2 = 400;
}
public static void main(String args[])
{
Ques10 q = new Ques10();
q.myMethod1();
}
```

**Now you need to analyze the preceding program and give a feedback to your mentor with explanation. Therefore, which of the following statements you can provide as a feedback to your mentee?**

- A. The program will compile successfully.
- B. The program will lead to compilation error as the non-static variables cannot be referenced from a static context.

- C. The program will compile successfully and lead to runtime error.
- D. The program will compile successfully and display “300,400” as output

**The correct option is B**

**Explanation:** The correct answer is B as it is not possible to access the non-static variable within the static method. Therefore options A, C, and D are incorrect.

**Question 11: Imagine while practicing the concept of primitive variables in Java, you came across the following program ?**

---

```
public class Ques11 {  
    public static void main(String args[])  
    { Ques11 q = new Ques11();  
        q.method(30);  
        byte b = 3;  
        q.method(b);  
    }  
    public void method(Integer i)  
    { System.out.print("Integer value is: " + i + " ");  
    }  
    public void method(short s)  
    { System.out.print("Short value is: " + s + " ");  
    }  
    public void method(byte t)  
    { System.out.print("Byte value is: " + t + " ");  
    }  
    public void method(int num)  
    { System.out.print("Int value is: " + num + " ");  
    }  
}
```

**What will be output of the preceding program?**

- A. The program will display “Int value is: 30 Byte value is: 3” as output.
- B. The program will display “Integer value is: 30 Byte value is: 3” as output.
- C. The program will display “Int value is: 30 Short value is: 3” as output.

D. The program will display “Integer value is: 30 Short value is: 3” as output  
**The correct option is A.**

**Explanation:** In the preceding program, widening is preferred over boxing and therefore while invoking the method () method through the argument 30, it will be widened to call the method (int num).

**Question12: Imagine you are working in the ABC Company and you are assigned a project with a team. Being a team leader you need to analyze the programs created by your team members. While analyzing the programs, you came across the following program:**

---

```
public class Ques12
{
    public static void main(String args[])
    {
        Ques12 q = new Ques12();
        q.myMethod (10,20);
        q.myMethod (new long[]{} );
        q.myMethod (new int[]{10,20});
    }
    void myMethod (short s1, short s2)
    {
        System.out.println ("short");
    }
    void myMethod (int i1, int i2)
    {   System.out.println ("int");
    }
    void myMethod (int ...args)
    {
        System.out.println ("intargs");
    }
}
```

**Which of the following statements are justified in the context of the preceding program?**

- A. The program will compile successfully and display “int intargs intargs” as output.
- B. The program will lead to compilation error.
- C. The program will compile successfully but lead to runtime exception.
- D. The program will display “short intargs intargs” as output.

**The correct option is B.**

**Explanation:** The program leads to compilation errors as while invoking the q.myMethod(new long[]{}{}) method, implementation of the myMethod() method does not have the long array type argument. Therefore the option B is the correct answer.

**Question13: Imagine you write the following program while understanding the concept of primitive variables?**

---

```
public class Ques13
{
    public static void main(String args[])
    {
        System.out.println(myMethod(myMethod(new int[]
{10,20}),myMethod(10,20)));
    }

    static int myMethod(int num1, int num2)
    {
        return 10;
    }

    static int myMethod(int... args)
    {
        return 20;
    }
}
```

**What will be output of the preceding program?**

- A. The program will compile successfully and display 10 as output.
- B. The program will lead to compile time error as the myMethod with int [], int [] argument is not defined.
- C. The program will compile successfully but lead to runtime exception.
- D. The program will compile successfully and display 20 as output

**The correct option is A.**

**Explanation:** In the preceding program, the code myMethod(new int[] {10,20}) invokes the myMethod(int...args) which returns 20. Moreover the code

myMethod(10,20) invokes the myMethod(int num1, int num2) method which returns 10. Finally the myMethod(myMethod(new int[] {10,20}), myMethod(10,20)) method becomes myMethod(20,10) which invokes myMethod(int num1, int num2), thereby returning 10, which is displayed. Therefore the correct answer is A.

**Question14: Imagine you are a Java programmer and you have created the following program?**

---

```
public class Ques14 {  
    public static void main(String[] args)  
    {  
        System.out.println (myMethod (new double[]{10, 20, 30}));  
        System.out.println (myMethod (new Double[]{10d, 20d, 30d}));  
        System.out.println(myMethod(10, 20, 30));  
        System.out.println(myMethod());  
    }  
    static double myMethod(double ... args)  
    {  
        double total = 0;  
        for (double temp : args) {  
            total += temp;  
        }  
        return total;  
    }  
    static double myMethod(Double ... args)  
    {  
        double total = 2;  
        for (double temp : args) {  
            total *= temp;  
        }  
    }
```

```
return total;  
}  
}
```

**What will be output of the preceding program?**

- A. The program will lead to compilation error.
- B. The program will compile successfully and display “60.0 12000.0 60.0” as output.
- C. The program will compile successfully but lead to runtime error.
- D. The program will compile successfully and display “60.0 60.0 12000.0” as output.

**The correct option is A.**

**Explanation:** The program will lead to compilation error while invoking the myMethod() method without any argument. This is because of the ambiguity caused due to declarations of the myMethod taking primitive double variable argument and no declaration of the MyMethod without any argument.

**Question15: Imagine being a Java programmer you write the following program**

---

```
public class Ques15  
{ String str;  
int i=10;  
static void myMethod() {  
System.out.println("The value of String variable is" + new  
Ques15().str.length());  
}  
public static void main(String args[]) {  
myMethod();  
}  
}
```

**Which of the following statements are true in the context of the preceding program?**

- A. The program will lead to compilation error as a non–static variable cannot be accessed from static context.
- B. The program will compile successfully but lead to runtime exception.
- C. The program will lead to compile time error as the String variable str is not assigned a value.
- D. The program will compile successfully and print 4 as output.

**The correct option is B.**

**Explanation:** The String variable str is not assigned a value so the default value null will be assigned. The myMethod() is a static method and the str variable is a non static variable, therefore the Ques15 class reference is used to access the str variable. The compiler will compile the program successfully as the default value null will be assigned to the str variable. However during execution the NullPointerException exception occurs as the length function is invoked on the str variable which is not initialized.

**Question16: Imagine you are a faculty in an institute and you have explained the concept of Inner classes to the students. While practicing the students created the following program and you were asked to analyze the program?**

---

```
public class Ques16
{
    void myMethod()
    {
        System.out.println("Welcome to the world of programming");
    }
}

class MyNest
{
    public static void main(String args[])
    {
        Ques16 q = new Ques16();
        q.myMethod();
    }
}
```

**What will be output of the preceding program?**

- A. The program will compile successfully and print “Welcome to the world of programming” as output.

- B. The program will compile successfully but lead to runtime error.
- C. The program will lead to compilation error.
- D. The program will compile successfully but no output is displayed

**The correct option is C.**

**Explanation:** The inner classes cannot have static declarations and so the preceding program will lead to compilation error.

Therefore the correct option is C.

**Question17: Sam works in Xyz Company as Java programmer and he designed the following program?**

---

```
class Rose
{
    public void sam()
    {
        int y[] = {4, 2, 8};
        for (int x=2; x<1+3*2-4; x++){
            System.out.print(x+" ");
            for (int j:y) {
                j=j*x-4;
                System.out.print(j+" ");
            }
        }
    }
    public static void main(String[] args)
    {
        Rose r = new Rose();
        r.sam();
    }
}
```

What would be the output of this program? Choose the correct option from the following:

- A. The program displays 2 4 2 8
- B. The program displays 2 4 0 12
- C. The program displays 2 4 4 16
- D. The program displays 3 4 0 12

**Correct option is B.**

**Explanation:** Correct option is B. Firstly the value of the variable x is 2, which is less than the 3 (value specified in expression) therefore control will transfer into for each loop and prints the array variable with modifications according to the expression. Array element is fetched into j. First array element is 4 and the

Expression is  $j*x-4$  i.e.  $4*2-4=4$ . In the same way, other array elements are Extracted and displayed with modifications.

**Question18: what will be the output of given code?**

---

```
public class Rose
{ protected void get(boolean x )
{
if(x)
{ System.out.println("True");
}
else
{ System.out.println("False");
}
}
public static void main(String[] args)
{
Rose r = new Rose();
r.get(true);
}
}
```

**What would be the output when the program is compiled?**

- A. Program will display True
- B. Program will display False
- C. Program will successfully compile but give runtime error
- D. Program will not compile successfully

**Option A is the correct answer**

**Explanation:** Option A is the correct answer because the value true is passed in the get () method from main method. In the get () method, the if expression evaluates to true and statement written in it is displayed. Therefore, the Option B is incorrect. Option C and D are incorrect because the program will successfully compile and execute.

**Question19: what will be the output of given code?**

```
public class Rose
{
protected void get(char x )
{
switch(x)
{ case 88: System.out.println( "X");break;
case 90: System.out.println( "Z");break;
case 89: System.out.println( "Y");break;
default: System.out.println( 0);break;
case -97: System.out.println("a");break;
}
}
public static void main(String[] args)
{
Rose r = new Rose();
r.get('X');
}
}
```

What would be the output of this program?

- A. Program will display X
- B. Program will display X0
- C. Program will not compile successfully
- D. Program will compile successfully but not execute

**Option C is the correct answer.**

**Explanation:** Option C is the correct answer because switch case is designed to accept character values but - 97 is an integer value. Therefore, options A, B, and D are incorrect.

**Question20: Sam as a developer in GIRI'S TECH HUB created the following program:**

---

```
class Rose {  
    static int j;  
    public int arr()  
    {  
        int y[] = { 5 , 7, 8 , 6};  
        j = y[2]; return j;  
    }  
    public static void main (String args[])  
    {  
        Rose r=new Rose();  
        int x = r.arr( );  
        System.out.println(x);  
        switch(x)  
        {  
            case 0: System.out.print(0 + " ");break;  
            case 2: System.out.print(2 + " ");break;  
            case 8: System.out.print(8 + " ");  
            case 5: System.out.print(5 + " ");break;  
            default: System.out.print("Default");  
        }  
    }  
}
```

**What would be the output when Sam compile and execute this program?**

- A. Program will not compile successfully
- B. Program will display 8 8 5
- C. Program will display 8 5
- D. Program will display 8

**Option B is the correct answer.**

**Explanation:** Option B is the correct answer. Firstly the control will transfer to arr () method and retrieves the array element at array [2] index and returns that value to calling routine. In the calling routine the retrieved

value is first displayed and then used in switch statement. Therefore, the result is 8 8 and 5 is displayed because the case 8 does not have break and 5 is the statement written in next case. Therefore, options C and D are incorrect. Option A is incorrect because program will successfully compile.

### **Static Keyword in Java**

---

Static keyword in java is used for Memory Management only. We can apply static keyword with variables, methods, block and nested classes. The static belongs to the class than instance of class.

#### **The static can be:**

1. Variables (also known as class variable)
2. Method (also known as class method)
3. Block
4. Nested class

#### **Now we will discuss about the Static variables in Java**

---

1) Static variables means variable can allocate its memory before creating object of class in PREMGEN space before JDK 8 and After JDK it is allocated in METASPACE and non static variable of class allocate its memory after object of class in heap section of memory.

#### **What is the PREMGEN or METASPACE**

---

**PREMGEN** stands for **Permanent Generation**. It is a special type of heap space. It is separate from the main memory (heap). JVM uses **PREMGEN** to keep track of loaded class metadata. All the static content is stored by the JVM to this memory section. Static content can be a static method, references to the static object and primitive variables. **PREMGEN** also contains information about byte code, names, and JIT. Before releasing java

7, String Pool is also available in this space and separate from it after releasing Java 7.

**The JVM memory is divided into two parts, which are as follows:**

1. PREMGEN Space (Permanent Generation)
2. Heap Space

Eden Space (Young Generation)

Survivor Space (Young Generation)

Old Generation

The **PREMGEN** space contains the internal representation of the Java classes that JVM holds. The Permanent Generation is the garbage data that is collected in the same way as heap's other parts collected. It is a special area of memory that contains meta-data of the program's classes and the program's objects. It also contains the class-loaders that are manually destroyed at the end of the use. Here, the Garbage Collector is not that efficient and due to which it causes the Out of Memory: PREMGEN space error quite a few times.

The main disadvantage of the **PREMGEN** space is that its maximum size is fixed. The 64 MB and the 82 MB are the maximum memory size for 32-bit and 64-bit version of JVM, respectively.

### **How to declare the static variables in java**

---

**Syntax:** static datatype variablename;

e.g. static int a=100;

2) static variable allocated memory before creating object of the class by class loader so we can access the static variable by using class name and instance variable or non static variable allocated by object by JVM so it is accessible by object. membername.

e.g class ABC

```
{ static int m=100;  
int n=200;
```

}

We can use the m using classname given below

System.out.println (ABC.m);

We can use the non static variable n by using objectname given below

ABC ab = new ABC ();

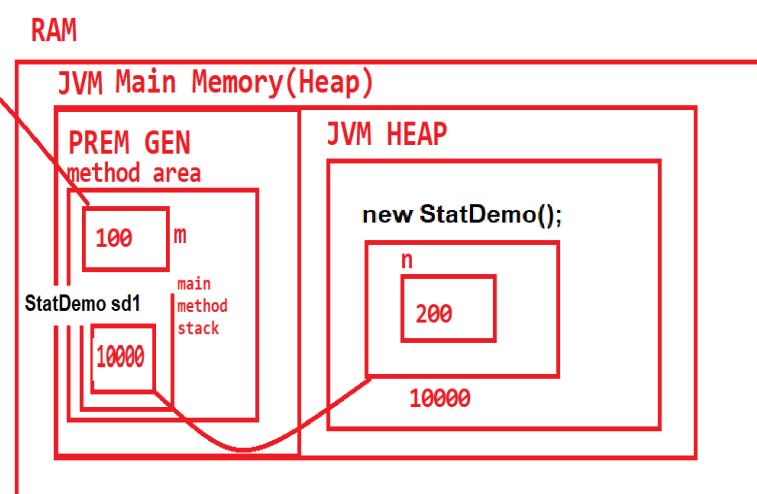
System.out.println ("N is "+ab.n);

### **Example of Static and non static variables with memory structure**

```
class StatDemo
{
    static int m=100;
    int n=200;
}
public class StatDemoApp
{
    public static void main(String x[])
    {
        System.out.println("M is "+StatDemo.m);

        StatDemo sd1 = new StatDemo();

        System.out.println("N is "+sd1.n);
    }
}
```



If we think about the above diagram we have the two variables declared in class StatDemo name as m and n. Here we declare m variable as static and n variable as non static i.e. instance variable so if we see the above diagram static variable m stored in PREMGEN memory area before creating object of class by the class loader so we use it by using classname.membername and n declared as non static so n allocated memory after creating object means n variable stored in object in JVM main heap memory so we use the variable n by using object name.

### **Output**

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StatDemoApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java StatDemoApp
M is    100
N is    200

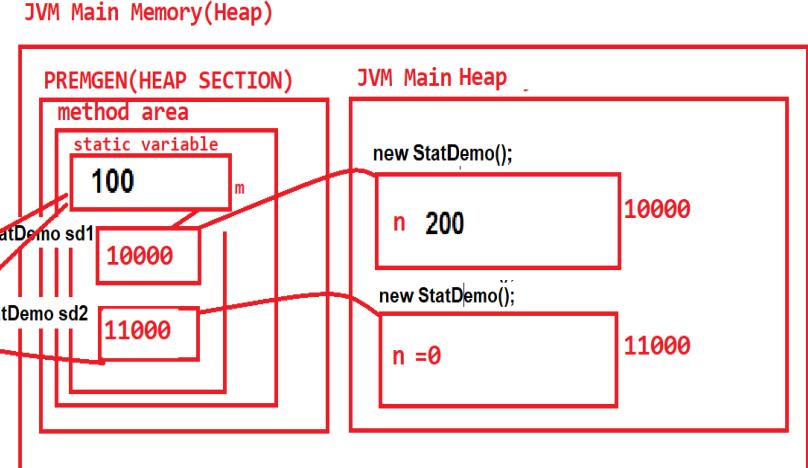
C:\Program Files\Java\jdk1.8.0_291\bin>
```

3) Static variable allocate share its common copy between more than one object of same class and non static variable share its separate copy for every object of class.

**Note:** we can access static variable by using class name as well as by using object but non static variable only accessible by using object name.

### **Following Example demonstrate the above concept**

```
class StatDemo
{
    static int m;
    int n;
}
public class StatDemoApp
{
    public static void main(String x[])
    {
        StatDemo sd1 = new StatDemo();
        StatDemo sd2 = new StatDemo();
        sd1.m=100;
        sd1.n=200;
        System.out.println("With First Object");
        System.out.println("M is "+sd1.m);
        System.out.println("N is "+sd1.n);
        System.out.println("With Second object");
        System.out.println("M is "+sd2.m);
        System.out.println("N is "+sd2.n);
    }
}
```



**Output**

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StatDemoApp.java
With First Object
M is 100
N is 200
With Second object
M is 100
N is 0
```

If we think about the above diagram we have the two variables name as m and n in class name as StatDemo we declare variable m as static and n as non static or instance variable so as per the diagram variable m stored in PREMGEN memory area before creating object of the class so we have the statement name as StatDemo sd1 =new StatDemo() here we create the object in heap section and sd1 as reference then variable n allocated in object whose address is 10000 mention in diagram and when we have the another statement name as StatDemo sd2=new StatDemo() then we create the one more object in heap section and its reference sd2 stored in method area the address of second object is 11000 in diagram. We have the statement sd1.m=100 means first object name as sd1 use the static variable m and store the 100 value in static variable m and when we have the one more statement name as sd1.n means use the variable n which is present in object whose address is 10000 and we store the value 200 in variable n and we have the Statement name as System.out.println("with first object") and System.out.println("M is "+sd1.m) and System.out.println("N is "+sd1.n); So we have the Output

With First Object

M is 100

N is 200

After that we have the statement System.out.println ("With second object");

System.out.println ("M is "+sd2.m) this statement access the value of m so it will print 100 because variable use the last existing value and we store the value in static variable m by object sd1 so m contain the value 100 stored by sd1 and static variable copy is common for all objects so sd2.m print the 100 value to us

But when we use the statement System.out.println ("N is "+sd2.n) then this statement use the value of n which is present second object whose address is 11000 in out diagram but we not store the any value in variable n present in object second whose address is 11000 so by default it contain 0 value so we have the output like as

With Second object

M is 100

N is 0

#### **4) The default values of static and non static/instance variables provided by Java if user not provide value**

Data Type	Default Values
int	0
float	0.0
double	0.0
long	0
String	null
boolean	false
byte	0

#### **Following Example demonstrate the above concept**

```
File Edit Format View Help
class DefVal
{
    boolean b;
    String s;
    int d;
    void show()
    { System.out.println("Default value of boolean is "+b);
        System.out.println("Default value of string is "+s);
        System.out.println("Default value of integer is "+d);
    }
}
public class VarDefValues
{
    public static void main(String x[])
    {
        DefVal dv = new DefVal();
        dv.show();
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac VarDefValues.java
C:\Program Files\Java\jdk1.8.0_291\bin>java VarDefValues
Default value of boolean is false
Default value of string is null
Default value of integer is 0

C:\Program Files\Java\jdk1.8.0_291\bin>mspaint
```

If we think about the above code then we declare the three variables name as Boolean a ,String s and int d and we not provide the any values or

initialized value in it manually and we have one more function name as show() and in this function we print the values of all three values and we create the object of DefVal class in main method and call the show() function then this function display the values of all variables so it will print the default values provided by java so b is Boolean so its default value is false,s is string so its default value is null and d is integer so its default value is 0.

**5) Life of static variable is present when ever application running in memory and the life of non static/instance variable is present when ever object running in memory.**

---

As per the above statement we need to find the life of object before finding the life of non static /instance variable. Object present in memory when ever object use by any reference when object not use by any reference then JVM automatically delete the memory of object called as Garbage Collection and all instance variable of class present in Object so when JVM release or delete the memory of object then automatically all instance variable get deleted but the static variable not present in object they present in PREMGEN space so JVM cannot delete the static variable when delete the object of class so static variable present in memory when ever program running in memory and another reason static variable persist in memory at application level because they are common in more than one object of class.

**The Following code example demonstrate the above concept**

---

```

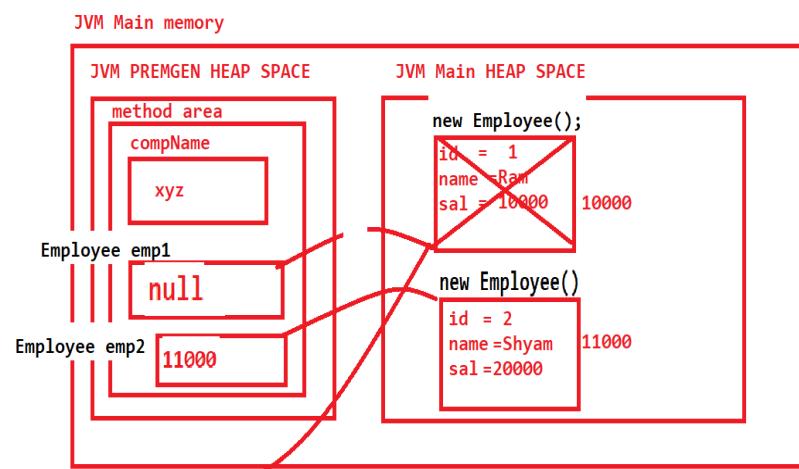
class Employee
{
    static String compName="XYZ";
    int id;
    String name;
    int sal;
}

Employee emp1 = new Employee();
emp1.id=1;
emp1.name="Ram";
emp1.sal=10000;

Employee emp2 = new Employee();
emp2.id=2;
emp2.name="Shyam";
emp2.sal=20000;

emp1 = null;

```



If we think about the code description we have the class name as Employee with four fields `compName` , `id` , `name` , `sal` here `compName="XYZ"` is a static variable so it is allocated in PREMGEN space before creating object of class and we have the two objects Employee `emp1 = new Employee()` and Employee `emp2=new Employee()` so here `emp1` having address 10000 and `emp2` having address 11000 so in `emp1` we store the data `id=1 ,name="Ram"` and `sal=10000` and in `emp2` we store data `id=2 name="Shyam"` and `sal=20000` when `emp1=null` then `emp1` not points to object whose address is 10000 as per our example whose `id` is 1 `name` is ram and `sal` is 10000 means `emp1` object not use by any reference of JVM delete the memory of `emp1` object so it contain three non static variable `id` ,`name` and `sal` so these variables automatically get deleted when JVM delete its object but `compName` is not deleted because it is not present in object so `compName` present in memory when ever your application or program running in memory

So the life of static variable is present up to application level so some people say static variables are application variable or class level variable also.

## **What is the garbage collection?**

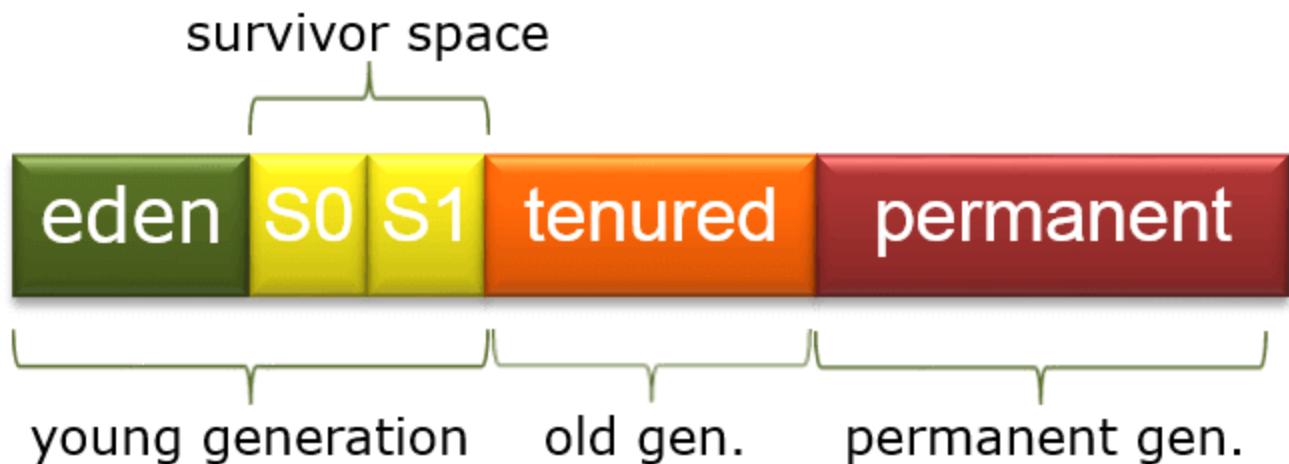
Java garbage collection is the process by which Java programs perform automatic memory management. Java programs compile to byte code that can be run on a Java Virtual Machine or JVM for short. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.

## **How Java Garbage Collection Works**

Java garbage collection is an automatic process. The programmer does not need to explicitly mark objects to be deleted. The garbage collection implementation lives in the JVM. Each JVM can implement garbage collection however it pleases; the only requirement is that it meets the JVM specification. Although there are many JVMs, Oracle's HotSpot is by far the most common. It offers a robust and mature set of garbage collection options.

While HotSpot has multiple garbage collectors that are optimized for various use cases, all its garbage collectors follow the same basic process. In the first step, unreferenced objects are identified and marked as ready for garbage collection. In the second step, marked objects are deleted.

Optionally, memory can be compacted after the garbage collector deletes objects, so remaining objects are in a contiguous block at the start of the heap. The compaction process makes it easier to allocate memory to new objects sequentially after the block of memory allocated to existing objects. All of Hotspot's garbage collectors implement a generational garbage collection strategy that categorizes objects by age. The rationale behind generational garbage collection is that most objects are short-lived and will be ready for garbage collection soon after creation.



**The heap is divided into three sections:**

- **Young Generation:** Newly created objects start in the Young Generation. The Young Generation is further subdivided into an Eden space, where all new objects start, and two Survivor spaces, where objects are moved from Eden after surviving one garbage collection cycle. When objects are garbage collected from the Young Generation, it is a minor garbage collection event.
- **Old Generation:** Objects that are long-lived are eventually moved from the Young Generation to the Old Generation. When objects are garbage collected from the Old Generation, it is a major garbage collection event.
- **Permanent Generation:** Metadata such as classes and methods are stored in the Permanent Generation. Classes that are no longer in use may be garbage collected from the Permanent Generation.

---

#### **MCQ Questions (static variables)**

---

#### **Question1: What will be the output of given code?**

---

```
public class Time {  
    int a = 50;  
    int b = 10;
```

```
public static void main (String args[]) {  
    a += b--;  
    System.out.println (a);  
}  
}
```

### **Options**

---

- a) 60
- b) 50
- c) 59
- e) Compilation Error

**Ans:** e (compilation Error – non static variable a cannot be reference from static context)

**Explanation:** we cannot use the non static variable in static function means as per our example we have the two variables name as a and b these declared as instance variable or non static and we have the function name as public static void main (String args []) this is the static function and we cannot use the non static variable a and b in static function so this is the major reason compiler generate the error to us non static variable a cannot reference from static context.

### **Question2: What will be the output of given code?**

---

```
class W  
{ static int c = 0;  
public static void main(String[] args)  
{ W w1 = c();  
W w2 = c(w1);  
W w3 = c(w2);  
W w4 = c(w3);  
}  
private W()
```

```
{  
    System.out.println("c = " + c);  
}  
static W c()  
{  return c++ <= 0 ? new W() : null;  
}  
static W c(W w)  
{  return w.c++ == 1 ? new W() : null;  
}  
}
```

### **Options**

---

- a) c = 1  
c = 2
- b) c = 1  
c = 2  
c = 3
- c) c = 1  
c = 2  
c = 3  
c = 4
- d) Compilation Error

**Ans:** option a

C=1

C=2

**Explanation:** if we think about the above code we have the one private default constructor name as W () and we have the static overloaded method name as C () which return the reference of W class.

If we think about the main () method in main method we have the statement

W w1= C (); means we call the C() method version which has no parameter and in this method we have the statement return c++ <=0 ?new W():null

here c++ is post increment so post increment having less priority than <= means here <= get executed before ++ means variable c compare before increment so the statement like as `0 <=0 ?new W():null` so `0<=0` is true so first condition check then ++ get executed means c++ execute means the value of c change from 0 to 1 means the value of c is 1 and then new W() option get executed so your default constructor executed and print the output `c=1` and in main method we have the one more statement `W w2=C(w1)` this statement get executed means C method get executed in which parameter is present means static `W c(W w)` function get executed in this function contain the statement return `w.c++ ==1 ? new W(): null` here `w.c++==1` first execute but here == having higher priority than ++ so `w.c==1` get compare first the value of c was 1 before increment so this statement get satisfy after comparison `w.c++` get execute so value c is incremented by 1 means the value of c was 1 before increment and after increment c is 2 and then new W() statement get executed means your default constructor get executed and print the value c it is 2 so our output is `C=1` and `C=2` and so on .

### **Question3: What will be the output of given code?**

```
class Output
{ final static short i = 2;
public static int j = 0;
public static void main (String [] args)
{
for (int k = 0; k < 3; k++)
{
switch (k)
```

```
{  
case i: System.out.print(" 0 ");  
case i-1: System.out.print(" 1 ");  
case i-2: System.out.print(" 2 ");  
}  
}  
}  
}
```

### **Options**

---

- a) 2 1 0 1 0 0
- b) 0 1 2 1 2 2
- c) 2 1 2 0 1 2
- d) 0 1 2

**Ans: option C (2 1 2 0 1 2)**

### **Explanation**

---

If we think about the above code we have the three variable declared in program i , j , k the initial value of i=2 and j=0 and k=0 we use the k in for loop means our statement is

```
for(k=0; k<3; k++){  
switch(k)  
{ case i: S.o.print("0");  
case i-1: S.o.print("1");  
case i-2: S.o.print("2");  
}  
}
```

Here initially k=0; so k<3 get satisfy and we pass k in switch switch(k) means switch(0) so we have the case i means case 2 not match case i-1 means case 2-1 means case 1 one match we have the third statement case 2-I means 2-2 means case 0 get satisfy and we get the result 2 first and when loop get executed second time then k++ happen then value of k is 1 so k<3 also satisfy this value of k pass in switch(k) means switch(1) so our

case i means case 2 not satisfy case 2-i means 2-1 means case 1 get satisfy so it will print 1 and we not use the break state after case so second case also get executed and print 2 means we have the output 2 1 2 after k++ happen so value of k=2 means k<3 this statement get satisfy so in switch we have the switch(k) means switch(2) case i means case 2 get executed then all three cases executed because we not use the break in case so our output is 0 1 2 so final output 2 1 2 0 1 2

### **Static block in java**

---

#### **Important points related with static block.**

- 1) Static block initialize only once in program
- 2) Static block execute very first in program even static block execute before main function also.
- 3) We cannot use the non static variable in static block.

#### **Example of Static Block**

---

```
public class Test
{
    static{
        System.out.println("I am static block");
    }
    public static void main(String x[])
    {
        System.out.println("I am in main methhod");
    }
}
```

C:\Program Files\Java\jdk1.8.0\_291\bin>java Test
I am static block
I am in main methhod

C:\Program Files\Java\jdk1.8.0\_291\bin>

If we think about the above code we have the class name as Test in this class we have the one static block and one main method when we run the program then static block get executed before main function and after that main function get executed means a static block get execute very first before main function also.

#### **What is the purpose of static block?**

---

Static block normally use in program when we want to load the code only once in application at the time of application loading

**Example:** suppose we want to load the configuration from xml at the time of application loading.

**Question1: what will be the output of given code?**

---

```
class B {  
    static int i;  
    static int j;  
    static {  
        i = 15;  
        j = i - 5;  
    }  
  
    public static void main(String[] args) {  
        int i = 0;  
        A a = null;  
        for (; i < 3; i++) {  
            a = new A();  
            a.i = B.i;  
            B.i += a.add(a.operate(i));  
        }  
        System.out.println(B.i + " " + B.j + " " + i + " " + a.i);  
    }  
}  
  
class A {  
  
    int i = 0;  
  
    int operate(int i) {  
        if (B.j - i == i * i * i) return -i;  
        return i * i;  
    }
```

```
}
```

```
int add(int i) {  
    return this.i + i;  
}
```

### **Options**

---

- a) 14 10 3 0
- b) 14 10 3 16
- c) 120 10 3 61
- d) Compile Time Error

**Answer:** Option C (120 10 3 61)

**Explanation:** I will explain in class room ( and I will share its video in app);

### **Question2: what will be the output of given code?**

---

```
class Increment  
{  
    public static void main(String[] args)  
    {  
        Simple s = new Simple();  
        Simple r = new Simple();  
        s.incr();  
        r.incr();  
        s.display();  
        r.display();  
    }  
}  
  
class Simple{  
    static int a = 5;  
    void incr()  
    {  
        a++;  
    }  
    void display()  
    {  
        System.out.println("a = " + a);  
    }  
}
```

```
}
```

### Options

---

- a) a = 5  
a = 5
- b) a = 6  
a = 6
- c) a = 7  
a = 7
- d) Compilation Error

### Output: Option C (a=7 a=7)

### Explanation

---

In this example we have the two classes Increment and Simple in Simple class we have the static variable name as and its initial value is 5 means a=5 and we have the two function name as incr() and display() in incr() function we increase the value of a by 1 and display function we display the value of variable a .

In Increment class we have the statements

```
Simple s = new Simple();
```

```
Simple r = new Simple();
```

When we call s.incr() method then value of variable a incremented by 1 means a=6 and we have the one more statement r.incr() then again value of a is incremented by 1 means a=7 and variable use the last existing value Note: static variable share its common copy between multiple objects of same class means s and r objects or references use the last of value a i.e 7 means when we call the s.display() and r.display() then we get the output of a=7 with s object and a=7 with r object so our output is a=7 and a=7

### Question3: what will be the output of given code

---

```
public class Cricket {
```

```
static boolean ball;
public static void main(String[] args) {
    short bat = 42;
    if (bat < 50 & !ball)
        bat++;
    if (bat > 50)
        ;
    else if (bat > 40) {
        bat += 7;
        bat++;
    }
    else
        --bat;
    System.out.println(bat);
}
```

---

**Options**

- a) 41
- b) 42
- c) 51
- d) Compilation Error or Runtime Error

**Ans: Option C (51)**

**Explanation: I will solve this problem in class room**

---

**Question4: what will be the output of given code?**

```
public class GuessCondition
{
    static int a = 40;
    public static void main(String args[])
{
```

```
System.out.print(a + " ");
add();
System.out.print(a);
}
private static void add()
{
a = a + 40;
}
```

**Options**

---

- a) 40 40
- b) 0 0
- c) 0 40
- d) 40 80

**Correct Ans: Option C (40 80)**

**Explanation:**

As per our code we have the class name as GuessCondition in this class we have the two functions name as public static void main(String args[]) and add() and we declare the one static variable name as a=400

In main we have the statement System.out.print(a+" "); this statement print the value of a i.e 40 and when we call the statement name as add() means we call the add function so your control jump from main method to add function definition and add function we have the statement a=a+40 means we increase the value a by 40 so the current value of a is 80 we print the value of variable a after calling point of add() so it will print the value of a 80 so we have the output is 40 80

**Question5: what will be the output of given code ?**

---

```
public class Student {
int rollno;
String name;
static String college = "RITA";
```

```
static void chage() {  
    college = "SRIT";  
}  
Student(int r, String n) {  
    rollno = r;  
    name = n;  
}  
void display() {  
    System.out.println(rollno + " " + name + " " + college);  
}  
public static void main(String arts[]) {  
    Student.chage();  
    Student s1 = new Student(516, "Kiran");  
    Student s2 = new Student(560, "Vishwanath");  
    Student s3 = new Student(517, "Kranthi");  
    s1.display();  
    s3.display();  
    s2.display();  
}
```

### **Options**

---

- a) 516 Kiran SRIT  
516 Kiran SRIT  
516 Kiran SRIT
- b) 516 Kiran SRIT  
517 Kranthi SRIT  
560 Vishwanath SRIT
- c) 516 Kiran RITA  
516 Kiran RITA  
516 Kiran RITA
- d) 516 Kiran RITA  
517 Kranthi RITA  
560 Vishwanath RITA

**Correct Answer : Option B**

**516 Kiran SRIT**

**517 Kranthi SRIT**

**560 Vishwanath SRIT**

**Explanation:**

As per our code we have the class name as Student with three fields rollno ,name and college we initialize the with default value name as RITA and in this class we have the one static function name as change() and in this function we change the value of college=SRIT and if we think about the main method of class in main method we call the function change using classname because it is a static means we call change using Student.change() when we call this statement then college name change from RITA to SRIT

In main method we have the following statement

**Student s1 = new Student(516, "Kiran"):** using this statement we initialize the object s1 with 516 as rollno and name as kiran

**Student s2 = new Student(560, "Vishwanath"):** using this statement we initialize the object s2 with 560 as rollno and vishwanath as name

**Student s3 = new Student(517, "Kranthi"):** using this statement we initialize the object s3 using 517 as rollno and Kranti as name

**Note:** college is a static and we change its value from RITA to SRIT so the value of college is SRIT and it is static variable so it is common in all objects of class

So when we call s1.display(), s3.display(); , s2.display(); we get the value of every object and college name common with all objects.

---

**Question 6: what will be the output of given code?**

```
public class Karbon {
```

```
    static int i = 5;
```

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

```
        GIRI'S TECH HUB PVT.LTD PUNE – 9175444433, 9049361265
```

```
Karbon test = null;  
System.out.print("i = " + test.i + ", ");  
System.out.print("i = " + Karbon.i);  
}  
}
```

### **Options**

---

- a) i = 0, i = 5
- b) i = 5, i = 5
- c) Compilation Error
- d) NullPointerException

### **Explanation:**

If we think about the above code we have the class name as Karbon and in this class contain the variable name as i and it is static variable and we initialize the value of variable i=5 and i allocate its memory before creating object of Karbon class means we can use the i without object of Karbon class so if we think about the main method then we have the following statement in main method

Karbon test =null; here we create the reference of Karbon class we not create its object so we can access the static variable by using reference of class also

So we have the statement in main System.out.print("I= "+test.i+"," ); this statement print the I=5 and we have the one more statement in main function name as System.out.print("I =" +Karbon.i); this statement print the value of i=5 because we can use the static variable by using classname also so we have the output like as I=5 I=5

### **Question7: what will be the output of given code?**

---

```
public class CanYouDolt  
{ static boolean bool;  
static int[] iary = new int[1];  
static char chr;
```

```
static boolean[] barr = new boolean[1];
public static void main(String args[]) {
boolean b = false;
if (bool) {
b = (chr == iary[chr]);
} else {
b = (barr[chr] = bool);
System.out.println(b + " " + barr[chr]);
}
}
```

### **Options**

---

- a) false false
- b) true true
- c) false true
- d) true false

**Ans: option a (false ,false)**

**Explanation: I will explain code in class room**

### **Question 8: what will be the output of given code?**

---

```
public class Sketch {
static int ad = 100;
static int bc = 200;
static {
ad += 1;
bc += 1;
}
public static void main(String args[]) {
ad += 5;
bc += 10;
System.out.println(ad + bc);
}
```

```
static {
ad += 100;
bc += 200;
}
}
```

### **Options**

---

- a) 317
- b) 615
- c) 617
- d) 315

**Ans: option b (617)**

**Explanation: I will explain in classroom.**

### **Question 9: what will be the output of given code?**

---

```
public class KeepItClean {
public static void main(String[] args) {
Valuable.status();
try(Valuable v = new Valuable(); Valuable v2 = new Valuable())
{
Valuable.status();
v.open();
Valuable.status();
v2.open();
Valuable.status();
}
Valuable.status();
}
}

class Valuable implements AutoCloseable {
static int numberofValuables = 5;
Valuable() {
open();
}
```

```
}

public void open() {
    numberOfValuables--;
    System.out.print("O" + numberOfValuables + " ");
}

public void close() {
    numberOfValuables++;
    System.out.print("C" + numberOfValuables + " ");
}

static void status() {
    System.out.print("S" + numberOfValuables + " ");
}
```

### **Options**

- a) S5 S5 O4 S4 O3 S3 S3
- b) S5 S5 O4 S4 O3 S3 C4 c5 S5
- c) S5 O4 O3 S3 O2 S2 O1 S1 S1
- d) S5 O4 O3 S3 O2 S2 O1 S1 C2 C3 S3

**Ans: Option D (S5 O4 O3 S3 O2 S2 O1 S1 C2 C3 S3)**

### **Local Variable In Java**

Local variable means variable declared within block or function or method called as local variable

e.g

```
class A
{
    void setValue(int x) //x is also local variable
    { int m; //local variable
    }
}
```

**If we want to work with local variable in java we have the some important points**

### 1) Local variable cannot access outside of his block

```
class A
{
    void setValue(int x,int y)
    {
    }
    void showAdd()
    {
        System.out.printf("Addition is %d\n",x+y);
    }
}
```

local variable

Here System will generate the  
compile time error

Note: Here x and y are the local variable in setValue x and y and we try to access the x and y in showAdd() function so it is not possible to access the local variable x and y outside of setValue() function so we have the compile time error.

Here if we want to solve this problem we have to copy the content of x and y Local variable values in any another instance variable and use in all class function shown in following diagram

```
class Value
{ int a,b;
  void setValue(int x,int y)
  {
    a=x;
    b=y;
  }
  void showAdd()
  {
    System.out.printf("Addition is %d\n",a+b);
  }
}
```

Note: Here we have the two variables a and b is instance variable and x and y local variable so local variable x and y cannot access outside of setValue() function but we store its values in instance variable a and b so we can use the value of x and y any all function on Value class through the a and b variable.

## **2) Local variable cannot have default values even garbage in java**

---

The above statement indicates if we want to use the any local variable we must have to initialize some value in it and if we not initialize the value in it then system generate the compile time error to us.

### **Example**

---

```
class ABC
{
  void show()
  {
    int no;
    System.out.println("No is "+no);
  }
}
```

Note: Here System generate the compile time error to us because no is local variable we must be initialize it before use it.

If we want to avoid the compile time error here we have to initialize the some value in local variable no shown in following diagram.

```
class ABC
{
    void show()
    {
        int no=5;
        System.out.println("No is " +no);
    }
}
```

Note: Here System generate the compile time error to us because no is local variable we must be initialize it before use it.

### 3) Local variable cannot declared as static

Because static variable life present when ever application running memory and local variable life present when ever block or function present in memory so we cannot declare the local variable as static.

### 4) Cannot apply any access specifier on local variable

---

— Means we cannot declare the local variable as private, protected and public

### 5) If Local variable name and instance variable name is same then instance variable not use in block in which local variable name is same

---

```
class Add
{
    int x,y;
    void setValue(int x,int y)
    {
        x=x; _____ Note: Here instance variable name and local variable
        y=y;             name is same so instance variable never use in block
    }                   in this block so local variable not copy its
    void showAdd()     value in instance variable
    {
        System.out.printf("Addition is %d\n",x+y);
    }
}
public class AdditionApp
{
    public static void main(String x[])
    {
        Add ad = new Add();
        ad.setValue(10,20);
        ad.showAdd();
    }
}
```

Here instance variable get use  
but the default value on instance variable  
is 0  
so we have the output  
**Addition is 0**

If we want to use the instance variable in block in which local variable name is same then we have to use this reference.

#### **Q. what is this reference in java?**

this is internal reference present in every class which is used to point to current running object in java this normally use when we want to call the same class member from its own definition as well as when we have the local variable name and instance variable name is same.

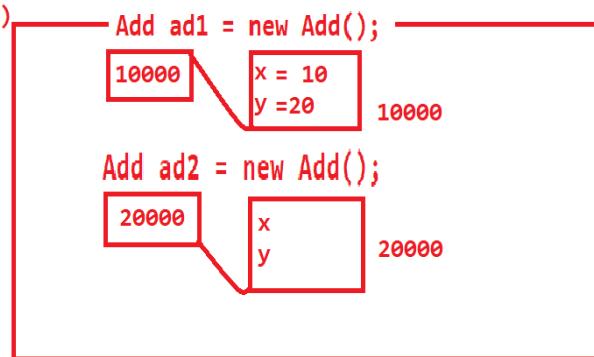
The Following Example shows the use of this reference

```

class Add
{
    int x,y;
    void setValue(int x,int y)
    {
        this.x=x;
        this.y=y;
    }
    void showAdd()
    {
        System.out.printf("Addition is %d\n",x+y);
    }
}
public class AddApp
{
    public static void main(String x[])
    {
        Add ad1 = new Add();
        ad1.setValue(10,20);
        ad1.showAdd();

        Add ad2 = new Add();
        ad2.setValue(5,4);
        ad2.showAdd();
    }
}

```



If we think about the above code we have the statement name as `Add` `ad1=new Add()` when we call the `ad1.setValue()` method then 10 and 20 parameter pass to `setValue()` function we have the statement `this.x=x` here this pointer or reference points to `ad1` reference means points to 10000 address as per our example means `this.x` use the value of `ad1` and again we have the one more object in our code name as `Add` `ad2 = new Add()` when we call the method name as `ad2.setValue()` in `setValue()` we have the statement name as `this.x` here this points to `ad2` reference means as per our example `ad2` reference having address 20000 thousand means this points to 20000 means `this.x` is equal with `ad2.x` so we can say this reference points to current running object in memory.

### MCQ Question on local variable

---

**Question1: what will be the output of given code?**

---

```
public class ScopeOfVariable {  
    public static void main(String[] args) {  
        int x = 10;  
        int y = 20;  
        {  
            System.out.print(x + ", " + y);  
        }  
        {  
            x = 15;  
            System.out.print(" - " + x + ", " + y);  
        }  
        System.out.print(" - " + x + ", " + y);  
    }  
}
```

---

**Options**

---

- a) 10, 20 - 15, 20 - 15, 20
- b) 10, 20 - 15, 20 - 10, 20
- c) Compilation Error
- d) Runtime Error

**Correct Ans:** a

**Explanation:** in above example we have the two variables name as x and y and these are the local variables in main function and we initialize the initial value in x=10 and y=20 we have the statement { System.out.print(x+", "+y) } here we print the values of x and y so we get the answer 10 , 20 then we have one more block { x=15 System.out.print("-"+x+", "+y) } here we initialize the value to x=15 so variable always use the last existing value so x use the 15 value so when we print the value using System.out.print("-"+x+", "+y) so x print the -15 and y print the 20 previous value because we not initialize the value in x and y and we have the last statement

System.out.print("-" + x + "," + y) here x print -15 and y print the 20 so we have the final output is 10 20 -15 20 -15 20.

**Question2: what will be the output of given code?**

```
public class UniversalMusic {  
    public static void main(String[] args) {  
        /*A*/ int pop = 'P';  
        /*B*/ char reggae = 'R';  
        {  
            /*C*/ System.out.println(pop + " " + reggae + " " + hiphop);  
            /*D*/ float hiphop = 31.3f;  
            /*E*/ System.out.println(pop + " " + reggae + " " + hiphop);  
            /*F*/ double jazz = hiphop;  
        }  
        /*G*/ System.out.println(pop + " " + reggae + " " + hiphop);  
        /*H*/ System.out.println(jazz);  
        {  
            /*I*/ short hiphop = 850062;  
            /*J*/ System.out.println(pop + " " + reggae + " " + hiphop);  
        }  
        /*K*/ System.out.println(pop + " " + reggae + " " + hiphop);  
    }  
}
```

**Options**

---

- a) A, C, G, H, K
- b) C, G, H, K
- c) C, G, H, I, K
- d) Compilation error

**Correct Answer: d**

**Explanation:** above program generate the compilation error to us because variable hiphop we declare within first { } and closing block and we try to use this variable outside of block so local variable cannot access outside of his block so compile generate the compilation error to us cannot find the symbol.

---

### **Question3 (what will be the output of given code)**

---

```
public class Industries {  
    public static void main(String[] args) {  
        String name = "TEXMO";  
        {  
            System.out.println(name + " Industries");  
            name = "texmo";  
        }  
        System.out.println(name + " Industries");  
    }  
}
```

---

### **Options**

---

- a) TEXMO Industries  
texmo Industries
- b) TEXMO Industries  
TEXMO Industries
- c) Compilation Error
- d) Runtime Error

**Correct Answer: a**

**Explanation : if we think about code**

```
String name = "TEXMO";  
{  
    System.out.println(name + " Industries");  
    name = "texmo";  
}
```

We have the code here name ="TEXMO" here name use the TEXMO value and we print the values using System.out.println(name+" Industries") so we

get output TEXMO industries and we again initialize the variable name using small texmo and we print the variable outside of block using System.out.println(name +"industries"); so we have the output texmo industries.

#### **Question4: what will be the output of above code**

---

```
public class LifeTime {  
    public static void main(String[] args) {  
        if (true) {  
            int x = 10;  
            System.out.println("Value of X = " + x);  
            x++;  
        }  
    }  
}
```

#### **Options**

---

- a) Value of X = 10
- b) Value of X = 0
- c) Compilation Error
- d) Runtime Error

#### **Correct Answer: a**

**Explanation:** if we think about the above code we have the statement name as if (true) this statement indicate we have the if statement is true so if get satisfied and we initialize the value x=10 so the value of x is 10 and when we print the value using System.out.println ("Value of x = "+x) so it will print the value of x is 10.

#### **How to Pass array as parameter to the function in java**

---

When we want to pass multiple parameter of same type to the function so passing single single parameter to the function is better way to pass array as parameter to the function.

### **Syntax to passing array as parameter to the function**

---

```
class classname  
{ returntype functionname(datatype variablename[])  
{ write here your logics  
}  
}
```

**Note:** when we pass array to the function definition then we have to pass the base address of array from function calling to the function definition.

**Example:** we want to pass six parameter to the function using array and we want to calculate its addition.

```

class Sum
{
    int s=0;

    void calSum(int x[])
    {
        for(int i=0; i<x.length; i++)
        {
            s = s + x[i];
        }
    }

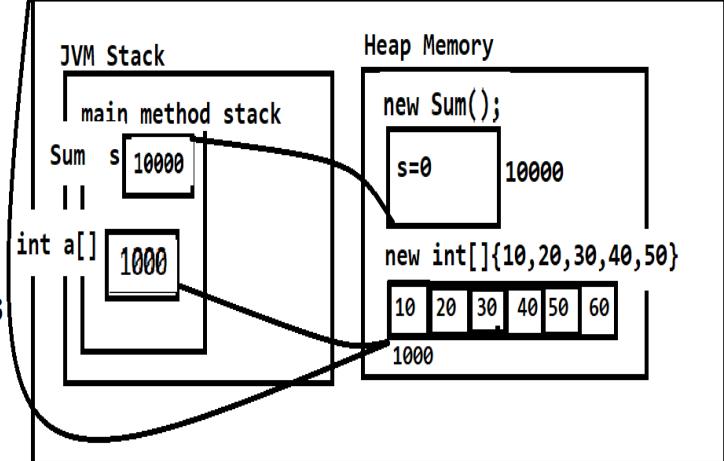
    System.out.printf("sum is %d\n",s);
}

public class SumApplication
{
    public static void main(String x[])
    {
        Sum s = new Sum();

        int a[]=new int[]{10,20,30,40,50};

        s.calSum(a);
    }
}

```



If we think about the above code we have the two classes name as `Sum` and `SumApplication` in `Sum` class we have the `calSum(int x[])` method which contain the array as parameter of type integer and in we have the one more class name as `SumApplication` in `SumApplication` class we have the method name as `public static void main(String x[])` and in the main method we have the statement `Sum s=new Sum()` means we create the object of `Sum` class when we create the object of `Sum` class then reference name `s` stored in main stack and `new Sum()` is our real object which store in heap section of memory then we have the statement name as `int a[]=new int[]{10,20,30,40,50}` this is our array here `a` is reference of array which stored in stack of main method and `new int[]{10,20,30,40,50}` is real object

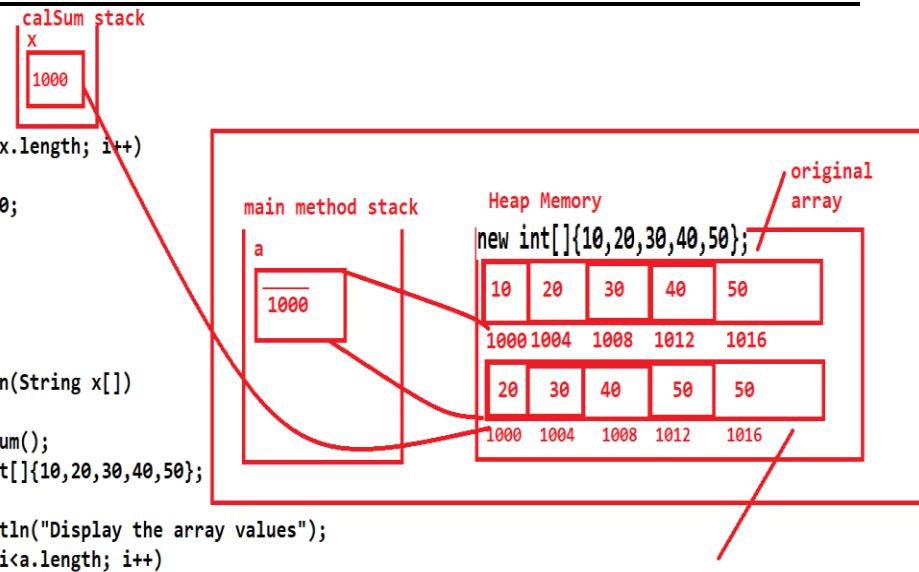
created in memory stored in heap section and new int[]{10,20,30,40,50} whose base address stored in array variable name as a when we call the method calSum() means we have the method name as s.calSum(a) from this method we pass the base address of array variable to calSum(int x[]) means we not pass duplicate copy of array just we pass the its address means variable x points to new int[]{10,20,30,40,50} from calSum() stack to heap section of memory means indirectly x use the memory of a array means x use the value of array a so we have the output is Sum is 150 because  $10+20+30+40+50=150$ .

### **MCQ Question on array as parameter using java**

```

class Sum
{
    int s=0;
    void calSum(int x[])
    {
        for(int i=0; i<x.length; i++)
        {
            x[i]=x[i]+10;
        }
    }
}
public class SumApplication
{
    public static void main(String x[])
    {
        Sum s = new Sum();
        int a[]=new int[]{10,20,30,40,50};
        s.calSum(a);
        System.out.println("Display the array values");
        for(int i=0; i<a.length; i++)
        {
            System.out.printf("a[%d] --->%d\n",i,a[i]);
        }
    }
}

```



### **Output**

```
D:\evenigadvjava>javac SumApplication.java

D:\evenigadvjava>java SumApplication
Display the array values
a[0] --->20
a[1] --->30
a[2] --->40
a[3] --->50
a[4] --->60

D:\evenigadvjava>
```

### **Explanation of above code**

If we think about the above code we have the two classes name as Sum and SumApplication in Sum class we have the function or method name as calSum(int x[]) which contain the array as parameter and in main method we have array name as int a[] = new int[]{10,20,30,40,50} as per this statement array allocate its memory in heap section and its reference present in main function and store the base address of array present in heap section of memory when we call the method name as calSum(a) then base address of array a pass to calSum(int x[]) function then x array store the base address of a means x points to the array a stored in memory and in calSum() function we have the statement

```
for (int i=0; i<x.length; i++) {
    x[i] =x[i]+10;
}
```

This statement modify the value of array x means indirectly we modify the value of array a and increment every value by 10 and stored in array and in

main function when we print the value of array a then we get the answer shown in above output.

**WAP to Create the class Name as Company with method name as addNewEmployee() with parameter name , id ,sal ,qualification and one more method name as showDetail() and display the employee details using this method.**

### **Source code**

---

```
class Company
{ private String name;
private int id;
private int sal;
private String qual;
public void addNewEmployee(String name,int id,int sal, String qual)
{ this.name=name;
this.id=id;
this.sal=sal;
this.qual=qual;
}
public void show ()
{ System.out.println (name+"\t"+id+"\t"+sal+"\t"+qual);
}
}
public class CompanyApplication
{ public static void main (String x [])
{ Company c = new Company ();
c.addNewEmployee ("Ram", 1, 10000,"BE");
c.show ();
}
}
```

### **Code Explanation**

---

If we think about the above code we have the two classes name as Company and CompanyApplication in CompanyApplication we create the object of Company class and in using this object we call the method name as addNewEmployee () in this method we pass the four parameter and store the data in object and we call the show () method for display the record of employee.

The major limitation of above code is if we have the number of parameter in addNewEmployee() like as name ,id,sal,qual,address,preexp,expsal,pan,adhar etc then it is verify difficult to pass parameter and remember its sequence at function calling point so if we want to avoid this problem java provide the POJO class concept

**Q. what is the POJO class?**

**POJO Class**

---

**POJO stands for plain old java objects POJO class means a class with setter and getter methods.**

**The Major Goal of POJO class is to store data and to retrieve data**

**It Work Like as Container in JAVA.**

**We can design the POJO class like as**

```
class Employee
{
    private int id;
    private String name;

    public void setId(int id)
    { this.id=id;
    }
    public int getId()
    { return id;
    }
    public void setName(String name)
    { this.name=name;
    }
    public String getName()
    { return name;
    }
}
```

Using setter method we can store data in object and using getter method we can retrieve data from object.

Following Example give detail about above concept

```

class Employee
{
    private int id;
    private String name;

    public void setId(int id)
    { this.id=id; }
    public int getId()
    { return id; }
    public void setName(String name)
    { this.name=name; }
    public String getName()
    { return name; }
}

public class EmployeeApplication
{
    public static void main(String x[])
    {
        Employee emp = new Employee();
        emp.setId(1);
        emp.setName("Ram");
        System.out.println(emp.getId()+"\t"+emp.getName());
    }
}

```

Employee emp = new Employee();  
10000  
id = 1  
name = Ram  
10000

**Normally we use POJO class to pass different type of parameter to the function**

**Note:** as per java standard if we have the more than seven different types of parameter in function so passing single single parameter is not good approach so better store all parameters in POJO class and pass to function.

### **Following Example show the use of POJO class**

---

```

class Company
{private int id;
private String name;
private long sal;
void addNewEmployee(int id,String name,long sal)
{ this.id=id;
this.name=name;
this.sal=sal;
}
void showDetail()

```

```
{ System.out.println(name+"\t"+id+"\t"+sal);
}
}
public class CompanyApp
{ public static void main(String x[]) { Company c =new Company();
    c.addNewEmployee(1,"Ram",10000);
    c.showDetail();
}
}
```

### **Output**

---

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac CompanyApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java CompanyApp
Ram      1      10000

C:\Program Files\Java\jdk1.8.0_291\bin>
```

If we think about above code in Company class we have the function name addNewCompany() and this function contain the three Parameter of different type id for int, name is string and sal is long Suppose consider if addNewEmployee() contain 50 parameter of different type so it is very tedious task to maintain the sequence and passing parameter to function So if we want to avoid this problem we have the POJO class concept.

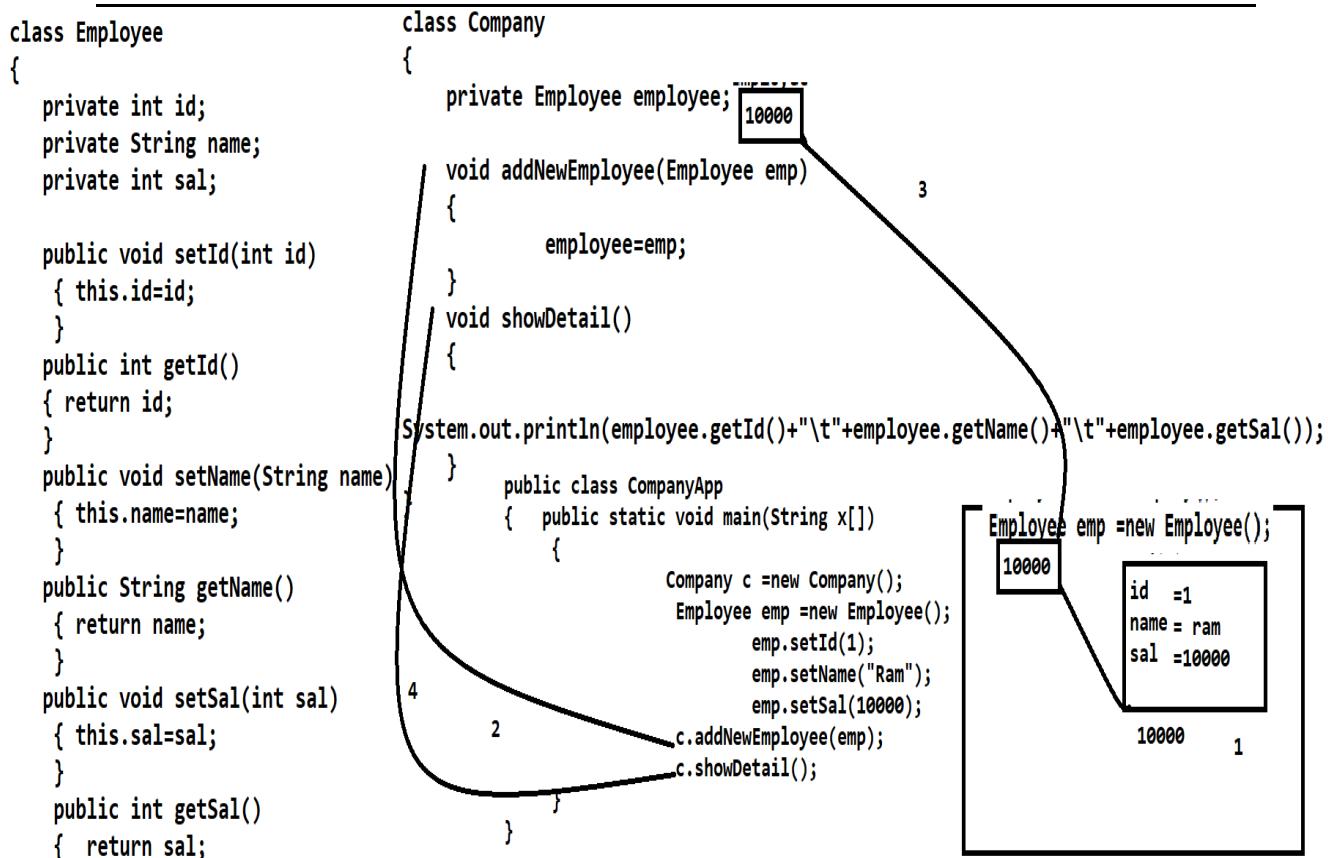
Means as per our example we create the class name as Employee and store the three values in Employee class id, name and sal.

We pass the reference of Employee class in addNewEmployee() function of Company class

Means when we call the addNewEmployee() we not need to pass single parameter to addNewEmployee

We create the object of Employee class and store all data in Employee object and pass Employee reference of addNewEmployee() function of Company class.

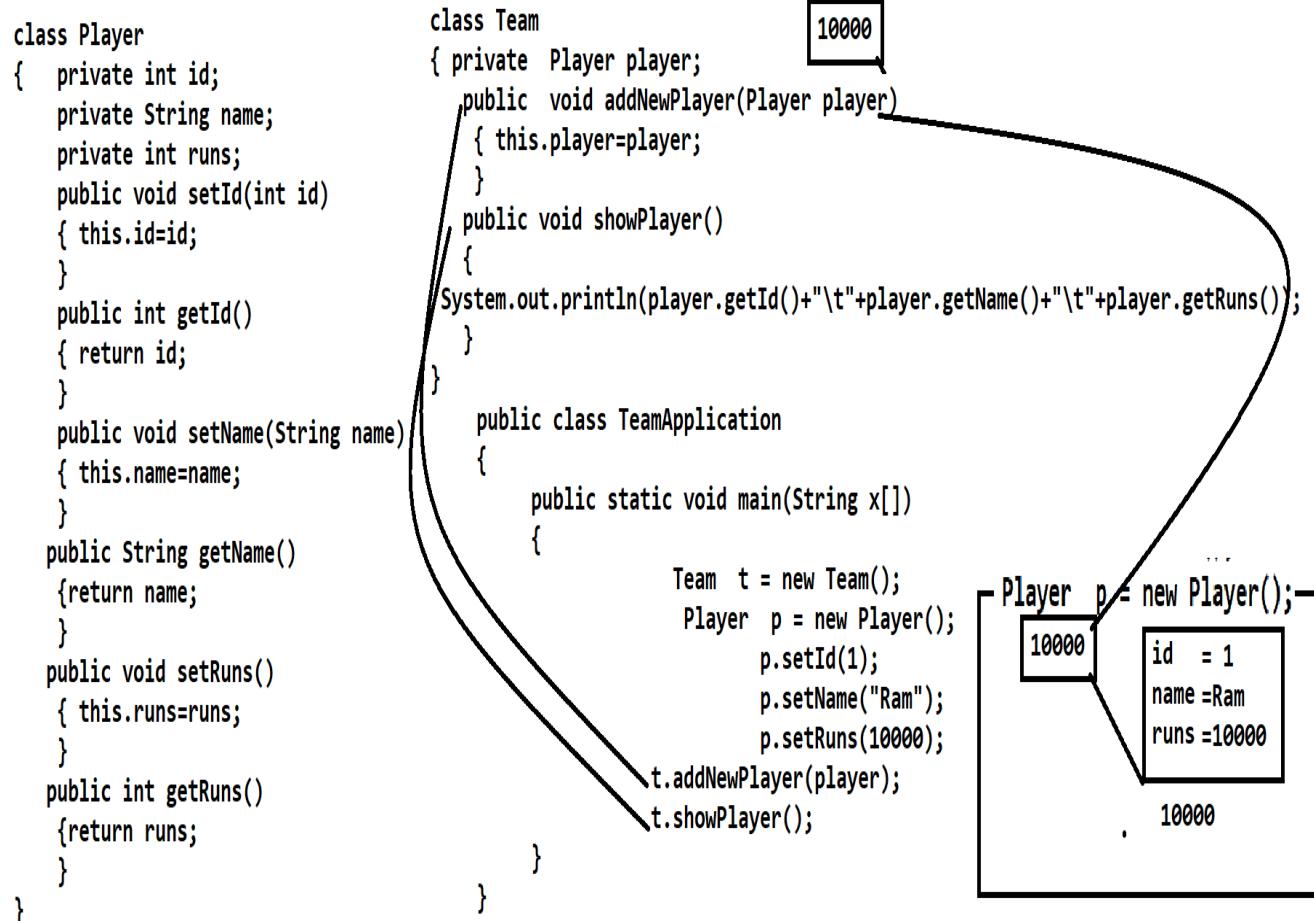
### **Following Program show the meaning of above paragraph**



### **Example**

WAP to create the class name as Player with field id, name and runs and create the class name as Team with two methods void addNewPlayer() and

`showPlayer()`. In `addNewPlayer ()` method pass reference of `Player` class and in `showPlayer ()` method display the information about the `Player`.



### Example

**WAP to create the class name as product with field name and price As well as Write class name as Bill and pass the reference of Product class in Bill class. Bill class contain the two methods void setProduct () and showBill ()**

### Example

```
class Product
{
    private String name;
    private int price;
    public void setName(String name)
    { this.name=name;
    }
    public String getName(){ return name;
    }
    public void setPrice(int price)
    { this.price=price;
    }
    public int getPrice()
    { return price;
    }
}
class Bill
{
    Product product;
    void setProduct(Product product)
    { this.product=product;
    }
    void showProduct()
    { System.out.println(product.getName()+"\t"+product.getPrice());
    }
}
public class BillingApplication
{public static void main (String x[]) {
    Bill b = new Bill();
    Product p = new Product ();
    p.setName ("Parle-G");
    p.setPrice (10);
    b.setProduct (p);
}}
```

```
b.showProduct();  
}  
}
```

### **MCQ Question on POJO class**

---

```
class Player  
{ private int id;  
    private String name;  
    public void setId(int id)  
    { this.id=id;  
    }  
    public int getId()  
    { return id;  
    }  
    public void setName(String name)  
    { this.name=name;  
    }  
    public String getName()  
    { return name;  
    }  
}  
class Team  
{  
    public void addPlayer(Player player)
```

```
{  
    player.setId(100);  
    player.setName("Dhoni");  
}  
}  
public class PlayerApplication  
{ public static void main(String x[])  
{    Player p = new Player();  
    p.setId(1);  
    p.setName("ABC");  
    Team t =new Team();  
    t.addPlayer(p);  
    System.out.println(p.getId()+"\t"+p.getName());  
}  
}
```

## **Output**

```
C:\Program Files\Java\jdk1.8.0_291\bin>java PlayerApplication  
100      Dhoni
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>
```

## **Explanation**

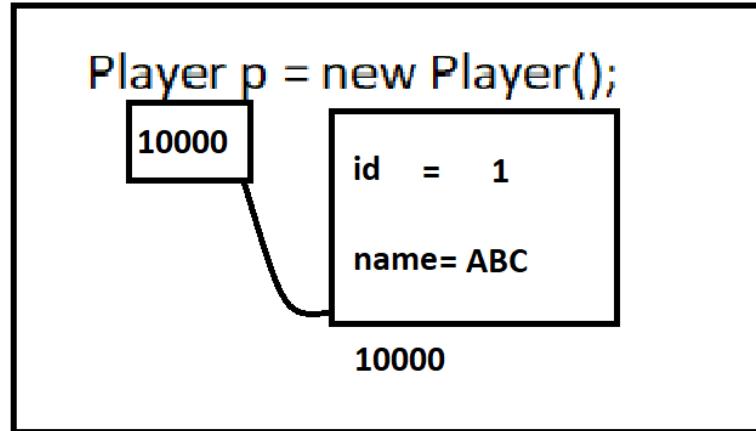
If we think about the above code then we have three class's one is Player which is POJO class and we use it for storing the information about the Player and in this class we have the two field name and id. We have the one more class name as Team and in Team class we have the method name as addNewPlayer(Team t) in this method we pass reference of Player class in addNewPlayer() and In PlayerApplication class contain main method so first now we will discuss about the main method present in PlayerApplication class.

In main method first we have the following statement

```
Player p = new Player();
p.setId(1);
p.setName("ABC");
```

as per above code we create the object of player class and store the id and name in Player class object shown in following diagram.

```
Player p = new Player();
p.setId(1);
p.setName("ABC");
```



Here we create the object of Player class whose address is 10000 as per our example and we store this address in reference name as p after that we have the statement

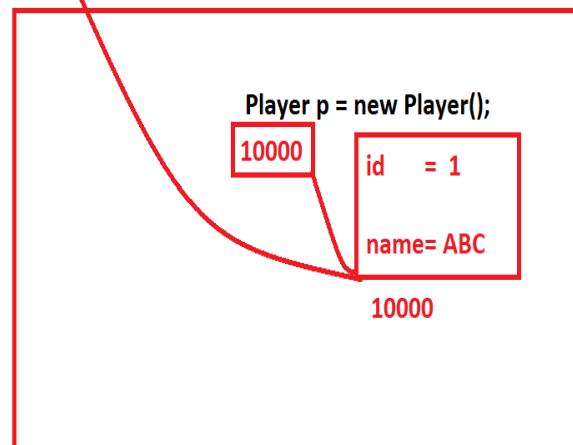
```
Team t =new Team();
t.addPlayer(p);
```

as per above code we create the object of Team class Team t = new Team and we call the method name as t.addPlayer(p) means we pass the reference of Player class to Team class method addPlayer() shown in following diagram.

```

class Team
{
    public void addPlayer(Player player)
    {
        player.setId(100);
        player.setName("Dhoni");
    }
}
public class PlayerApplication
{
    public static void main(String x[])
    {
        Player p = new Player();
        p.setId(1);
        p.setName("ABC");
        Team t = new Team();
        t.addPlayer(p);
        System.out.println(p.getId()+"\t"+p.getName());
    }
}

```

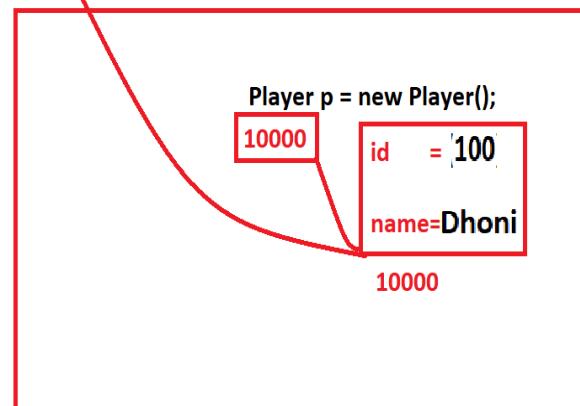


When we pass the reference of Player to `addPlayer()` method of Team class means as per our example reference `p` present in main method and reference Player present in `addPlayer()` method points to same the memory i.e 10000 addressas per our example so in `addPlayer()` method we have the two statements `player.setId(100)` and `player.setName("Dhoni")` means we modify the content on 10000 address space from `addPlayer()` method means so 100 and Dhoni get override on 10000 address space shown in following diagram.

```

class Team
{
    public void addPlayer(Player player)
    {
        player.setId(100);
        player.setName("Dhoni");
    }
}
public class PlayerApplication
{
    public static void main(String x[])
    {
        Player p = new Player();
        p.setId(1);
        p.setName("ABC");
        Team t = new Team();
        t.addPlayer(p);
        System.out.println(p.getId()+"\t"+p.getName());
    }
}

```



After t.addPlayer (p) we have the statement given below

System.out.println (p.getId () +"\t"+p.getName ()); this statement print the 100 and dhoni so our program print the output 100 and dhoni.

### Array of Objects concept in java

Array of objects is used for store the more than one object data in single name reference called as array of objects.

### How to create the array of objects in java

```

classname ref[] = new classname[size] ; //this array of reference
for (int i=0; i<ref.length; i++)
{
    ref[i] = new classname () ; // array of objects
}

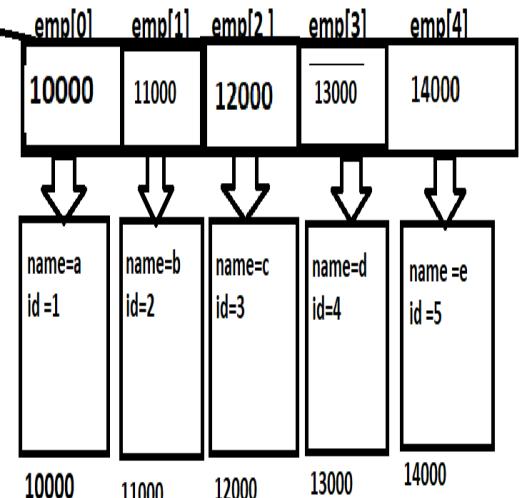
```

When we create the array of objects we have to create the array of reference because if we have multiple objects then we have to create the number of references for store that objects.

**Example:** suppose consider we have the Employee class with field id and name and we want to create the five employee objects store data in it and display its record.

```
import java.util.*;
class Employee
{
    private int id;
    private String name;
    public void setId(int id)
    { this.id=id; }
    public int getId()
    { return id; }
    public void setName(String name)
    { this.name=name; }
    public String getName()
    { return name; }
}
```

```
public class EmployeeApplication
{
    public static void main(String x[])
    { Employee emp[] = new Employee[5];
        for(int i=0; i<emp.length; i++)
        { Scanner xyz = new Scanner(System.in);
            System.out.println("Enter the name and id");
            String name=xyz.nextLine();
            int id=xyz.nextInt();
            emp[i]=new Employee();
            emp[i].setName(name);
            emp[i].setId(id);
        }
        System.out.println("display the detail of employee");
        for(int i=0; i<emp.length; i++)
        { System.out.println(emp[i].getName()+"\t"+emp[i].getId());
        }
    }
}
```



## Output

```
C:\Program Files\Java\jdk1.8.0_291\bin>java EmployeeApplication
Enter the name and id
a
1
Enter the name and id
b
2
Enter the name and id
c
3
Enter the name and id
d
4
Enter the name and id
e
5
display the detail of employee
a      1
b      2
c      3
d      4
e      5
```

## Source code of above diagram

---

```
import java.util.*;
class Employee
{
    private int id;
    private String name;
    public void setId(int id)
    { this.id=id;
    }
    public int getId()
```

```
{ return id;
}
public void setName(String name)
{ this.name=name;
}
public String getName()
{ return name;
}
}
public class EmployeeApplication
{
    public static void main(String x[])
    {
        Employee emp[] = new Employee[5];
        for(int i=0; i<emp.length;i++)
        { Scanner xyz = new Scanner(System.in);
            System.out.println("Enter the name and id");
            String name=xyz.nextLine();
            int id=xyz.nextInt();
            emp[i]=new Employee();
            emp[i].setName(name);
            emp[i].setId(id);
        }
        System.out.println("display the detail of employee");
        for(int i=0; i<emp.length; i++)
        { System.out.println(emp[i].getName()+"\t"+emp[i].getId());
        }
    }
}
```

### **Description of above code**

---

If we think about the above code we have the two classes name as Employee it is our POJO class and EmployeeApplication class which contain main method in main method class we have the statement

Employee emp[] = new Employee[5] this statement indicate the array of reference of Employee class means we have the 5 references because we want to store the five object address in it and we have the for loop given below

```
for(int i=0; i<emp.length;i++)
{ Scanner xyz = new Scanner(System.in);
  System.out.println("Enter the name and id");
  String name=xyz.nextLine();
  int id=xyz.nextInt();
  emp[i]=new Employee();
  emp[i].setName(name);
  emp[i].setId(id);
}
```

---

For(int i=0; i<emp.length; i++): this statement travel your for loop five times after that we have the statement name as Scanner xyz = new Scanner(System.in) this statement indicate we have the scanner class for input after that we have the statement String name=xyz.nextLine() for accept the input of type string int id=xyz.nextInt() for accept the input of type integer emp[i]=new Employee(): this statement indicate we create the new object of Employee every time and store its address on array specified index after that we have the statement emp[i].setName(name) and emp[i].setId(id) this statement indicate we store the data in object using its index

And we have the one more for loop data fetching from array of objects.

```
for(int i=0; i<emp.length; i++)
{ System.out.println(emp[i].getName()+"\t"+emp[i].getId());}
```

---

## Constructor

---

Constructor is a function same name as class name but without return type.

Syntax:

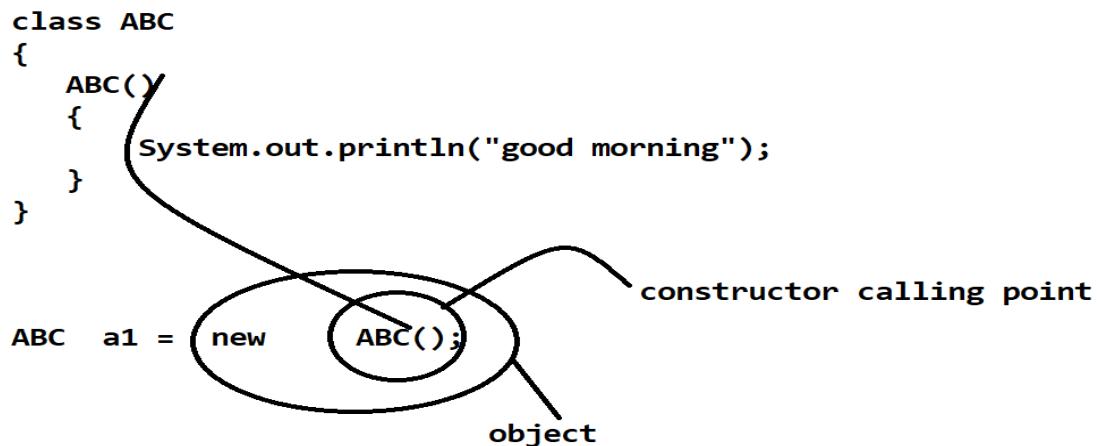
```
class classname
{
    classname()
    {
    }
}
```

**Q. why use constructor or what is the benefit of constructor?**

---

**The major goal of constructor is**

- 1) Call the function automatically when class object get created.
- 2) To initialize the memory of object means when we call the constructor then JVM allocate memory for object.



## Source Code

---

```
package org.techhub;

class ABC
{
    ABC()
    {
        System.out.println("I am constructor");
    }
}

public class ConstructorApplication {

    public static void main(String x[])
    {
        ABC a1 = new ABC();
    }
}
```

## Types of constructor

---

**1) Default constructor** : default constructor means if we not pass parameter to the constructor called as default constructor.

```
class ABC
{
    ABC()
    {
        System.out.println("I am constructor");
    }
}

public class ConstructorApplication {
```

```
public static void main(String x[])
{
    ABC a1 = new ABC();
}
}
```

**Note:** if we not declare the constructor within class then java add the by default constructor in class called as implicit constructor.

## Can we see the implicit constructor in java?

Yes we can see the implicit constructor in java

The screenshot shows a Windows desktop environment. On the left, there is a Notepad window titled 'A - Notepad' containing the following Java code:

```
class A
{
}
```

A red oval highlights the opening brace '{' of the class definition, with the annotation 'it is code of programmer' written below it in red. A red arrow points from this oval to the corresponding closing brace '}' in the Command Prompt window.

On the right, there is a Command Prompt window titled 'Command Prompt' showing the following output:

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac A.java

C:\Program Files\Java\jdk1.8.0_291\bin>javap A
Compiled from "A.java"
class A {
    A();           this is the implicit
}                  constructor
C:\Program Files\Java\jdk1.8.0_291\bin>
```

A red oval highlights the constructor declaration 'A();' in the output, with the annotation 'this is the implicit constructor' written above it in red. Another red arrow points from this oval to the opening brace '{' of the class definition in the Notepad window. A second red oval highlights the closing brace '}' in the output, with the annotation 'it is code of compiler' written below it in red.

**2) Argument constructor:** Augmented constructor or Parameterized constructor if we pass parameter to constructor called as parameterized constructor. When we pass the parameter to constructor then we need to pass parameter from where object get created.

**Following Example shows the Parameterized constructor**

```
class Add
{ int a,b; 100,200
  Add(int a,int b)
  {
    this.a=a;
    this.b=b;
  }
  void showAdd()
  {
    System.out.printf("Addition is %d\n",a+b);
  }
}
public class AddApplication
{
  public static void main(String x[])
  {
    Add ad = new Add(100,200);
    ad.show();
  }
}
```

### **3) Overloaded constructor:**

**Before constructor overloading we need to know what is the overloading.**

**Overloading is part of compile time polymorphism.**

**What is the meaning of Polymorphism?**

---

**Polymorphism means if we use the same thing for different purpose called as polymorphism**

**Poly means many and morphs means form called as polymorphism**

### **Example of Polymorphism is**

---

Suppose consider Person is best example of Polymorphism

He can change its behavior according its requirement

Suppose person is employee of company Then his behavior like as employee and same person is parent then He can change its behavior as parent if same person is husband then He can change its behavious like husband.

### **There are two types of polymorphism**

---

**a) Compile time polymorphism:** compile time polymorphism method calling is decide at the time of program compilation called as compile time polymorphism.

#### **There are two types of compile time polymorphism in java**

---

**i) Function overloading:** function overloading means if we use the same function name with different parameter with different data type with different sequence called as function overloading.

### **Example**

---

```
void add(int x,int y)
{
}
void add(float x,float y)
{
}
void add(int x,float y)
{
}

void add(float x,int y)
{
}

}
```

If we want to work with function overloading we have the some important points.

- 
- a) Function name should be same
  - b) Parameter list or parameter type must be different or minimum sequence should be different
  - c) In function overloading return type is not consider  
Means there is no compulsion return must be same for all function you can give the different return type to the function
  - d) Which function get executed is depend on how much parameter pass in it and its sequence.

Can you give real time example where we can use the function overloading in project?

---

Suppose consider we have training center admission module implementation and training center takes two types of admission

A) Fresher admission

## B) Working Professional admission

In this scenario we design two functions with same name

Void admission()

### **First function for fresher like as**

---

```
void admission(String name,String email,String collegeName,int  
passyear)  
{  
}
```

### **Second function for Working Professional**

---

```
void admission(int expinyear,String name,String email,String  
contact,String compName,int currentCTC,int expectedPackage)  
{  
}
```

### **Following example demonstrate the function overloading**

---

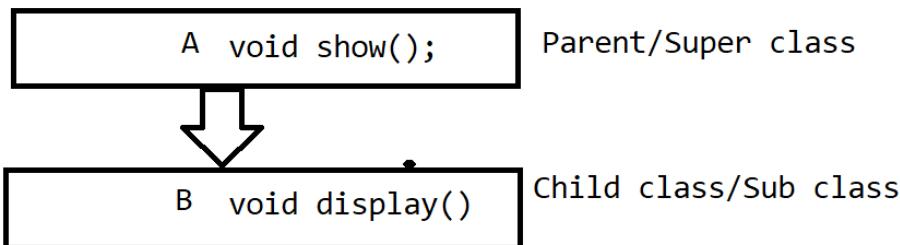
```
File Edit Format View Help
class Square
{
    void calSquare(int x)
    {
        System.out.printf("Square of integer is %d\n",x*x);
    }
    void calSquare(float x)
    {
        System.out.printf("Square of float is %f\n",x*x);
    }
}
public class SquareApplication
{
    public static void main(String x[])
    {
        Square s1 = new Square();
        s1.calSquare(5); //call integer function here
        s1.calSquare(5.4f); //call floating function
    }
}
```

**ii) Constructor overloading.**

- b) Run time polymorphism
- 4) this() constructor

## Inheritance

Inheritance means to transfer the properties of one class to another class called as inheritance.



Note : in Class B contain two function name as display() and show()

Because B is Child class of A means All properties from class A present in class B.

## Why use the inheritance or what is the benefit of inheritance?

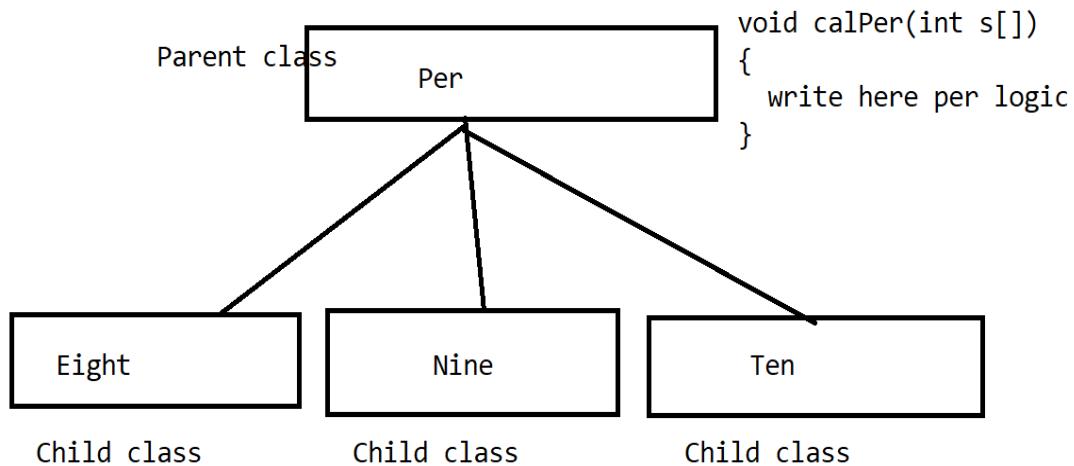
The major benefit of inheritance given below

**1) Code reusability:** means as per our example we not need to create the object of class A we can access the all member of class A using class B object

**2) Generalization:** generalization means if we have some logics and we want to reuse multiple then we can create the logic and use it multiple places without creating object just we have to inherit it.

### Suppose consider we have the example

We have the one class name as per with method calPer() and we have the three different classes name as Eight , Nine and Ten we want to calculate the percentage for all classes so writing calPer(int []) in every class we can declare the method in single parent class and inherit the parent class in Eight Nine and Ten classes.



Note : Here we not need not create the object of Per class  
Per logic present in Eight , Nine And Ten Virtually from its parent class

## **How to perform the inheritance using java or How to transfer the properties from one class to another class in java**

---

If we want to perform or if we want to transfer the properties of one class to another class in java we have the extends keyword.

```
class A //parent class
{
}
class B extends A //child class
{}
```

**Note:** In The case of java we cannot create any program without inheritance.

Because in the case of java every class having a default parent class called as Object class.

### **Example**

---

### Internal meaning

```
class A           class A extends java.lang.Object
{
{
}
}
```

## Q. Why java provide Object class as parent to every class?

Because Object class contain the some common methods those required to every class in java

- 1) boolean equals(Object)
- 2) int hashCode()
- 3) String toString()
- 4) Object clone()
- 5) Class getClass()
- 6) void finalize()
- 7) void wait()
- 8) void wait(int)
- 9) void notify()
- 10) void notifyAll()
- 11) static { } block

If we want to see this method we can the following command on command prompt.

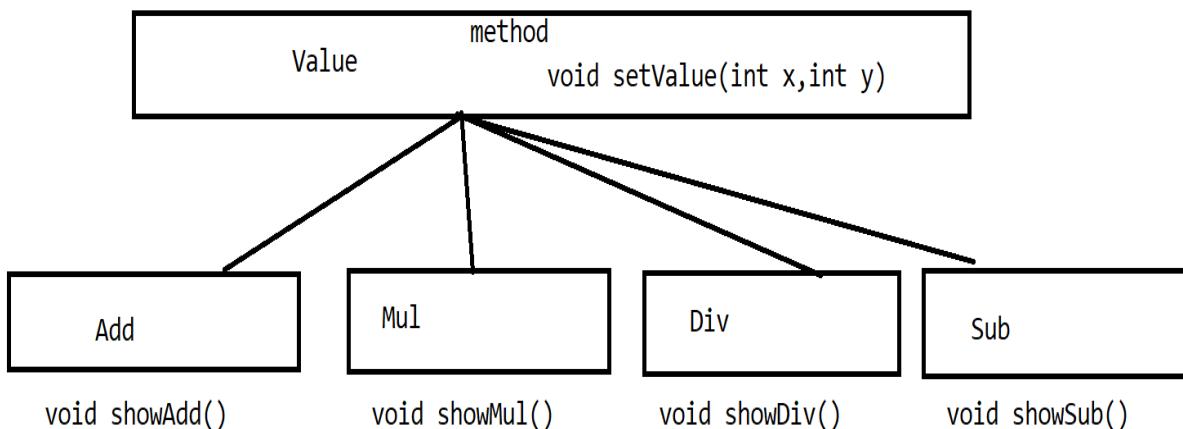
```
C:\Users\Admin>javap java.lang.Object      javap shwo the detail of every class
Compiled from "Object.java"
public class java.lang.Object {
    public java.lang.Object();
    public final native java.lang.Class<?> getClass();
    public native int hashCode();
    public boolean equals(java.lang.Object);
    protected native java.lang.Object clone() throws java.lang.CloneNotSupportedException;
    public java.lang.String toString();
    public final native void notify();
    public final native void notifyAll();
    public final native void wait(long) throws java.lang.InterruptedException;
    public final void wait(long, int) throws java.lang.InterruptedException;
    public final void wait() throws java.lang.InterruptedException;
    protected void finalize() throws java.lang.Throwable;
    static {};
}
```

**Note:** all methods we will discuss in some later chapters.

In the case of inheritance we not need to create the object of parent class we can access the parent member using child object.

## Example

WAP to create the Calculate Example using Inheritance following  
Diagram shows the Example detail



In above example we have the parent class name as Value with function name as setValue() with two parameter and we have the four child classes name as Add which is used for perform addition operation using showAdd() function , Mul class which is used for perform multiplication with showMul() ,Div class which is used for calculate division with showDiv() function and Sub class which is used for calculate the subtraction with showSub() function

Here we have the some option like as

Case 1: for addition operation

Case 2: for multiplication

Case 3: division operation

Case 4: subtraction operation

As per our example we not need to create the object of Value class we can use the Value content using Add,Sub ,Mul and Div class objects.

Source Code given below

---

```
package org.techhub;

import java.util.*;
class Value {
    int a, b;

    void setValue(int x, int y) {
        a = x;
        b = y;
    }
}

class Add extends Value {
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
}

class Mul extends Value {
    void showMul() {
        System.out.printf("Multiplication is %d\n", a * b);
    }
}

class Sub extends Value {
    void showSub() {
        System.out.printf("Substraction is %d\n", a - b);
    }
}
```

---

}

```
class Div extends Value {  
    void showDiv() {  
        System.out.printf("Division is %d\n", a / b);  
    }  
}
```

```
public class CalculatorApplication {
```

```
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Scanner xyz = new Scanner(System.in);  
        int choice;  
        System.out.println("1:Addition");  
        System.out.println("2:Multiplication");  
        System.out.println("3:Division");  
        System.out.println("4:Subtraction");  
        System.out.println("Enter the two values");  
        int a = xyz.nextInt();  
        int b = xyz.nextInt();  
        System.out.println("Enter your choice");  
        choice = xyz.nextInt();  
        switch (choice) {  
            case 1:  
                Add ad = new Add();  
                ad.setValue(a, b);  
                ad.showAdd();  
                break;  
            case 2:  
                Mul m = new Mul();  
                m.setValue(a, b);  
        }  
    }  
}
```

```
        m.showMul();
        break;
    case 3:
        Div d = new Div();
        d.setValue(a, b);
        d.showDiv();
        break;
    case 4:
        Sub s = new Sub();
        s.setValue(a, b);
        s.showSub();
        break;
    default:
        System.out.println("Wrong choice");
    }
}
Output
```

---

```
1:Addition
2:Multiplication
3:Division
4:Substraction
Enter the two values
10
20
Enter your choice
1
Addition is 30
```

## Constructor in Inheritance

---

When we have the default constructor in parent class and constructor in child class and when we create the object of child class then by default parent constructor get executed before child class constructor.

Means JVM call the parent object internally before child object So parent constructor gets executed automatically before child constructor.

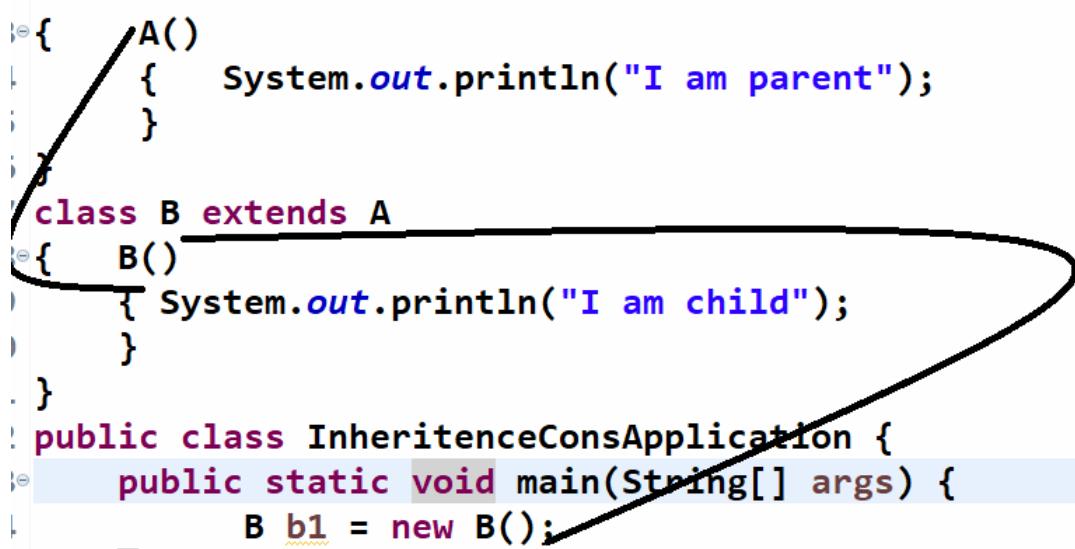
## Source Code Example

---

```
package org.techhub;
class A
{
    A()
    {
        System.out.println("I am parent");
    }
}

class B extends A
{
    B()
    {
        System.out.println("I am child");
    }
}

public class InheritanceConsApplication {
    public static void main(String[] args) {
        B b1 = new B();
    }
}
```



## Output

---

```
I am parent
I am child
```

**Note:** when we have the parameter in parent class constructor then JVM not execute the parent of constructor before child constructor automatically in this case we have the use the super() constructor for passing parameter to parent constructor from child class constructor.

super() constructor must be first line of code in child class constructor

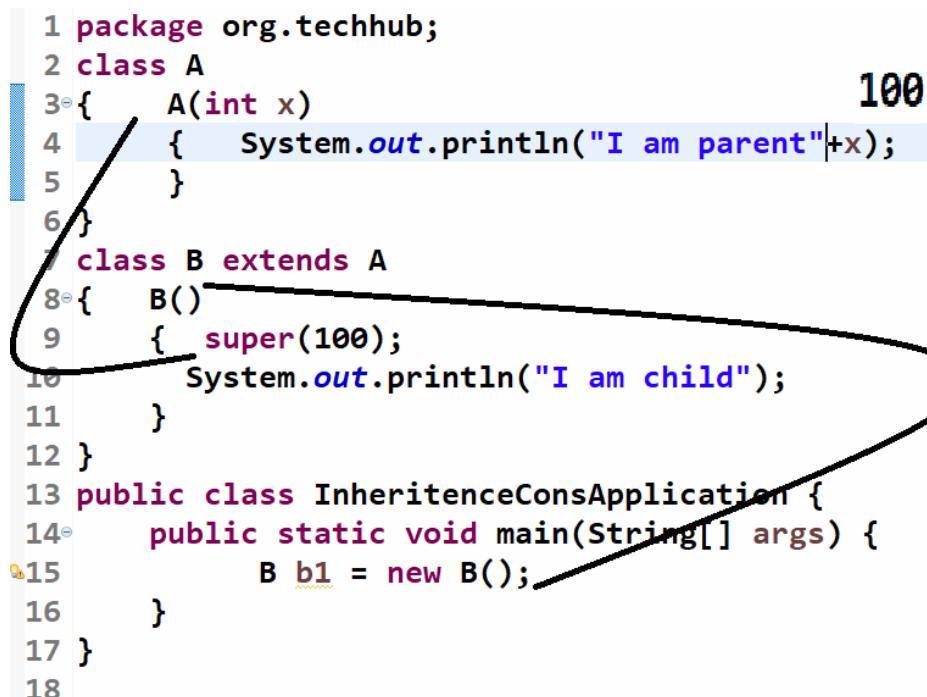
Example given below

---

```

1 package org.techhub;
2 class A
3 {   A(int x)
4     {   System.out.println("I am parent"+x);
5     }
6 }
7 class B extends A
8 {   B()
9     {   super(100);
10        System.out.println("I am child");
11    }
12 }
13 public class InheritanceConsApplication {
14     public static void main(String[] args) {
15         B b1 = new B();
16     }
17 }
18

```




---

**Output**

**I am parent100  
I am child**

**Q. what is the super () constructor?**

---

Super() constructor is used for perform constructor chaining in inheritance between child and parent means when we have parameter present in parent class then child class having responsibility to pass parameter to parent constructor then we have to use the super() constructor in child class constructor with first line.

## Final Keyword

---

Final keyword can use with variable, with function and with class also.

**Final variable:** final variable means variable cannot modify its value once we assign it In Short we can say final variable is used for declare the constant in java.

```
public class FinalVarApplication {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        final int a=10;  
        ++a;  
        Sys The final local variable a cannot be assigned. It must be blank and not using a compound assignment  
        } 1 quick fix available:  
            Remove 'final' modifier of 'a'  
    } Press 'F2' for focus
```

If we think about above code it will generate the error to us because we try to modify the value of variable a and we declare the variable a as final variable. We cannot modify the value of final variable.

## Final method

---

Final method means method cannot override in child class called as final method.

Before learn the final method we have to know what the method is overriding.

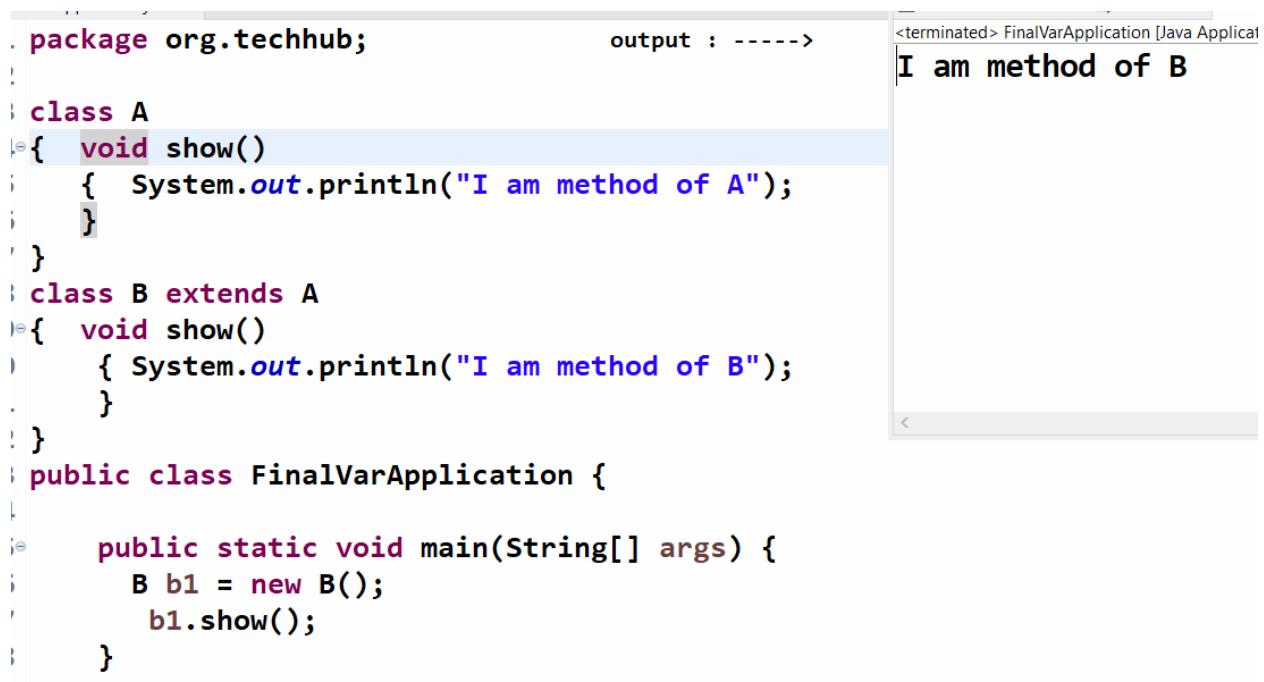
## Method overriding

---

Method overriding means if we define the method in parent class and redefine the same method in child class called as method overriding.

```
class A
{
    void show()
    {
        System.out.println("I am method A");
    }
}
class B extends A
{
    void show()  Here we override the show() method
    {
        System.out.println("I am method B");
    }
}
```

**Note:** in the case of method overriding if we create the object of child class and try to call the overridden method then by default child logic get executed.



The screenshot shows a Java code editor with the following code:

```
package org.techhub;
class A {
    void show() {
        System.out.println("I am method of A");
    }
}
class B extends A {
    void show() {
        System.out.println("I am method of B");
    }
}
public class FinalVarApplication {

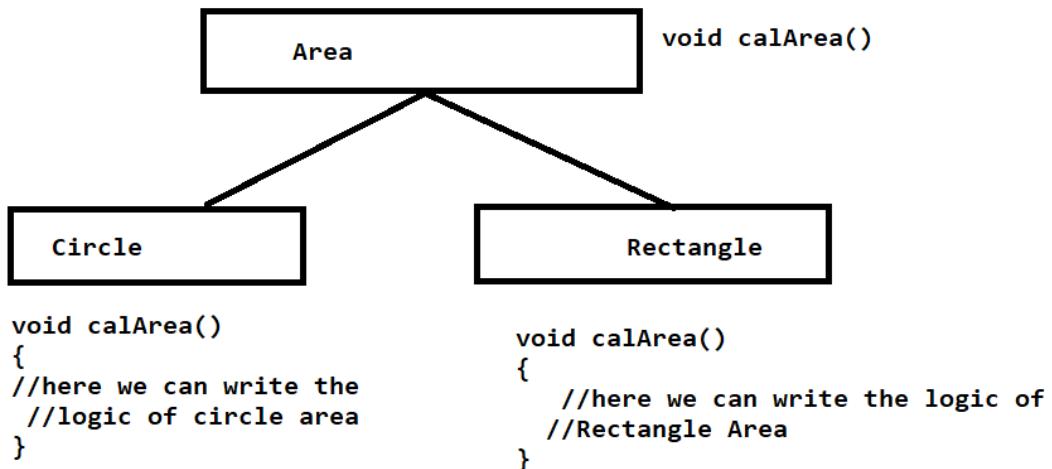
    public static void main(String[] args) {
        B b1 = new B();
        b1.show();
    }
}
```

To the right of the code, there is a terminal window titled "output : ----->". The terminal displays the output of the program:

```
<terminated> FinalVarApplication [Java Application]
I am method of B
```

## Q. Why we need to perform method overriding or what is the benefit of method overriding?

Method overriding is used normally when we want to customize the parent logic in child class according to requirement of child. Then we can use the method overriding.



Note: when we create the object of Circle class and call the calArea() method then we get result of circle area.

When we create the object of Reactangle class and call the calArea() method then we get result of Reactangle Area

## Example2

---

```
class Music
{
    void play()
    {
        System.out.println("playing music");
    }
}
class Sony extends Music
{
    void play()
    {
        System.out.println("Here sony music playing logic");
    }
}
class Samsung extends Music
{
    void play()
    {
        System.out.println("Here samsung music playing logic");
    }
}
```

## Example3

---

```

class Bank
{
    int getInterestRate()
    { return 0;
    }
}
class ICICI extends Bank
{
    int getInterestRate()
    { return 8;
    }
}
class HDFC extends Bank
{
    int getInterestRate()
    { return 9;
    }
}
public class OverridingApp
{ public static void main(String x[])
    { ICICI i = new ICICI();
        System.out.println("ICICI Interest rate is "+i.getInterestRate()); // 8
        HDFC h = new HDFC();
        System.out.println("HDFC Interest rate is "+h.getInterestRate());// 9
    }
}

```

Note: every bank charge interest rate but every bank having different rate so we create one generalize class name as Bank and customize the interest according to bank and its expenses.

means we customize the getInterestRate() method in ICICI with 8 percent and we customize the getInterestRate() method in HDFC with interest rate is 9

## Example4

---

```

class Company
{ int per;
  void perCriteria()
  { per=60;
  }
}
class FresherHiring extends Company
{
    void perCriteria()
    {
        per = 50;
    }
}
public class OverridingApp
{ public static void main(String x[])
    {
        FresherHiring f = new FresherHiring();
        f.perCriteria();
    }
}

```

Note: here in Company class we set the perCriteria() as 60% but we customize the freshere criteria with 50% means we customize the Company class in FresherHiring class

But some company want to fix the percentage and they are won't give permission to modify its logic in child class then we can declare the parent method as final.

```
class Company
{
    int per;
final void perCriteria()
{
    per=60;
}
}
class FresherHiring extends Company
{
    void perCriteria()
    {
        per = 50;
    }
}
public class OverridingApp
{
    public static void main(String x[])
    {
        FresherHiring f = new FresherHiring();
        f.perCriteria();
    }
}
```

Here compile will generate the error  
we cannot override the final method  
in child class and we try to override.

## **Q. How to avoid the method overriding or how to prevent the method overriding?**

If we want to avoid the method overriding we can declare the method as final in parent class.

Final keyword is used for restrict the method overriding means it is used for avoid the method overriding means it is used for parent logic modification from child class.

## Final class

---

If we use the final keyword with class then we cannot inherit the class in any another child class

Final class normally use for create the immutable classes in java.

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

Here Compiler will generate the error at compile time because we try to inherit the class A in class B

## **Abstract class and abstract methods**

---

Abstract class means class cannot create its object and abstract method means method cannot have definition.

abstract class A

```
{abstract void show ();  
}
```

A a1 = new A(); //it is not allowed

### **Q. Why use the abstract method and abstract classes ?**

---

- 1) abstract class is used for achieve abstraction
- 2) abstract method is used for achieve dynamic polymorphism

### **Q. What is the abstraction ?**

---

abstraction means to hide the implementation detail from end user at designing level called as abstraction. Means in abstraction just we provide the prototype of the work we not provide the detail description about the task or work. We write its implementation part or logical part or discretionary part where we want to implement it called as abstraction

**There are two ways to achieve abstraction in java**

- 
- i) Using abstract class
  - ii) Using interface

### **If we not write the logic of abstract method so where we can write the logic of abstract?**

---

if we want to write the logic of abstract method we need to inherit the abstract class in any another class and override the abstract method and write its logic.

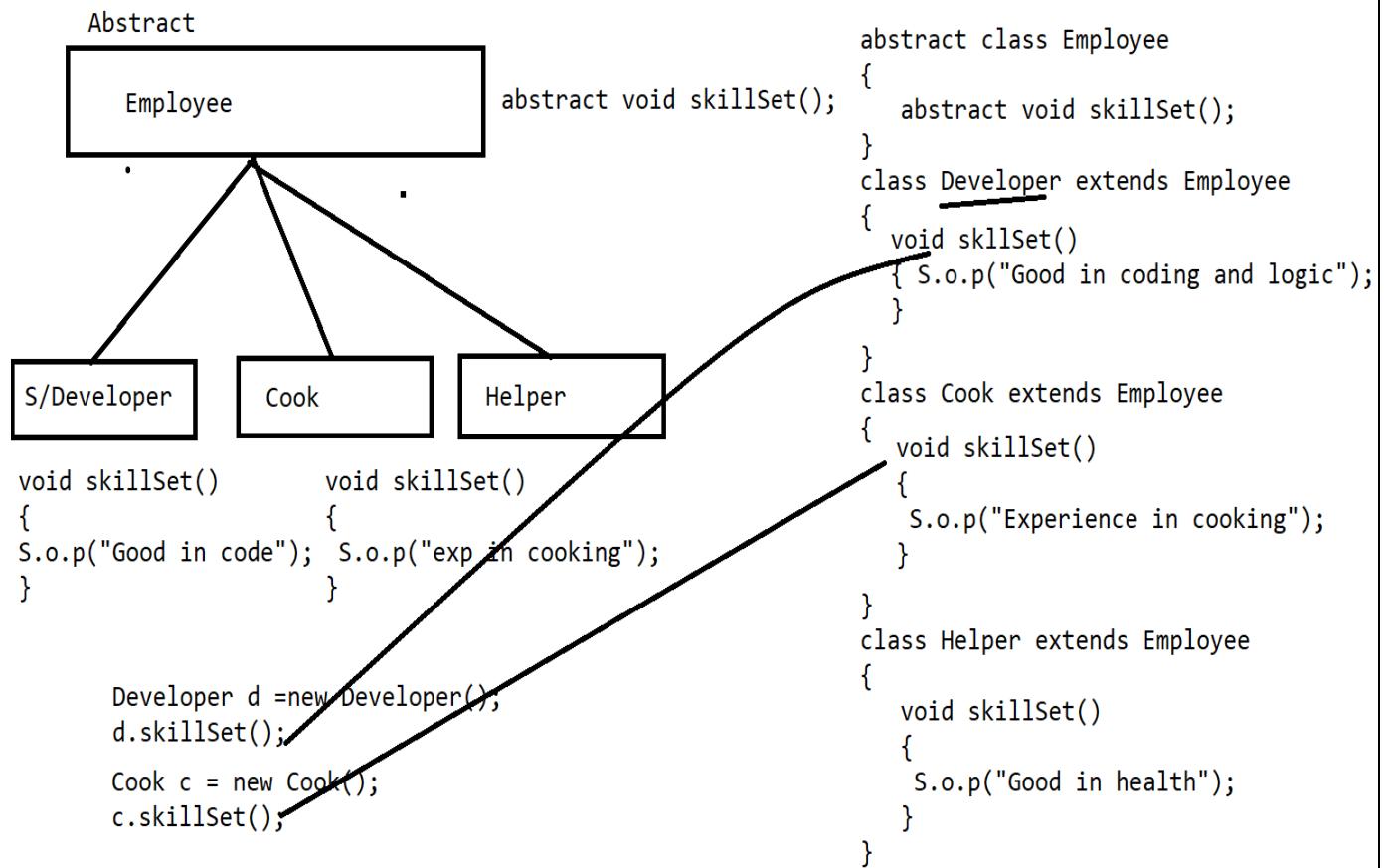
### **Q. What is the benefit of overriding abstract method in child class?**

---

The benefit is we can modify the or we can write the different logic in every abstract class according to its requirement

### Example

Suppose we want to hire the employee for different field but we required skillSet() for every employee but we cannot predict the skillSet() of employee. It is depend on in which field we want to hire employee.



**Note:** in the case of abstract class we not need to create the object of parent class. We can access the abstract member using its child class.

## Following code show example of abstract class

---

```
package org.techhub;
abstract class Employee {
    abstract void skillSet();
}
class Developer extends Employee {

    @Override
    void skillSet() {
        // TODO Auto-generated method stub
        System.out.println("Good in coding");
    }
}
class Cook extends Employee {
    @Override
    void skillSet() {
        // TODO Auto-generated method stub
        System.out.println("Exp in cooking");
    }
}
class Helper extends Employee
{
    @Override
    void skillSet() {
        // TODO Auto-generated method stub
        System.out.println("Good In Health");
    }
}
```

```
    }
}

public class AbstractApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Developer d = new Developer();
        d.skillSet();
        Cook c = new Cook();
        c.skillSet();
        Helper h = new Helper();
        h.skillSet();
    }
}
```

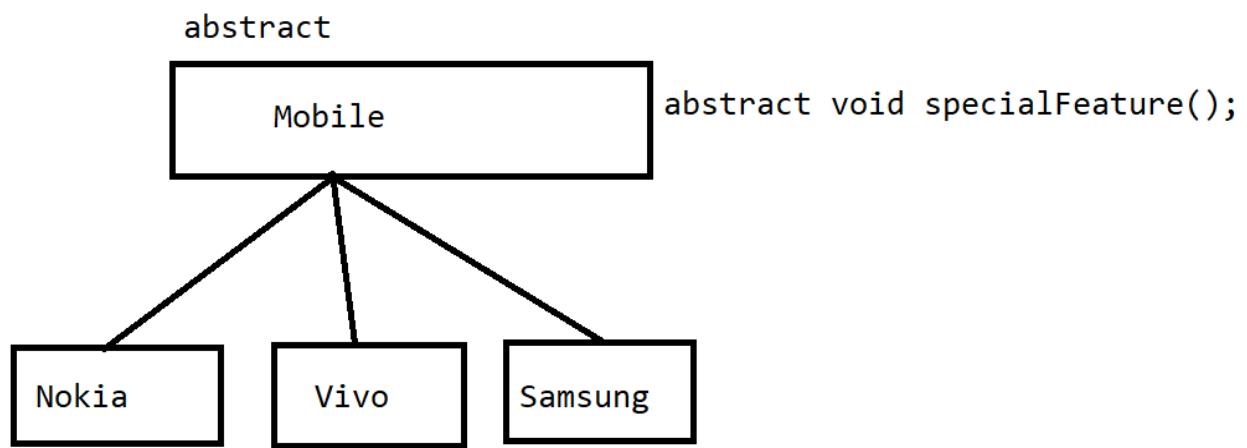
## **One More Example of Abstraction**

---

Suppose we want to purchase with some specialFeature()

Then we cannot predict the specialFeature() of mobile it is  
Vary from company to company

So here we need to declare the one abstract class name as Mobile  
With abstract method name as specialFeature () and we need to  
create the child classes of the mobile like as Nokia,Vivo etc  
and we can write the different logic of specialFeature in every  
mobile child class.



```
abstract class Mobile
{
    abstract void specialFeature();
}
class Nokia extends Mobile
{
    void specialFeature()
    { S.o.p("Good Battery Backup");}
}
class Vivo extends Mobile
{
    void specialFeature()
    { S.o.p("Good Camera Quality");}
}
class Samsung extends Mobile
{
    void specialFeature()
    { S.o.p("good in look");}
}
```

### Example

---

```
package org.techhub;
abstract class Mobile
{
    abstract void specialFeature();
}
class Nokia extends Mobile
{
    @Override
```

```
void specialFeature() {
    // TODO Auto-generated method stub
    System.out.println("Good In Battery Backup");
}
}
class Vivo extends Mobile
{
    @Override
    void specialFeature() {
        // TODO Auto-generated method stub
        System.out.println("Good In Camera Quality");
    }
}
public class MobileExampleWithAbstraction {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Nokia n = new Nokia();
        n.specialFeature();
        Vivo v = new Vivo();
        v.specialFeature();
    }
}
```

If we want to work with Abstract method and abstract class we have the some important points

---

- i) Abstract class cannot create its object
- ii) abstract method cannot have logic

**iii) If we have abstract method in class then class must be abstract**

**Example:**

```
class ABC
{ abstract void show();}
```

Above code is not valid because we cannot declare the abstract method in non abstract class. So if we want to declare the abstract method we have the correct given below

**Example:**

```
abstract class ABC
{
    abstract void show();}
```

**Note:** in abstract class there is possibility non abstract method may be exist

```
abstract class ABC{
    abstract void show();
    void display()
    {
    }}
```

If we declare the abstract and non abstract method in abstract class Then class called as concrete class

**iv) If abstract class contains more than one abstract methods then all method must be Override where abstract class get inherit.**

```
abstract class A
{
    abstract void show();
    abstract void display();
}

class B extends A
{
    //we have error because we not override display()
    //because A contain two abstract methods so we need to override
    //all methods

    void show()  //override all method if we not required
    {
        System.out.println("I required show");
    }
}

class C extends A
{
    //error we have error because we not override show()
    //because A contain two abstract methods so we need to override
    //all methods

    void display() //override all method if we not required.
```

```
{  
    System.out.println("I required display method");  
}  
}
```

If we want to solve above problem we must be override all abstract method in child class where we inherit the abstract class

abstract class A

```
{ abstract void show();  
abstract void display();  
}
```

class B extends A

```
{  
void show()  
{  
System.out.println("I required show");  
}  
void display()  
{  
}  
}
```

class C extends A

```
{  
void display()  
{  
System.out.println("I required display method");  
}
```

```
void show() {  
}  
}  
}
```

Example

---

```
package org.techhub;  
  
abstract class A  
{ abstract void show();  
    abstract void display();  
}  
class B extends A  
{  
    void show()  
    {  
        System.out.println("I required show method");  
    }  
  
    @Override  
    void display() {  
        // TODO Auto-generated method stub  
    }  
}  
class C extends A  
{  
    void display()  
    {  
        System.out.println("I required display method");  
    }  
  
    @Override  
    void show() {  
        // TODO Auto-generated method stub  
    }
```

```

        }
    }

public class AbsMethodRuleApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        B b1 = new B();
        b1.show();
        C c1 = new C();
        c1.display();
    }
}

```

**Note:** above approach is not good approach in coding

So if we want to solve above problem we have one more approach called as adapter class.

### **Q. What is the Adapter class?**

---

Adapter class is mediator class which contains all blank method definition of abstract class and which is able to provide specific method to its child class called as adapter class.

```

package org.techhub;
abstract class ABC
{
    abstract void s1();
    abstract void s2();
    abstract void s3();
    abstract void s4();
    abstract void s5();
}
class ADP extends ABC

```

```
{      @Override
    void s1() {
        // TODO Auto-generated method stub
    }
    @Override
    void s2() {
        // TODO Auto-generated method stub
    }
    @Override
    void s3() {
        // TODO Auto-generated method stub
    }
    @Override
    void s4() {
        // TODO Auto-generated method stub
    }
    @Override
    void s5() {      // TODO Auto-generated method stub
    }
}
class FChild extends ADP
{
    void s1()
    {
        System.out.println("I need s1 method");
    }
}
class SChild extends ADP
{
    void s2()
    {
        System.out.println("I need s2 method");
    }
}
```

```
        }
    }
public class AdapterImplementationApp
{
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        FChild fc= new FChild();
        fc.s1();
        SChild sc= new SChild();
        sc.s2();
    }
}
```

#### v) We cannot create object of abstract class but can create its reference

---

If we want to create reference of abstract class we need to create the object of its child class.

#### Example

---

```
package org.techhub;
abstract class Test
{
    abstract void show();
}

class TestChild extends Test
{
    @Override
    void show() {
        // TODO Auto-generated method stub
        System.out.println("I am abstract method");
    }
}

public class AbsRefApp {
    public static void main(String[] args) {
```

```
// TODO Auto-generated method stub
Test t = new TestChild();
t.show();
}
```

If we create the reference of abstract class using that reference we can call only those member declared within parent. We cannot access the any original member of child class using abstract class reference.

```
package org.techhub;
abstract class Test
{ abstract void show();
}

class TestChild extends Test
{ @Override
    void show() {
        // TODO Auto-generated method stub
        System.out.println("I am abstract method");
    }
    void display()
    {
        System.out.println("I am display method");
    }
}

public class AbsRefApp {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Test t = new TestChild();
        t.show();
        t.display();
    }
}
```

```
}
```

**Note:** above code generate the compile time error to us The method display() is undefined for the type Test Because display() is original method of TestChild class and we try to call it using Test i.e. abstract class reference so it is not possible to call child class method using parent reference so compiler generate the error to us

But if we create the object of child and reference of child class then using that reference we can call the parent member as well as child member.

Example

---

```
package org.techhub;
abstract class Test
{ abstract void show(); }
class TestChild extends Test
{ @Override
    void show() {
        // TODO Auto-generated method stub
        System.out.println("I am abstract method");
    }
    void display()
    {
        System.out.println("I am display method");
    }
}
public class AbsRefApp {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        TestChild t = new TestChild();
        t.show();
```

```
t.display();
}
}
```

### Note: What is the benefit of parent reference?

---

The Major benefit of parent reference is to achieve loose coupling.

### Q .what is the meaning of coupling?

---

Coupling means if we one class is dependent on another class called as coupling.

### There are two types of coupling?

---

- 1) **Tight Coupling:** tight coupling means if one class is 100% dependent on another class called as tight coupling.
- 2) **Loose Coupling:** loose coupling means if one class is partially dependent on another class called as loose coupling.

### Example

---

```
package org.techhub.loosecoupling;
import java.util.*;
abstract class Mobile
{
    abstract void specialFeature();
}
class Nokia extends Mobile
{
    @Override
    void specialFeature() {
        // TODO Auto-generated method stub
        System.out.println("Good Battery backup");
    }
}
```

```
    }

}

class Vivo extends Mobile
{
    @Override
    void specialFeature() {
        // TODO Auto-generated method stub
        System.out.println("Good Camera Quality");
    }

}

class RamShowRoom
{
    void saleMobile(Mobile mobile)
    {
        mobile.specialFeature();
    }
}

public class LooseCouplingApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        RamShowRoom r = new RamShowRoom();
        Scanner xyz = new Scanner(System.in);
        System.out.println("1:Nokia");
        System.out.println("2:Vivo");
        System.out.println("Enter your choice");
        int choice=xyz.nextInt();
        switch(choice)
        {
            case 1:
```

```
Mobile m=new Nokia();
r.saleMobile(m);
break;
case 2:
m=new Vivo();
r.saleMobile(m);
break;
default:
    System.out.println("wrong choice");
}
}
```

Example

---

```
package org.techhub.loosecoupling;
import java.util.*;
abstract class Value
{ int first,second;
    abstract void setValue(int x,int y);
    abstract void performOperation();
}
class Add extends Value
{ @Override
    void setValue(int x, int y) {
        // TODO Auto-generated method stub
        first=x;
        second=y;
    }
    @Override
    void performOperation() {
        // TODO Auto-generated method stub
        System.out.printf("Addition is %d\n",first+second);
    }
}
```

```
        }
    }
class Mul extends Value
{   @Override
    void setValue(int x, int y) {
        // TODO Auto-generated method stub
        first=x;
        second=y;
    }
    @Override
    void performOperation() {
        // TODO Auto-generated method stub
        System.out.printf("Multiplication is %d\n", first*second);
    }
}
class Sub extends Value
{   @Override
    void setValue(int x, int y) {
        first=x;
        second=y;
    }
    @Override
    void performOperation() {
        // TODO Auto-generated method stub
        System.out.printf("Substration is %d\n",first-second);
    }
}
class Calculator
{
    void performCalculation(Value value)
    {
        value.setValue(10, 20);
```

```
        value.performOperation();
    }
}

public class CalculateApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
Scanner xyz = new Scanner(System.in);
Calculator c = new Calculator();
int choice;
Value v=null;
System.out.println("Enter your choice");
choice=xyz.nextInt();
switch(choice)
{
case 1:
    v=new Add();
    c.performCalculation (v);
    break;
case 2:
    v =new Mul();
    c.performCalculation (v);
    break;
case 3:
    v=new Sub();
    c.performCalculation(v);
    break;
default :
    System.out.println ("Wrong choice");
}
}
}
```

## **Interface**

---

Interface is same like as abstract class in java

### **Q. Why use the interface?**

---

- 1) To Achieve 100% abstraction
- 2) To Achieve Dynamic Polymorphism
- 3) To Achieve Multiple Inheritance

### **Q. Why Java Not Use the Classes For Multiple Inheritance**

---

Java not use the classes for multiple inheritance because of diamond problem

### **Q. What is the diamond problem?**

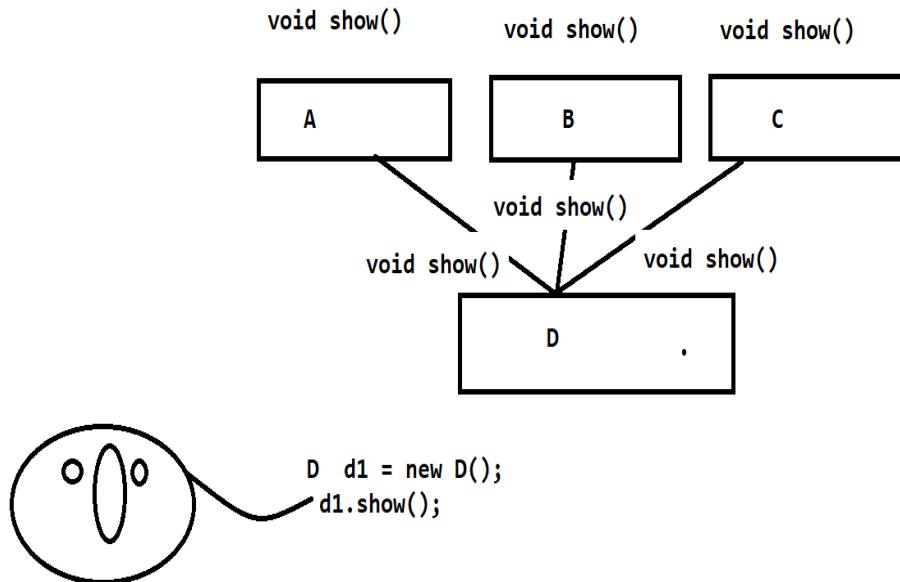
---

Multiple inheritances means more than one parent classes and single child class called as multiple inheritances.

In this case there is possibility in different parent may be contain the same name method and if we create object of child class and try to call the method whose name same in different parent class then compiler may be get confused called as diamond problem.

### **Example**

---



If we think about above diagram then we have the three different parent class names as A B and C

And we have only one child class name as D and Three parent class contain method name as show()

Which is inherited in D class and if we create the object of D class and try to call the method show ()

Then compiler gets confused and if we want to solve this problem we have the interface concept.

## **How to Declare the**

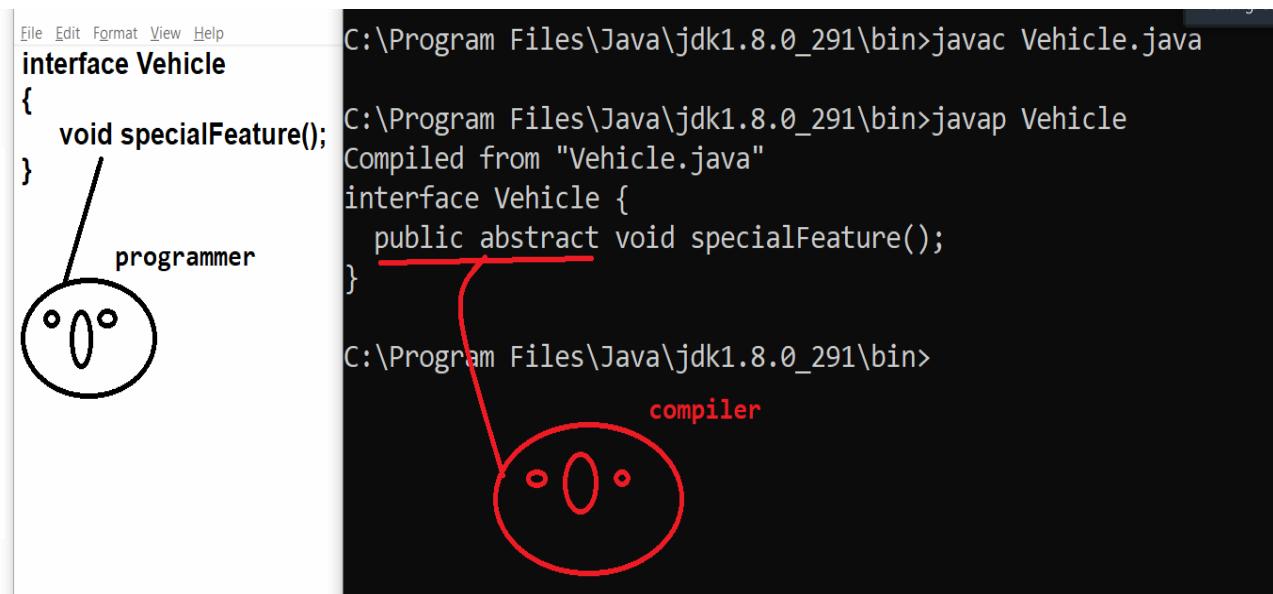
**syntax:**

```
interface interfacename
{
    returntype functionname(arguments);
}
```

Example

```
interface Vehicle
{
    void specialFeature();
}
```

**Note:** we cannot create the object of interface because it is internally abstract class and we cannot write the logic of interface method because it is internally public abstract and abstract method cannot have logic so we cannot write the logic of interface method.



## **Q. If interface method cannot have logic Then where we can Write logic of interface method?**

If we want to write the logic of interface method then we need to implement the interface in any another class and override its method and write its logic.

```
File Edit Format View Help
interface Vehicle
{
    void specialFeature(); //public abstract void specialFeature();
}
class Bike implements Vehicle
{
    public void specialFeature()
    {
        System.out.println("Need Two Wheel");
    }
}
public class InterfaceApplication
{
    public static void main(String x[])
    {
        Bike b = new Bike();
        b.specialFeature();
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>java InterfaceApplication
Need Two Wheel
```

## If we want to work with interface in java we have the some important points

- 1) Interface cannot create its object
- 2) Interface method cannot have definition
- 3) Interface variables are by default public static final  
Means if we want to declare the variable within interface we must have to initialize some value in it.  
If we declare the variable within interface and if we not initialize value then java compiler will generate error to us

```
interface ABC
{
    float PI=3.14f; //public static final PI
}
class MNO implements ABC
{
}
public class InterfaceApplication
{
    public static void main(String x[])
    {
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac InterfaceApplication.java
C:\Program Files\Java\jdk1.8.0_291\bin>java InterfaceApplication
C:\Program Files\Java\jdk1.8.0_291\bin>javap ABC.java
Error: class not found: ABC.java

C:\Program Files\Java\jdk1.8.0_291\bin>javap ABC
Compiled from "InterfaceApplication.java"
interface ABC {
    public static final float PI;
}

C:\Program Files\Java\jdk1.8.0_291\bin>
```

#### **4) If interface contain more than one methods then all method must be override where interface get implemented**

```
interface ABC
{
    public void s1();
    public void s2();
    public void s3();
}
class MNO implements ABC
{
    public void s1()
    { System.out.println("I required this method");
    }
    public void s2()
    {
    }
```

```
public void s3()
{
}
}

class PQR implements ABC
{
    public void s2()
    { System.out.println("I required this method");
    }
    public void s1()
    {
    }
    public void s3()
    {
    }
}

class STV implements ABC
{
    public void s2()
    { System.out.println("I required this");
    }
    public void s1()
    {
    }
    public void s3()
    {
    }
}

public class AbsInfApp
{
    public static void main(String x[])
    {
    }
}
```

```
}
```

```
}
```

**Note:** this is the major limitation of interface methods or abstract method in java.

If we want to solve this problem in java we have the adapter class concept.

### **Q. what is the adapter class ?**

Adapter class is intermediately class which contain the all methods of interface and it is able to provide the specific method to interface implementor classes called as adapter class.

### **Example**

interface ABC

```
{
```

```
    public void s1();
```

```
    public void s2();
```

```
    public void s3();
```

```
}
```

class ADP implements ABC

```
{
```

```
    public void s1()
```

```
    {
```

```
    }
```

```
    public void s2()
```

```
    {
```

```
    }
```

```
    public void s3()
```

```
    {
```

```
    }
```

```
}
```

class MNO extends ADP

```
{
```

```
public void s1()
{ System.out.println("I required this method");
}

}
class PQR extends ADP
{
    public void s2()
    { System.out.println("I required this method");
    }
}
class STV extends ADP
{
    public void s3()
    { System.out.println("I required this");
    }
}
public class AbsInfApp
{
    public static void main(String x[])
    {
        MNO m = new MNO();
        m.s1();
        PQR p = new PQR();
        p.s2();
        STV s=new STV();
        s.s3();
    }
}
```

**5) Interface cannot create its object but can create its reference if we want to create the reference of interface we must be create the object of its implementor class.**

The screenshot shows a Java code editor and a terminal window. The code editor contains the following Java code:

```
interface IP
{
    void show();
}

class IC implements IP
{
    public void show()
    { System.out.println("I am interface method");
    }
}

public class InterfaceRefApp
{
    public static void main(String x[])
    {
        IP i = new IC();
        i.show();
    }
}
```

The terminal window shows the command prompt and the output of running the program:

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac InterfaceRefApp.java
C:\Program Files\Java\jdk1.8.0_291\bin>java InterfaceRefApp
I am interface method
C:\Program Files\Java\jdk1.8.0_291\bin>
```

A small black box in the top right corner of the slide contains the text "Talking: GIR".

The Major goal of interface reference is to achieve dynamic polymorphism or loose coupling.

## Example of Loose Coupling

```
import java.util.*;
interface Value
{ void setValue(int x,int y);
  void performOperation();
}
class Add implements Value
{ int first,second;
  public void setValue(int x,int y)
  {
    first=x;
    second=y;
  }
}
```

```
public void performOperation()
{
    System.out.println("Addition is "+(first+second));
}
}
class Mul implements Value
{ int first,second;
  public void setValue(int x,int y)
  {
    first=x;
    second=y;
  }
  public void performOperation()
  { System.out.println("Multiplication is "+(first*second));
  }
}
class Calculator
{
  void performCal(Value v)
  {
    v.setValue(100,200);
    v.performOperation();
  }
}
public class LooseCouplingApp
{
  public static void main(String x[])
  {
    Scanner xyz = new Scanner(System.in);
    Value v=null;
    Calculator c = new Calculator();
    System.out.println("1:Addition");
```

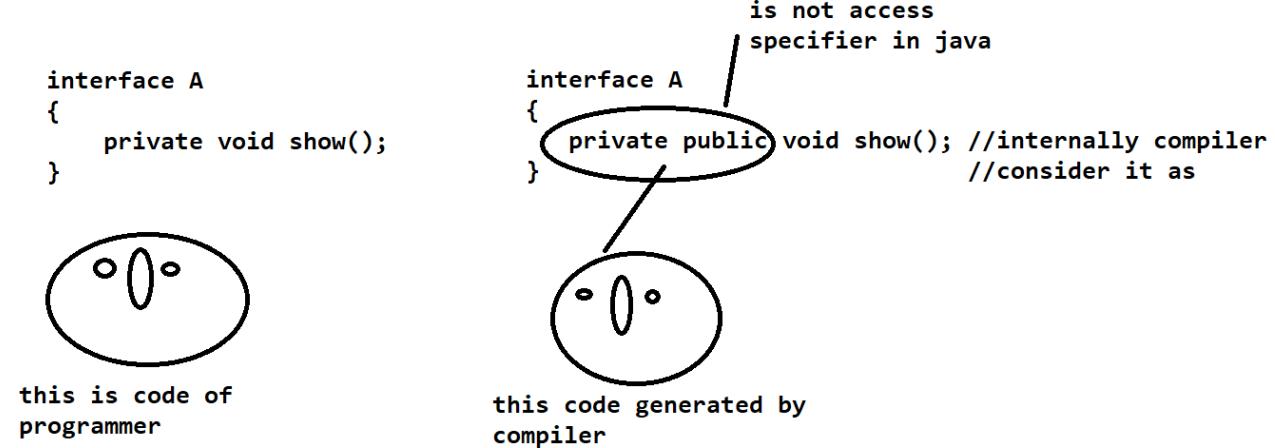
```
System.out.println("2:Multiplication");
System.out.println("Enter your choice");
int choice=xyz.nextInt();
switch(choice)
{
    case 1:
        v = new Add();
        c.performCal(v);
        break;
    case 2:
        v = new Mul();
        c.performCal(v);
        break;
    default:
        System.out.println("Wrong choice");
}
}
```

## 6) We cannot declare the interface method as private, protected, static and final

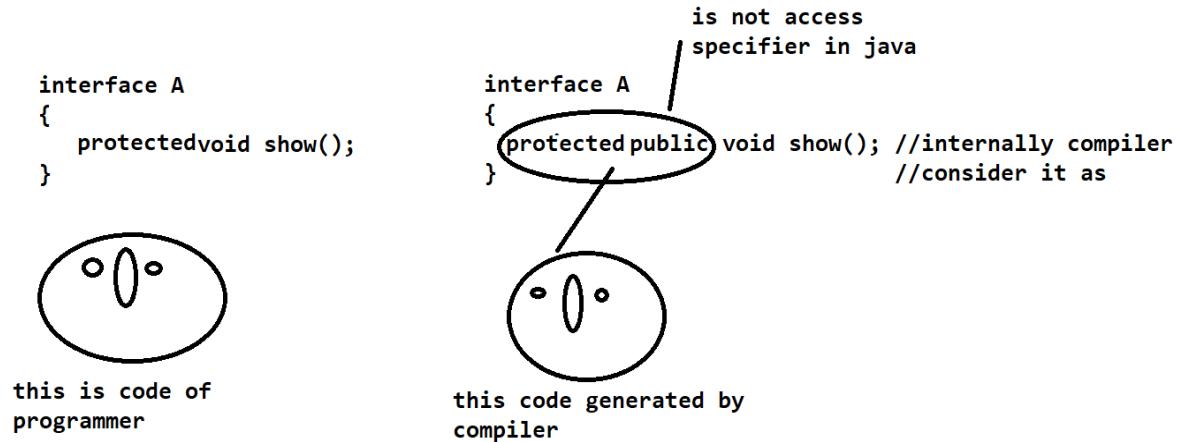
**private:** there are two reason we cannot declare interface method as private.

- i) private method cannot support to inheritance as well as not support to overriding and interface cannot work without inheritance and overriding.
- ii) interface is method is by default public and if we declare it as private then internally compiler consider it as private public and in private public is not access specifier so we cannot declare interface method as private.

Above statement meaning shown in following diagram



**Protected:** we cannot declare interface method as protected because interface method is by default public and if we declare it as protected then internally compiler consider it as protected so in java there is no public protected access specifier.



**final:** we cannot declare interface method as final because final method cannot override in child class and interface method must be

override in child class so we cannot declare interface method as final

**static:** static method must have definition and interface method cannot have definition.

## 7) If we want to inherit the interface to interface we have the extends keyword

```
File Edit Format View Help
interface A
{
    void show();
}
interface B extends A
{ void display();
}
class C implements B
{
    public void show()
    { System.out.println("I am show method");
    }
    public void display()
    { System.out.println("I am display method");
    }
}
public class InterfaceInheritance
{ public static void main(String x[])
{
    C c1 = new C();
    c1.show();
    c1.display();
}
}
```

Output:

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac InterfaceInheritance.java  
C:\Program Files\Java\jdk1.8.0_291\bin>java InterfaceInheritance  
I am show method  
I am display method  
C:\Program Files\Java\jdk1.8.0_291\bin>
```

## **How To Achieve Multiple Inheritance using java**

Multiple Inheritance means more than one parent and single child called as multiple inheritance.

In multiple parent there should be only one parent class and remaining are interfaces.

### **Example**

```
interface A  
{  
    void show();  
}  
interface B  
{  
    void show();  
}  
class C  
{  
    void show()  
    { System.out.println("I am c method");  
    }  
}  
class D extends C implements A,B  
{  
    public void show()
```

```
{  
    System.out.println("I am D method");  
}  
}  
public class MultipleApp  
{  
    public static void main(String x[])  
    {  
        D d1 = new D();  
        d1.show();  
    }  
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac MultipleApp.java  
C:\Program Files\Java\jdk1.8.0_291\bin>java MultipleApp  
I am D method
```

## Collection Framework

### Q. What is the Collection?

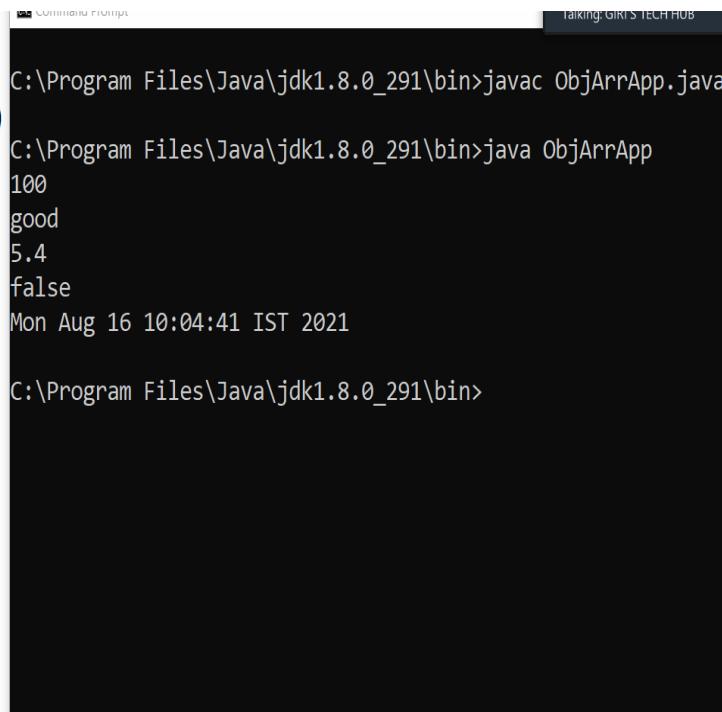
Collection is ready made implementation of data structure provided by java to us.

### Q. Why use the Collection or what is the benefit of Collection?

- 1) Ability to store the different type of data
- 2) Collection can change its dynamic size at run time as well as shrink its size at run time.
- 3) Collection can provide the readymade data structure algorithm like as inserting element, deleting element, searching element etc

**Note:** if we use the Object class array in java then we can store different type of data in it also.

```
public class ObjArrApp
{
    public static void main(String x[])
    {
        Object obj[]=new Object[5];
        obj[0]=100;
        obj[1]="good";
        obj[2]=5.4f;
        obj[3]=false;
        obj[4]=new java.util.Date();
        for(int i=0; i<obj.length; i++)
        {
            System.out.println(obj[i]);
        }
    }
}
```



The screenshot shows a terminal window titled 'Terminal' with the command 'javac ObjArrApp.java' followed by 'java ObjArrApp'. The output displays the following values: 100, good, 5.4, false, and Mon Aug 16 10:04:41 IST 2021.

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac ObjArrApp.java
C:\Program Files\Java\jdk1.8.0_291\bin>java ObjArrApp
100
good
5.4
false
Mon Aug 16 10:04:41 IST 2021
```

If we think about above diagram then we can store the different type of data in Object class array.

## **But there is some limitation**

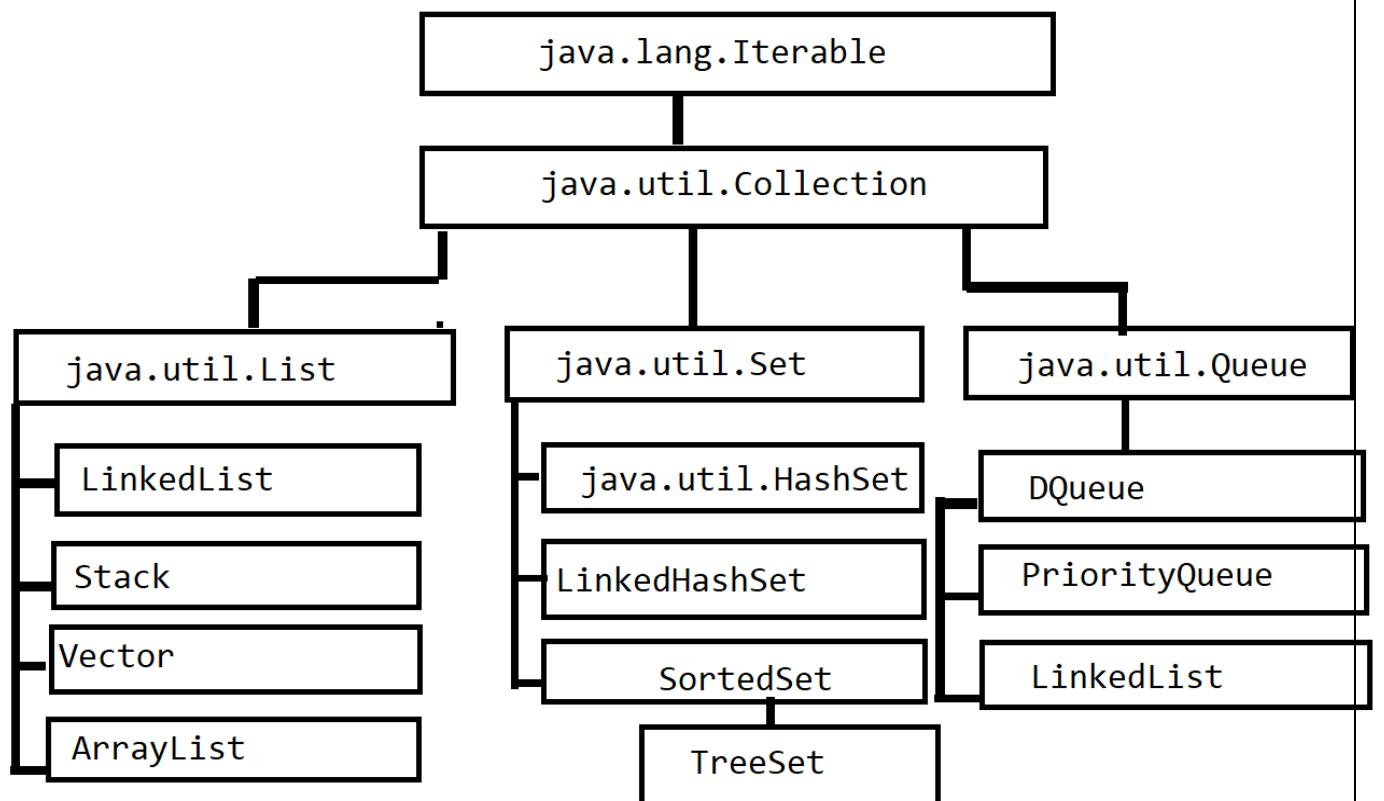
---

1) If we want to perform any data structure operation we need to perform manually like as inserting data on specified position, deleting data from specified position, searching data etc

If we want to avoid this problem in java we have the Collection Framework.

**If we want to work with Collection Framework in java we have the Following Class Hierarchy**

---



If we think about above diagram we have the top most interface name as `java.lang.Iterable`

## **What is the Role of Iterable interface in Collection API?**

---

The Major goal of Iterable interface is to fetch data from collection framework. Because data fetching is common operation in Collection API so they design the Iterable as parent to all and provide the methods to us for data fetching purpose. Iterable is interface from `java.lang` package and which provide the `iterator()` method to us for data fetching purpose and iterator method return the reference of `Iterator()` interface.

**Internally Iterable Look Like as**

---

### **Syntax**

---

```
public interface java.lang.Iterable<T> {  
    public abstract java.util.Iterator<T> iterator();  
    public void forEach(java.util.function.Consumer<? super T>);  
    public java.util.Spliterator<T> spliterator();  
}
```

Note: `forEach()` and `splitIterator()` we will discuss later

### **Iterator interface**

---

Iterator interface return by `iterator()` method declared within Iterable interface and Iterator interface provide the following method to us for fetching data from any type of Collection

**boolean hasNext():** this method check wheather element present in collection or not if present return true otherwise return false.

**Object next():** this method is used for fetch data from collection and move cursor on next element.

**void remove():** this method can remove the data from collection at the time of data fetching.

### Following Code Example Describe The Working of Iterator interface

---

```
Stack s = new Stack();
s.add(100);
s.add(200);
s.add(300);
s.add(400);
```



```
Iterator i = s.iterator();
```

```
while(i.hasNext())
{
    Object obj = i.next();
```

Output

\_\_\_\_\_

100

### Working of Above Code

---

In above code we create the object of Stack class and store the four element in 100,200 ,300 and 400 using add method() of Stack. When we call the statement `Iterator i = s.iterator()` means your reference of Iterator i.e `i` in our example points to Stack.

### **While loop working shown below**

---

As per our example first Iterator reference i.e `i` points 100 element it is first element in stack. Means `while(i.hasNext())` first check 100 present in stack or not so in our example 100 is present in Stack so `i.hasNext()` method return true means your loop will be execute and then next statement get executed i.e `i.next()` this method can fetch element from collection and move cursor on next element i.e 200 as per our example so again while loop get executed means `while(i.hasNext())` get executed and again `while(i.hasNext())` check second element present in stack or not means it will check 200 as per our example so 200 present in stack so this statement return true and then again loop executed and `i.next()` method get executed and this method fetch 200 from stack and move cursor on next element i.e 300 an so on when all element fetch by Iterator and if element not present in stack then `while(i.hasNext())` method return false and loop get terminated.

### **Why Iterable provide the iterator() and refence of Iterator interface**

---

The Major benefit of iterator it is parent of all types of collection means we can use the Iterator interface by using `iterator()` method with all types of collection. Means we can use the above mention logic with stack for data fetching with all types of collection

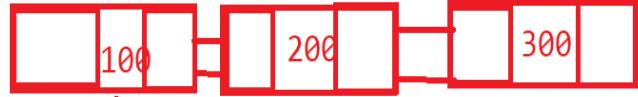
## Sample code Example

---

```
LinkedList lst=new LinkedList();
lst.add(100);
lst.add(200);
lst.add(300);
```

```
lst.add(400);
```

```
Iterator i = lst.iterator();
while(i.hasNext())
{
    Object obj = i.next();
    System.out.println(obj);
}
```



this logic is common  
for all types of  
collection for data  
fetching purpose

In collection hierarchy we have the second top most interface is `java.util.Collection`

### Q. why `java.util.Collection` interface is parent of all collection APIs

---

Collection provide the methods to us for common operation with collection means Collection interface contain methods which is commonly required to all types of collection.

#### Common operation with collection

---

1) adding element collection

2) removing element from collection

- 3) search element from collection
- 4) count the number of element in collection

Etc

Means above operation is required to all types of collection means above operation can perform with stack, queue, linkedlist ,vector,array etc So Collection api design the Collection interface and provide the common method to all types of collection for perform above operation.

## Methods of Collection interface

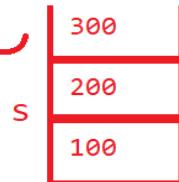
---

```
public interface java.util.Collection<E> extends java.lang.Iterable<E> {  
    public abstract int size();  
    public abstract boolean isEmpty();  
    public abstract boolean contains(java.lang.Object);  
    public abstract java.util.Iterator<E> iterator();  
    public abstract java.lang.Object[] toArray();  
    public abstract <T> T[] toArray(T[]);  
    public abstract boolean add(E);  
    public abstract boolean remove(java.lang.Object);  
    public abstract boolean containsAll(java.util.Collection<?>);  
    public abstract boolean addAll(java.util.Collection<? extends E>);  
    public abstract boolean removeAll(java.util.Collection<?>);  
    public boolean removeIf(java.util.function.Predicate<? super E>);  
    public abstract boolean retainAll(java.util.Collection<?>);  
    public abstract void clear();  
    public abstract boolean equals(java.lang.Object);  
    public abstract int hashCode();  
    public java.util.Spliterator<E> spliterator();  
    public java.util.stream.Stream<E> stream();  
    public java.util.stream.Stream<E> parallelStream();  
}
```

Now we will discuss about method of Collection interface

**int size()**: this method is used return number of element present in collection.

```
Stack s = new Stack();
s.add(100);
s.add(200);
s.add(300);
    3
int nelement = s.size();
```



```
System.out.println("Number of element " + nelement);
```

**Output:**

```
Number of element : 3
```

**boolean isEmpty()**: this method check wheather collection is empty or not if empty return true otherwise return false.

```
ArrayList al = new ArrayList();
al.add(100);
al.add(200);
al.add(300);
    false
boolean b = al.isEmpty();
```



```
if(b) if(false)
{
    System.out.println("ArrayList is      empty");
}
else
{
    System.out.println("ArrayList is not empty");
}
```

**Output:** ArrayList is not empty

If we think above code then we call the isEmpty() method of ArrayList and ArrayList contain 3 element so it is not empty so isEmpty() method return false in variable b and if we pass b in if statement so if return false so else get executed so we get output ArrayList is not empty.

**boolean contains(Object value):** this method is used for searching purpose means if we want to search specific element in collection framework then we can use the contains method and this method return true if we found the element in collection otherwise return false.

```
ArrayList al = new ArrayList();
al.add(100);
al.add(200);
al.add(300);
    true
boolean b = al.contains(200);

if(b) if(true)
{
    System.out.println("element found");
}
else
{
    System.out.println("Element not found");
}

Output :
Element Found
```



If we think about above code we pass 200 value in contains method and 200 present in ArrayList so this method return true in variable b and we pass variable b in if statement means if contain true value so if get executed and we found the element found.

**boolean add(E):** this method is used for add the element in collection if element added then return true otherwise return false.

```
ArrayList al = new ArrayList();
    true
boolean b=al.add(100);           100
if(b) if(true)
{
    System.out.println("Element Added ");
}
else
{
    System.out.println("Eleement Not Added");
}
```

**Output:**

---

Element Added.

If we think about above code we call the add() method with 100 value when element added in collection this method return true in variable b and we pass b in if means if contain true so we get output element Added

**boolean remove(Object value):** this method can remove the element from collection and if element remove from collection return true otherwise return false.

```

ArrayList al = new ArrayList();
al.add(100);
al.add(200);
al.add(300);

boolean b = al.remove(200);

if(b)
{
    System.out.println("Element Removed Success");
}
else
{
    System.out.println("element not removed");
}

```

If we think about above code we pass the 200 value in remove() method and this value present in ArrayList and remove() method remove the value from ArrayList and return true value in variable b and we pass the b in if statement and b contain true means if also true and we get output element removed success.

Basically there are three types of collection in java

## **Types of Collection in java**

---

**1) List:** List collection can store the any type of data and infinite size and arrange data using indexing format but allow the duplicated values or data.

**2) Set:** Set Collection can store any type of data and infinite size and manage data using hasing technique and not allow the duplicate elements.

**3) Queue:** Queue is collection can manage data in first in first out format and arrange data using indexing and allow the duplicated data

**Now we will Discuss about the List Collection**

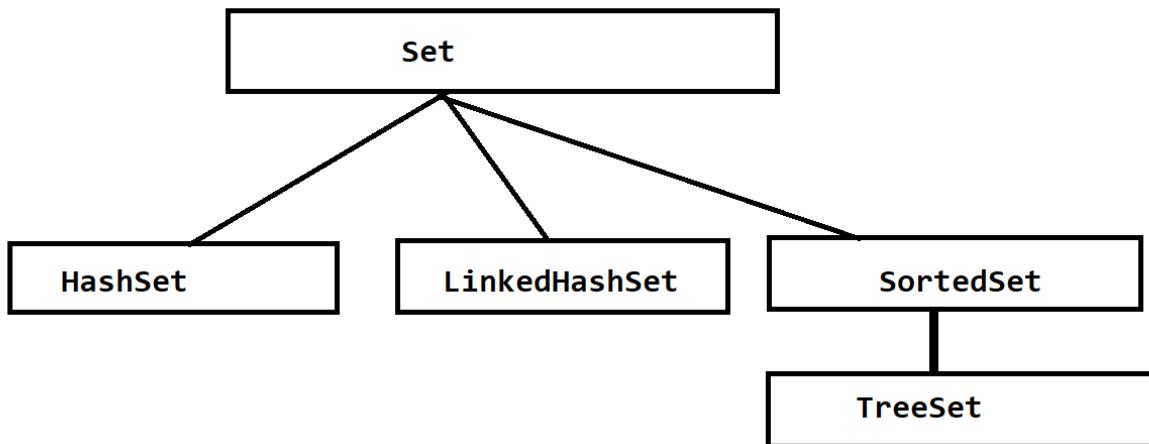
---

## **Set Collection**

Set Collection which is used for store the unique values and store the data by using hashing technique called as Set Collection Set cannot maintain the index for element.

### **Three Types of Set Collection**

**Following Diagram shows the hierarchy of Set collection**



**1) HashSet:** HashSet store the unique elements and Generate the element sequence randomly

**2) LinkedHashSet:** LinkedHashSet store the unique values and generate the element sequence according to the user sequence.

**3) TreeSet:** TreeSet is used for store the unique values and generate the all values in ascending order.

**Now we want to create the application to perform basic operation on Set collection using HashSet**

1. Add Element in HashSet
2. View All element from HashSet
3. Search Element in HashSet
4. remove element from HashSet
5. Count the all element of HashSet

Etc

### **Example of HashSet**

```
package org.techhub;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        HashSet hs = new HashSet();
        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("1:Add Element in HashSet");
            System.out.println("2:View All Element from HashSet");
            System.out.println("3:Search Element in HashSet");
            System.out.println("4:Remove element from HashSet");
            System.out.println("5:Count the element from HashSet");
            System.out.println("Enter your choice");
            int choice = xyz.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Enter the value in set");
                    int val = xyz.nextInt();
                    hs.add(val);
                    break;
                case 2:
                    Iterator i = hs.iterator();
                    while (i.hasNext()) {
                        Object obj = i.next();
                        System.out.println(obj);
                    }
                    break;
                case 3:
                    System.out.println("Enter the search value");
                    int sval = xyz.nextInt();
                    boolean b = hs.contains(sval);
                    if (b) {
                        System.out.println("Element found");
                    } else {
                        System.out.println("element not found");
                    }
            }
        }
    }
}
```

```

        break;
case 4:
    System.out.println("Enter value for delete");
    val = xyz.nextInt();
    b = hs.remove(val);
    if (b) {
        System.out.println("Value removed success...");
    } else {
        System.out.println("Value not removed");
    }
    break;
case 5:
System.out.println("Number of element in collection " + hs.size());
    break;
default:
    System.out.println("Wrong choice");

}
} while (true);// infinite loop

}

```

}

---

Example of LinkedHashSet

```

package org.techhub;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LinkedHashSet hs = new LinkedHashSet();
        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("1:Add Element in HashSet");
            System.out.println("2:View All Element from HashSet");
            System.out.println("3:Search Element in HashSet");
            System.out.println("4:Remove element from HashSet");

```

```
System.out.println("5:Count the element from HashSet");
System.out.println("Enter your choice");
int choice = xyz.nextInt();
switch (choice) {
case 1:
    System.out.println("Enter the value in set");
    int val = xyz.nextInt();
    hs.add(val);
    break;
case 2:
    Iterator i = hs.iterator();
    while (i.hasNext()) {
        Object obj = i.next();
        System.out.println(obj);
    }
    break;
case 3:
    System.out.println("Enter the search value");
    int sval = xyz.nextInt();
    boolean b = hs.contains(sval);
    if (b) {
        System.out.println("Element found");
    } else {
        System.out.println("element not found");
    }
    break;
case 4:
    System.out.println("Enter value for delete");
    val = xyz.nextInt();
    b = hs.remove(val);
    if (b) {
        System.out.println("Value removed success...");
    } else {
        System.out.println("Value not removed");
    }
    break;
case 5:
```

```

        System.out.println("Number of element in collection " + hs.size());
        break;
    default:
        System.out.println("Wrong choice");

    }
}

} while (true); // infinite loop

}

```

## Example of TreeSet Collection

---

```

package org.techhub;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        TreeSet hs = new TreeSet();
        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("1:Add Element in HashSet");
            System.out.println("2:View All Element from HashSet");
            System.out.println("3:Search Element in HashSet");
            System.out.println("4:Remove element from HashSet");
            System.out.println("5:Count the element from HashSet");
            System.out.println("Enter your choice");
            int choice = xyz.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Enter the value in set");
                    int val = xyz.nextInt();
                    hs.add(val);
                    break;
                case 2:
                    Iterator i = hs.iterator();
                    while (i.hasNext()) {

```

```
        Object obj = i.next();
        System.out.println(obj);
    }
    break;
case 3:
    System.out.println("Enter the search value");
    int sval = xyz.nextInt();
    boolean b = hs.contains(sval);
    if (b) {
        System.out.println("Element found");
    } else {
        System.out.println("element not found");
    }
    break;
case 4:
    System.out.println("Enter value for delete");
    val = xyz.nextInt();
    b = hs.remove(val);
    if (b) {
        System.out.println("Value removed success...");
    } else {
        System.out.println("Value not removed");
    }
    break;
case 5:
    System.out.println("Number of element in collection " + hs.size());
    break;
default:
    System.out.println("Wrong choice");
}
} while (true); // infinite loop
}
}
```

## How To Arrange data in descending order using TreeSet

If we want to arrange data in descending order using TreeSet we have the

descendingSet() method of TreeSet and this method return the reference of NavigableSet interface and NavigableSet interface is used for fetch data of Set using descending order.

```
package org.techhub;
import java.util.*;
public class TreeSetApplication {

    public static void main(String[] args)
    {
        // TODO Auto-generated method stub
        TreeSet ts = new TreeSet();
        ts.add(4);
        ts.add(1);
        ts.add(20);
        ts.add(33);
        ts.add(2);
        ts.add(5);
        NavigableSet nav=ts.descendingSet();
        Iterator i=nav.iterator();
        while(i.hasNext())
        { Object obj = i.next();
            System.out.println(obj);
        }
    }
}
```

## **Mini Project Application By using HashSet**

---

WAP to create Bag with ball collection and we want to add the balls in bag and we want extract the single ball from back but we want to extract the ball randomly and we want to show the information with ball color, company ,weight ,size

### **1) Create the POJO class for store the ball information**

```
package org.techhub.ball;
```

```
public class Ball {  
  
    private String color;  
    public String getColor() {  
        return color;  
    }  
    public void setColor(String color) {  
        this.color = color;  
    }  
    public int getWeight() {  
        return weight;  
    }  
  
    public void setWeight(int weight) {  
        this.weight = weight;  
    }  
    public int getSize() {  
        return size;  
    }  
    public void setSize(int size) {  
        this.size = size;  
    }  
    public String getCompany() {  
        return company;  
    }  
    public void setCompany(String company) {  
        this.company = company;  
    }  
  
    private int weight;  
    private int size;  
    private String company;  
}
```

**2) create the class name as BallExtraction and declaration the void addBall() method in it and add the new ball in HashSet collection**

```

package org.techhub.ball;
import java.util.*;
public class BallExtraction {
    HashSet bag = new HashSet();
    public void addBall(Ball ball) {
        bag.add(ball);
    }
}

```

**3)** create the class with main method and in main method we have to create the object of BallExtraction first and then we have to create the object of Ball and store the information in ball object and pass to ball object to BallExtraction for store in collection.

```

package org.techhub.ball;
import java.util.*;
public class BallExtractionApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        BallExtraction be = new BallExtraction();
        do {
            Scanner xyz = new Scanner(System.in);
            int choice;
            System.out.println("1:Add Ball in Bag");
            System.out.println("2:View All Balls in bag");
            System.out.println("3:Extract the ball from bag");
            System.out.println("enter your choice");
            choice = xyz.nextInt();
            switch (choice) {
                case 1:
                    Ball ball = new Ball();
                    xyz.nextLine();
                    System.out.println("enter the ball color");

```

```
String colorName = xyz.nextLine();
System.out.println("enter the ball company");
String compName = xyz.nextLine();
System.out.println("Enter the ball weight");
int weight = xyz.nextInt();
System.out.println("Enter the ball size");
int size = xyz.nextInt();
ball.setColor(colorName);
ball.setCompany(compName);
ball.setWeight(weight);
ball.setSize(size);
be.addBall(ball);
break;
case 2:
break;
case 3:
break;
default:
System.out.println("Wrong choice");
}
} while (true);

}
```

**4) create the method viewAllBalls() in BallExtraction and fetch all data from collection and display it**

### **BallExtraction.java**

---

```
package org.techhub.ball;

import java.util.*;

public class BallExtraction {
    HashSet bag = new HashSet();
```

```
public void addBall(Ball ball) {  
    bag.add(ball);  
}  
public void viewAllBalls()  
{  
    Iterator i=bag.iterator();  
    while(i.hasNext())  
    {  
        Object obj = i.next();  
        Ball b =(Ball)obj;  
        System.out.println(b.getColor()+"\t"+b.getCompany()+"\t"+b.getSize()  
+ "\t"+b.getWeight());  
    }  
}
```

### BallExtractionApplication.java

---

```
package org.techhub.ball;  
  
import java.util.*;  
  
public class BallExtractionApplication {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        BallExtraction be = new BallExtraction();  
        do {  
            Scanner xyz = new Scanner(System.in);  
            int choice;  
            System.out.println("1:Add Ball in Bag");  
            System.out.println("2:View All Balls in bag");  
            System.out.println("3:Extract the ball from bag");  
        }  
    }  
}
```

```
System.out.println("enter your choice");
choice = xyz.nextInt();
switch (choice) {
    case 1:
        Ball ball = new Ball();
        xyz.nextLine();
        System.out.println("enter the ball color");
        String colorName = xyz.nextLine();
        System.out.println("enter the ball company");
        String compName = xyz.nextLine();
        System.out.println("Enter the ball weight");
        int weight = xyz.nextInt();
        System.out.println("Enter the ball size");
        int size = xyz.nextInt();
        ball.setColor(colorName);
        ball.setCompany(compName);
        ball.setWeight(weight);
        ball.setSize(size);
        be.addBall(ball);
        break;
    case 2:
        be.viewAllBalls();
        break;
    case 3:
        break;
    default:
        System.out.println("Wrong choice");
}
} while (true);

}
```

5) Write the method in BallExtraction class name as pickUpBall() and fetch ball from collection randomly

## BallExtraction.java

---

```
package org.techhub.ball;

import java.util.*;

public class BallExtraction {
    HashSet bag = new HashSet();

    public void addBall(Ball ball) {
        bag.add(ball);
    }

    public void viewAllBalls()
    {
        Iterator i=bag.iterator();
        while(i.hasNext())
        {
            Object obj = i.next();
            Ball b =(Ball)obj;

            System.out.println(b.getColor()+"\t"+b.getCompany()+"\t"+b.getSize()
+ "\t"+b.getWeight());
        }
    }

    public void pickUpBall()
    {

        Iterator i=bag.iterator();
        if(i.hasNext())
        {
            Object obj = i.next();
            Ball b=(Ball)obj;
            System.out.println(b.getColor()+"\t"+b.getCompany());
            i.remove();
        }
        else
        {
            System.out.println("You bag is empty no ball found");
        }
    }
}
```

```
        }
    }
}
```

---

### BallExtractionApplication.java

---

```
package org.techhub.ball;

import java.util.*;

public class BallExtractionApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        BallExtraction be = new BallExtraction();
        do {
            Scanner xyz = new Scanner(System.in);
            int choice;
            System.out.println("1:Add Ball in Bag");
            System.out.println("2:View All Balls in bag");
            System.out.println("3:Extract the ball from bag");
            System.out.println("enter your choice");
            choice = xyz.nextInt();
            switch (choice) {
                case 1:
                    Ball ball = new Ball();
                    xyz.nextLine();
                    System.out.println("enter the ball color");
                    String colorName = xyz.nextLine();
                    System.out.println("enter the ball company");
                    String compName = xyz.nextLine();
                    System.out.println("Enter the ball weight");
                    int weight = xyz.nextInt();
                    System.out.println("Enter the ball size");
                    int size = xyz.nextInt();
                    ball.setColor(colorName);
                    ball.setCompany(compName);
                    ball.setWeight(weight);
                    ball.setSize(size);
            }
        }
    }
}
```

```
        be.addBall(ball);
        break;
    case 2:
        be.viewAllBalls();
        break;
    case 3:
        be.pickUpBall();
        break;
    default:
        System.out.println("Wrong choice");
    }
} while (true);

}
```

## Collections Class

---

Collections class is utility class from java.util package which is specially design for perform the operations on List Collection or Collection. Collections class provides the some static method to us to perform operation on Collection.

**Example:** suppose consider we have the ArrayList and we want to perform sorting with ArrayList Then Collections class provide the sort () method to us to perform sorting on ArrayList.

### **Q. What is the diff between Collection and Collections?**

---

- 1) Collection is interface from java.util package and Collections is class from java.util package.
- 2) Collection interface provide the implementation of data structure algorithm and Collections class is utility class which provide the static method to us to perform operation Collection Framework like as finding max element from collection, min element from collection, sort the collection etc these operation perform by Collections class.

### **Methods of Collections class**

---

**public static void sort(List):** this method is used for sort the data from list collection. This is the overloaded method

**public static void sort(List, Comparator):** this method is used for sort the user defined objects .

**public static void reverse(List):** this method is used for reverse the list collection

**public T min(java.util. Collection):** this method can find the minimum element from collection

**public T max(java.util. Collection) :** this method is used for find the max element from collection.

Etc

### **Now we want to use the Collections.sort () method**

---

This method is used for perform the sorting on List Collection.

Now we have the simple example we have the List collection with some values and we want to perform sorting on List Collection

Example

```
package org.techhub;
import java.util.*;
public class CollectionsApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        List list = new ArrayList();
        list.add(100);
        list.add(400);
        list.add(300);
        list.add(200);
        list.add(900);
        list.add(700);
        list.add(50);
        System.out.println("Before Sorting");
        for(Object obj:list)
        {
            System.out.println(obj);
        }
        Collections.sort(list); //this method perform sorting operation
        automatically on Collection Framework
        System.out.println("After Sorting");
        for(Object obj:list)
        {
            System.out.println(obj);
        }
    }
}
```

We want to find the max element from List collection

---

For that we have the max() method of Collections class and it is also static method of Collections class.

```
package org.techhub;
import java.util.*;
public class CollectionsApplication {
```

```
public static void main(String[] args) {  
  
    List list = new ArrayList();  
    list.add(100);  
    list.add(400);  
    list.add(300);  
    list.add(200);  
    list.add(900);  
    list.add(700);  
    list.add(50);  
    System.out.println("Before Sorting");  
    for(Object obj:list)  
    {  
        System.out.println(obj);  
    }  
    System.out.println("Max Element is "+Collections.max(list));  
    System.out.println("Min Element is "+Collections.min(list));  
}  
}
```

**WAP to create the collection of Employee Data and sort the all employee by using its id.**

**We want to employee detail empid, empname, empsal**

---

### Steps to implement the above program

---

**1) Create the POJO class name as Employee with three fields id , name and sal**

---

```
package org.techhub;  
public class Employee {  
    private int id;  
    private String name;  
  
    public int getId() {
```

```
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getSal() {
        return sal;
    }
    public void setSal(int sal) {
        this.sal = sal;
    }
    private int sal;
}
```

## **2) Create the class with main method and create the object of ArrayList**

---

```
package org.techhub;
import java.util.*;
public class EmployeeStorageApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList al =new ArrayList();

    }
}
```

## **3) Create the objects of Employee class and store in ArrayList**

```
package org.techhub;
import java.util.*;
public class EmployeeStorageApplication {
```

```

public static void main(String[] args) {
    // TODO Auto-generated method stub
    ArrayList al =new ArrayList();
    Employee emp[]=new Employee[5]; //array of reference
    for(int i=0; i<emp.length;i++)
    {
        Scanner xyz = new Scanner(System.in);
        System.out.println("Enter the name id and salary of employee");
        String name=xyz.nextLine();
        int id=xyz.nextInt();
        int sal=xyz.nextInt();
        emp[i]=new Employee(name,id,sal); //Array of objects
        al.add(emp[i]);
    }
    System.out.println("Employee Records before sorting");
    for(Object obj:al)
    {
        Employee e =(Employee)obj;
        System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
    }
    Collections.sort(al);
    System.out.println("Employee Records After sorting");
    for(Object obj:al)
    {
        Employee e =(Employee)obj;
        System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
    }
}
}

```

**Note: above code generate the runtime exception to us**

**Why?**

Because we use the Collections.sort() method in above code and we store the Employee class object in ArrayList so Collections.sort() method by default sort the list collection if collection contain primitive data type but here in our above code store the employee class object in ArrayList so Employee class

object is customize object it is not primitive data type so we get the exception at run time

### **Why Collections.sort () method not sort the by default user defined class object?**

---

User defined object contain different type of data so Collections class cannot predict the sorting technique on object so it will raise at program run time means as per our example we have ArrayList and in ArrayList we store Employee class objects and Employee contain the three different type of data means it contain String ,integer and salary so we cannot predict sorting perform by using id or sorting perform by using salary or sorting perform by name so Collections.sort() method not perform sorting directly on User defined objects.

### **How to perform sorting on user defined by using Collections.sort () method**

---

If we want to perform sorting on user defined objects we have the two interfaces in java

- 1) Comparable interface
- 2) Comparator interface.

#### **Comparable interface**

---

Comparable interface is used for perform the sorting with user defined objects with the help of Colloections.sort() method.

#### **Steps to work with Comparable interface**

---

##### **1) add the java.lang package in application**

**Note:** java.lang is default package of java so have to no need to import it

##### **2) Create the POJO class and implement the Comparable interface in it.**

```
package org.techhub;  
public class Employee implements Comparable {
```

```
private int id;
private String name;

public Employee()
{
}

public Employee(String name,int id,int sal)
{
    this.name=name;
    this.id=id;
    this.sal=sal;
}
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public int getSal() {
    return sal;
}
public void setSal(int sal) {
    this.sal = sal;
}
private int sal;
}
```

**3) override the compareTo() method of Comparable interface in POJO class and perform the comparision**

If we want to work with `compareTo()` method we have the some following rules.

- 1) If current object is greater than specified object then return positive integer
- 2) If current object is less than specified object then return negative integer
- 3) if current object is equal to specified object then return zero

Example

---

```
package org.techhub;
public class Employee implements Comparable {
    private int id;
    private String name;

    public Employee()
    {

    }

    public Employee(String name,int id,int sal)
    {
        this.name=name;
        this.id=id;
        this.sal=sal;
    }
    public int getId()
    {
        return id;
    }
    public void setId(int id)
    {
        this.id = id;
    }
    public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        this.name = name;
    }
    public int getSal()
    {
        return sal;
    }
}
```

```
    }
    public void setSal(int sal) {
        this.sal = sal;
    }
    private int sal;

    @Override
    public int compareTo(Object o) {
        // TODO Auto-generated method stub
        Employee e=(Employee)o;
        if(this.id>e.id)
        {
            return 1;
        }
        else if(this.id<e.id)
        {return -1;
        }
        else
        {
            return 0;
        }
        return 0;
    }
}
```

#### Main method class

---

```
package org.techhub;
import java.util.*;
public class EmployeeStorageApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList al =new ArrayList();
        Employee emp[]=new Employee[5]; //array of reference
        for(int i=0; i<emp.length;i++)
        { Scanner xyz = new Scanner(System.in);
            System.out.println("Enter the name id and salary of employee");
        }
    }
}
```

```

String name=xyz.nextLine();
int id=xyz.nextInt();
int sal=xyz.nextInt();
emp[i]=new Employee(name,id,sal); //Array of objects
al.add(emp[i]);
}
System.out.println("Employee Records before sorting");
for(Object obj:al)
{
    Employee e =(Employee)obj;
    System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
}
Collections.sort(al);
System.out.println("Employee Records After sorting");
for(Object obj:al)
{
    Employee e =(Employee)obj;
    System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
}
}
}

```

**Note:** The major limitation of Comparable interface is to can't perform the sorting using more than one attribute of object.

## Comparator interface

Comparator interface is used for perform the sorting with multiple attribute of objects

Suppose consider in above program we sort the employee record by using id of employee using Comparable interface by developer A and we have one more developer B and B want to sort the employee records by using its salary And if B developer try to modify the logic in compareTo() method of Comparable interface so Developer A logic may get vanish and if we want to solve this problem we have the one more interface for sorting purpose name as Comparator interface.

## **Steps to works with Comparator interface**

---

- 1) add the java.util package in application**
- 2) create the one more class and implement the Comparator interface in it and override its compare() method and write the comparison logics**

```
package org.techhub;
import java.util.Comparator;
public class SortEmployeeBySal implements Comparator {
    @Override
    public int compare(Object o1, Object o2) {
        // TODO Auto-generated method stub
        Employee emp1 = (Employee) o1;
        Employee emp2 = (Employee) o2;
        if (emp1.getSal() > emp2.getSal()) {
            return 1;
        } else if (emp1.getSal() < emp2.getSal()) {
            return -1;
        } else {
            return 0;
        }
    }
}
```

## **3) use the Collections.sort() in following fashion**

---

When we use the Comparator interface we have to use the Collections.sort () method in given fashion.

**Syntax: Collections.sort (List,Comarator):** as per this syntax we have to pass first parameter as list collection and second parameter child class object of Comparator interface.

```
package org.techhub;
import java.util.*;
public class EmployeeStorageApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}
```

```
ArrayList al =new ArrayList();
Employee emp[] =new Employee[5]; //array of reference
for(int i=0; i<emp.length;i++)
{ Scanner xyz = new Scanner(System.in);
System.out.println("Enter the name id and salary of employee");
String name=xyz.nextLine();
int id=xyz.nextInt();
int sal=xyz.nextInt();
emp[i]=new Employee(name,id,sal); //Array of objects
al.add(emp[i]);
}
SortEmployeeBySal s=new SortEmployeeBySal();
Collections.sort(al,s);
System.out.println("Employee Records After sorting");
for(Object obj:al)
{
Employee e =(Employee)obj;
System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
}
}
```



## List Collection

---

List Collection allows the duplicated data and can store infinite number of data and manage data by using index technique like as array.

### Methods of List Collection or Interface

---

**public abstract E get (int)** : this method can retrieve data from List collection using its index.

### Example

---

```
ArrayList al = new ArrayList();
0   1   2   3
al.add(100);
al.add(200);
al.add(300);
al.add(400);

for(int i=0; i<al.size(); i++)
{
    Object obj = al.get(i);

    System.out.println(obj);
}
```

The diagram shows an ArrayList named 'al' with four elements: 100, 200, 300, and 400. The elements are represented as boxes with their values inside. Above each element is its index: 0, 1, 2, and 3. The indices are aligned under the first element, and the boxes are aligned under the second element, and so on.

### Code Description:

---

In above code when we execute the statement `for(int i=0; i<al.size(); i++)` Here `al.size()` method return the size of collection as per our example it is 4 means statement work like as `for(int i=0;i<4;i++)`

When we execute the statement `al.get(i)` means first our `i` is 0 in for loop so statement like as `al.get(0)` means fetch data from 0<sup>th</sup> index of ArrayList when loop second time then `i` increase by 1 means 1 is less than 4 and statement get executed `al.get(1)` means fetch second position data from ArrayList and so on

**public abstract E set(int, E)** : this method is used for replace the element in collection on specified index.

```

ArrayList al = new ArrayList();
al.add(100);
al.add(200);
al.add(300);
al.add(400);

```

0	1	2	3
100	200	300	400

Before Replacement ArrayList

Here we replace the 300 value on second position of ArrayList

0	1	2	3
100	200	600	400

**public abstract void add(int, E):** this method can add the element on specified index in Collection and move the remaining element on next index.

```

ArrayList al = new ArrayList();
al.add(100);
al.add(200);
al.add(300);
al.add(400);
al.add(1,1000);

```

0	1	2	3
100	200	300	400

Before inserting value

we added 1000 element on 1st index of ArrayList and shifted previous element by index position

0	1	2	3	4
100	1000	200	300	400

**public abstract E remove(int):** this method can remove the element using some specified index

**public abstract int indexOf(java.lang.Object):** this method can return the index of specified element in collection Normally we use this method for searching purpose in collection if element found return its index and if element not found return -1

```

ArrayList al = new ArrayList();
0   1   2   3
al.add(100);
al.add(200);
al.add(300);
al.add(400);

2
int index = al.indexOf(300); //this method return the index of 300
i.e 2

```

## Where we can use this method or scenario where we can use this method

---

**1) Search the element from collection:** indexOf() method return the index of particular element from collection and if element not found return -1 means when we found the -1 we can declare element not present in collection and if found any value other than -1 we can consider element present in collection.

```

ArrayList al = new ArrayList();
0   1   2   3
al.add(100);
al.add(200);
al.add(300);
al.add(400);

Scanner xyz = new Scanner(System.in);
System.out.println("Enter the search value\n");
int value =xyz.nextInt();

int index = al.indexOf(value);
if(index!=-1)
{
    System.out.println("Element Found");
}
else
{
    System.out.println("Element Not Found");
}

```

If we think about above code then we have the for input

int value=xyz.nextInt() and we provide input 200 then next statement get executed i.e int index = al.indexOf(value); means we pass 200 value of

indexOf() method e.g int index=al.indexOf(200) and in our example 200 having 1<sup>st</sup> index

then we have the next statement is if(index!=-1) means the index of 200 is 1 so our if statement look like as if(1!=-1) so this is the true condition so we get out put element found

suppose consider we provide the 600 value to indexOf() method means your code look like as int index = al.indexOf(600) so it is not present in ArrayList collection so it will return -1 in index so our if statement look like as if(-1!=-1) so it is false condition and our else get executed and we found the output element not found.

**public abstract java.util.ListIterator<E> listIterator(int):** this method can travel the List collection in forward and in backward direction

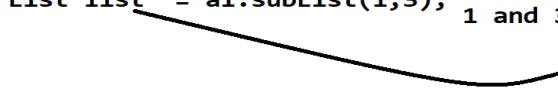
**Note:** example of this method we will discuss later in chapter.

**public abstract java.util.List<E> subList(int, int) :** it is used for extract the some specified position of List collection between two indexes.

```
ArrayList al = new ArrayList();
al.add(100);
al.add(200);
al.add(300);
al.add(400);
al.add(500);
```

0	1	2	3	4
100	200	300	400	500

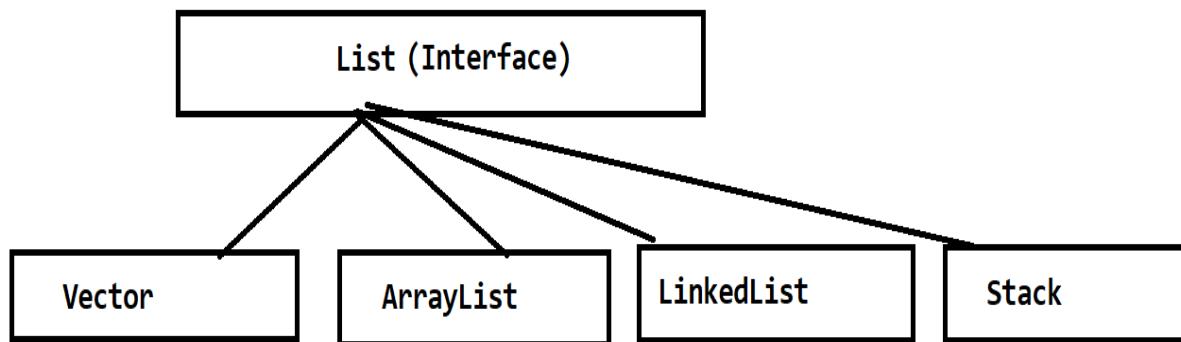
```
List list = al.subList(1,3); we extract the List collection between
1 and 3 index.
```



1	2	3
200	300	400

## **There are major four implementor classes of List Collection**

---



**Now we will discuss about the Vector class**

### **Q.what is the Vector?**

---

- 1) Vector is legacy collection and implement of List collection
- 2) Vector is popular as dynamic array
- 3) Vector is synchronized and Thread safe collection

### **Q.what is the legacy Collection ?**

---

Legacy collection means those classes is not part of collection in previous version but later they added as part of collection called as legacy collection. Means before jdk 1.2 Vector is not part of Collection but from jdk1.2 version of java Vector added as part collection so people say it is a legacy collection. If we want to work with Vector or create the object of Vector we have the three types of constructor means Vector having overloaded constructor.

### **Constructors of Vector class**

---

**Vector():** this constructor create the object of Vector class with default capacity provided by java collection framework. Default Capacity of Vector is 10 element.

*If we want to see the capacity of Vector we have the int capacity() method of Vector class.*

### **Following Example display the capacity of Vector**

---

**package org.techhub;**

```

import java.util.*;
public class VectorApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Vector v = new Vector();
        System.out.println("Capacity of vector is "+v.capacity());
    }
}

```

Output

```

<terminated> VectorApplication [Java Application]
Capacity of vector is 10
|
```

If we want to set the default capacity of Vector as per your choice we have the second constructor

**Vector(int initialCapacity)**: this constructor is used for create the vector with initial capacity provided by java collection framework.

```

1 package org.techhub;
2
3 import java.util.*;
4 public class VectorApplication {
5
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8
9         Vector v = new Vector(5);
10        System.out.println("Capacity of vector is "+v.capacity());
11    }
12
13 }
```

```

<terminated> VectorApplication [Java Application] C:\Users\Admin
Capacity of vector is 5
```

**Note :** if we use the Vector then Vector occupy doble memory than its current capacity if capacity of Vector is cross.

```

package org.techhub;
import java.util.*;
```

```
public class VectorApplication {  
    public static void main(String[] args) {  
        Vector v = new Vector(5);  
        System.out.println("Capacity of vector is "+v.capacity());  
        v.add(100);  
        v.add(200);  
        v.add(300);  
        v.add(400);  
        v.add(500);  
        v.add(600);  
  
        System.out.println("After Capacity Increment "+v.capacity());  
    }  
}
```

Output

---

```
Capacity of vector is 5  
After Capacity Increment 10
```

---

If we want to decide the incremented capacity according to your choice then we have the third constructor Vector given below

### **Vector(int intialCapacity,int incrementedCapacity)**

---

**int initialCapacity :** this parameter is used for set up the initial capacity of Vector.

**int incrementalCapacity :** this parameter is used for set the incremental capacity of Vector after its current capacity cross.

```
import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {
        Vector v = new Vector(5,2);
        System.out.println("Capacity of vector is "+v.capacity());
        v.add(100);
        v.add(200);
        v.add(300);
        v.add(400);
        v.add(500);
        v.add(600);

        System.out.println("After Capacity Increment "+v.capacity());
    }
}
```

## Output

---

```
Capacity of vector is 5
After Capacity Increment 7
```

---

In above example we set the first parameter as 5 means the initial capacity of Vector is 5 and second parameter as 2 means incremental capacity of Vector is 2

Means when we add the sixth element in Vector then next capacity is 7 and when we add the 8<sup>th</sup> element in Vector then next capacity is 9 and so on.

## How Vector Constructor work internally

---

Vector constructor work by using constructor chaining concept internally.

Wh

```
class Vector
{ Object elementData[];
    int capacityIncrement;
    Vector()
    {
        this(10);
    }
    Vector(int initialCapacity)
    { this(initialCapacity,0);
    }
    Vector(int initialCapacity,int capacityIncrement)
    {
        if(initialCapacity<0)
            return throw new IllegalArgumentException("Invalid capacity "+initialCapacity);
        this.elementData=new Object[initialCapacity];
        this.capacityIncrement=capacityIncrement;
    }
}
```

## **Basic operations with Vector**

---

1. add element
- 2) fetch element
- 3) search element
- 4) remove element
- 5) count the element in Vector
- 6) update element
- 7) insert element on specified position

## **WAP using Vector class to perform following operations**

---

- case 1: Add the new element in collection
- case 2: show all data from Vector
- case 3: search element from vector
- case 4: remove element from vector
- case 5: count the number of element present in Vector
- case 6: update or replace element on specified index in Vector
- case 7 : insert the element on specified position in Vector

## **Example**

---

```
package org.techhub;
import java.util.*;
```

```
public class VectorApplication {  
    public static void main(String[] args) {  
        int choice;  
        Vector v = new Vector();  
        do {  
            Scanner xyz = new Scanner(System.in);  
            System.out.println("Enter your choice");  
            choice = xyz.nextInt();  
            switch (choice) {  
                case 1:  
                    System.out.println("Enter the value in collection");  
                    int val = xyz.nextInt();  
                    v.add(val);  
                    break;  
                case 2:  
                    for (int i = 0; i < v.size(); i++) {  
                        System.out.println(v.get(i));  
                    }  
                    break;  
                case 3:  
                    System.out.println("Enter the Element for searching in Collection");  
                    val = xyz.nextInt();  
                    boolean b = v.contains(val);  
                    if (b) {  
                        System.out.println("Element Found " + val);  
                    } else {  
                        System.out.println("Element not found ");  
                    }  
                    break;  
                case 4:  
                    System.out.println("Enter the element for remove");  
                    int value = xyz.nextInt();  
                    int index = v.indexOf(value);  
                    if (index != -1) {  
                        v.remove(index);  
                    } else {  
                        System.out.println("Element Not Present in collection remove");  
                    }  
            }  
        }  
    }  
}
```

```

        break;
    case 5:
        System.out.println("Number of element in Vector " + v.size());
        break;
    case 6:
        System.out.println("enter the index for replacement");
        index = v.size();
        if (index <= (v.size() - 1)) {
            System.out.println("Enter the value for replacement");
            val = xyz.nextInt();
            v.set(index, val);
        } else {
            System.out.println("Index not present");
        }
        break;
    case 7:
        System.out.println("enter the index for replacement");
        index = v.size();
        if (index <= (v.size() - 1)) {
            System.out.println("Enter the value for replacement");
            val = xyz.nextInt();
            v.insertElementAt(val, index);
        } else {
            System.out.println("Index not present");
        }
        break;
    case 8:
        System.exit(0);
        break;
    default:
        System.out.println("Wrong choice");
    }
} while (true);
}

```

## Cursors in Collection or Iterators in Collection

---

Cursor or Iterators is used in collection for fetching data from collection.

## **There are major five types of Cursor in Collection**

---

- i) Enumeration
- ii) Iterator
- iii) ListIterator
- iv) for each in jdk 1.5 version of java
- v) for in jdk 1.8 version of java

### **Enumeration:**

---

Enumeration is read only cursor means we cannot perform any modification on Collection using Enumeration

It only works with legacy collection classes.

Example: Vector , HashTable,Dictionary etc

If we want to create reference of Collection we have the elements() method of Collection this method return reference of Enumeration interface.

**Syntax: Enumeration ref = collref.elements();**

**Enumeration interface provide the following method to us for fetch data from collection**

**boolean hasMoreElements():** this method check wheather element present in collection or not if present return true otherwise return false.

**Object nextElement():** this method can fetch data from collection and move cursor on next element.

### **Example**

---

```
import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {

        Vector v = new Vector();
        v.add(100);
        v.add(200);
        v.add(300);
        Enumeration enm = v.elements();
```

```
while(enm.hasMoreElements())
{
    Object obj = enm.nextElement();
    System.out.println(obj);
}
}
```

## **Output**

---

```
100
200
300
```

## **Iterator**

---

**Iterator is used for fetch data from collection it work only with forward direction**

### **Q.what is diff between the Enumeration And Iterator**

---

- a) Enumeration is read only cursor and Iterator is not read only means Iterator can remove the eleemnt from collection at the time data fetching
- b) Iterator can work legacy as well as non legacy collection and Enumeration only works with legacy collection
- 3) Enumeration contain two methods boolean hasMoreElement() and Object nextElement() and Iterator contain three methods boolean hasNext(),Object next() and void remove();

If we want to create the reference of Iterator we have to use the iterator() method of Iterable interface and this method return reference of Iterator interface.

## **Now We want to create program to fetch data from collection using Iterator**

---

```
package org.techhub;
import java.util.*;
public class DemoDataFetching
{
    public static void main(String x[])
    {
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        Iterator i = al.iterator();
        while(i.hasNext())
        {
            Object obj = i.next();
            System.out.println(obj);
        }
    }
}
```

### Output

---

```
<terminated>
100
200
300
400
```

### 3) ListIterator :

---

ListIterator is used for fetch data in forward direction as well as in backward direction.

### Q. what is the diff between ListIterator and Iterator ?

---

The Major diff between Iterator and ListIterator is

- 1) Iterator can fetch data in forward direction but ListIterator can fetch data in forward as well as in backward direction
- 2) Iterator can remove the data from collection at the time of data fetching but ListIterator can remove,replace,add data in collection at the time of data fetching
- 3) ListIterator can only works with List Collection and Iterator can work with any collection in java.
- 4) methods of ListIterator  
boolean hasNext(),boolean hasPrevious(),Object next(),Object previous(),  
void remove(Object),void add(Object),void replace(Object) etc  
methods of Iterator boolean hasNext() ,Object next() and void remove()
- 5) ListIterator is child of Iterator interface.

**If we want to create the reference of ListIterator we have the method name as `listIterator(int index)`:**

**Syntax:** ListIterator ref = collectionref.listIterator(int index);

### **Example**

---

```
package org.techhub;
import java.util.*;
public class DemoDataFetching
{
    public static void main(String x[])
    {
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        ListIterator li = al.listIterator(al.size());
        while(li.hasPrevious())
        {   Object obj = li.previous();
```

```
        System.out.println(obj);
    }
}
}
```

#### 4) fetch data using for each statement of jdk 1.5 version

---

```
package org.techhub;
import java.util.*;
public class DemoDataFetching
{
    public static void main(String x[])
    {
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        for(Object obj:al)
        {
            System.out.println(obj);
        }
    }
}
```

#### 5) fetch data using jdk 1.8 version of foreach

---

```
package org.techhub;
import java.util.*;
public class DemoDataFetching
{
    public static void main(String x[])
    {
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        al.forEach(val->System.out.println(val));
    }
}
```

## Examples

---

### WAP to store the 5 values in Vector and calculate its addition

**Note:** Collection can store data in the form of Object class means Object is parent of all classes in java so Collection can store any type of data in it but when we fetch data from collection and perform any operation on it then we need to perform type conversion from Object class to its specified type.

We cannot perform any operation on Object class directly like as addition, multiplication ,division etc

```
package org.techhub;
import java.util.*;
public class DemoDataFetching
{
    public static void main(String x[])
    {
        Vector v = new Vector();
        v.add(100);
        v.add(200);
        v.add(300);
        v.add(400);
        v.add(500);
        Iterator i = v.iterator();
        Integer sum =0;
        while(i.hasNext())
        {
            Object obj = i.next();
            sum = sum +(Integer)obj;
        }
        System.out.printf("Sum is %d\n",sum);
    }
}
```

## Output

---

```
<terminated> DemoDataFetching [Java Application]
```

```
Sum is 1500
```

In above code have the `Object obj = i.next()` and `sum = sum+(Integer)obj`. We convert Object class in to the integer form because Object can hold data but we cannot perform any operation on object so perform operation on object we have to convert the Object class in to the specified its type. So we convert Object class reference to Integer value.

## **WAP to create the collection of Employee Data and show it**

---

### **Steps**

---

#### **1) create the pojo class name as Employee**

---

#### **Sample code**

---

```
class Employee
{ private int id;
  private String name;
  private int sal;
  public void setId(int id)
  { this.id=id;
  }
  public int getId()
  { return id;
  }
  public void setName(String name)
  { this.name=name;
  }
  public String getName()
  { return name;
  }
  public void setSal(int sal)
  { this.sal=sal;
  }
  public int getSal()
  { return sal;
  }
```

```
}
```

## 2) create the object of Vector class

---

```
package org.techhub;
import java.util.*;
public class DemoDataFetching
{
    public static void main(String x[])
    {
        Vector v = new Vector();
    }
}
```

## 3) Create the object of Employee pojo classe and store data in it using setter method

---

```
import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {

        Vector v = new Vector();
        Employee emp1 = new Employee();
        emp1.setId(1);
        emp1.setName("Ram");
        emp1.setSal(10000);

        Employee emp2 = new Employee();
        emp2.setId(2);
        emp2.setName("Shyam");
        emp2.setSal(20000);

        Employee emp3 = new Employee();
        emp3.setId(3);
        emp3.setName("Dinesh");
        emp3.setSal(10000);;
```

```
}
```

#### 4) store the all pojo objects in collection

```
package org.techhub;

import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {

        Vector v = new Vector();
        Employee emp1 = new Employee();
        emp1.setId(1);
        emp1.setName("Ram");
        emp1.setSal(10000);

        Employee emp2 = new Employee();
        emp2.setId(2);
        emp2.setName("Shyam");
        emp2.setSal(20000);

        Employee emp3 = new Employee();
        emp3.setId(3);
        emp3.setName("Dinesh");
        emp3.setSal(10000);

        v.add(emp1);
        v.add(emp2);
        v.add(emp3);

    }
}
```

#### 5) Fetch Data from collection using Iterator

```
package org.techhub;

import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {
```

```

Vector v = new Vector();
Employee emp1 = new Employee();
emp1.setId(1);
emp1.setName("Ram");
emp1.setSal(10000);
Employee emp2 = new Employee();
emp2.setId(2);
emp2.setName("Shyam");
emp2.setSal(20000);

Employee emp3 = new Employee();
emp3.setId(3);
emp3.setName("Dinesh");
emp3.setSal(10000);
v.add(emp1);
v.add(emp2);
v.add(emp3);
Iterator i = v.iterator();
while(i.hasNext())
{
    Object obj = i.next();
    System.out.println(obj.getId()+"\t"+obj.getName()+"\t"+obj.getSal());
}
}

```

**Note:** if we think about above code it will generate the compile time error to us because we store the Employee class object in Vector collection and when fetch data then we get data in Object class but when we want to use any data then we have to convert the data in its original format means as per our example we have to convert Object class in to the Employee class and then we can fetch data using employee object with getter methods.

### **Your Correct Code is given below**

---

```

package org.techhub;
import java.util.*;
public class VectorApplication {
    public static void main(String[] args) {

```

```

        Vector v = new Vector();

```

```

Employee emp1 = new Employee();
emp1.setId(1);
emp1.setName("Ram");
emp1.setSal(10000);
Employee emp2 = new Employee();
emp2.setId(2);
emp2.setName("Shyam");
emp2.setSal(20000);
Employee emp3 = new Employee();
emp3.setId(3);
emp3.setName("Dinesh");
emp3.setSal(10000);

v.add(emp1);
v.add(emp2);
v.add(emp3);

Iterator i = v.iterator();
while(i.hasNext())
{
    Object obj = i.next();
    Employee emp=(Employee)obj;//we convert Object to Employee
    class because Collection contain employee data
}

```

```

System.out.println(emp.getId()+"\t"+emp.getName()+"\t"+emp.getSal());
}
}
}

```

## **Output**

---

<b>1</b>	<b>Ram</b>	<b>10000</b>
<b>2</b>	<b>Shyam</b>	<b>20000</b>
<b>3</b>	<b>Dinesh</b>	<b>10000</b>

**WAP program create the class name as student with id, name and per and store its objects in Collection and display the only students whose percentage is greater than 70**

## **Steps**

---

**1) create the pojo class name as Student with three field name id and per**

```
package org.techhub;

public class Student
{
    private int id;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPer() {
        return per;
    }
    public void setPer(float per) {
        this.per = per;
    }
    private String name;
    private float per;
}
```

**2) Create the new class with main and create the object of Vector class**

---

```
package org.techhub;
import java.util.*;
public class StudentVectorApplication {

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

```
// TODO Auto-generated method stub
Vector v= new Vector();

}

}
```

### **3)Create the object of Student and set data in it**

---

```
package org.techhub;
import java.util.*;
public class StudentVectorApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Vector v= new Vector();

        Student s1 = new Student();
        s1.setId(1);
        s1.setName("Ram");
        s1.setPer(60.5f);

        Student s2 = new Student();
        s2.setId(2);
        s2.setName("dinesh");
        s2.setPer(50.5f);

        Student s3 = new Student();
        s3.setId(3);
        s3.setName("Rajesh");
        s3.setPer(80.5f);

    }
}
```

### **4) Add the students object in Vector class**

```
package org.techhub;
import java.util.*;
public class StudentVectorApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Vector v= new Vector();

        Student s1 = new Student();
        s1.setId(1);
        s1.setName("Ram");
        s1.setPer(60.5f);

        Student s2 = new Student();
        s2.setId(2);
        s2.setName("dinesh");
        s2.setPer(50.5f);

        Student s3 = new Student();
        s3.setId(3);
        s3.setName("Rajesh");
        s3.setPer(80.5f);

        v.add(s1);
        v.add(s2);
        v.add(s3);

    }

}
```

#### 4) Use The Iterator and fetch data from collection In the form of Student

---

```
package org.techhub;

import java.util.*;

public class StudentVectorApplication {
```

```

public static void main(String[] args) {
    // TODO Auto-generated method stub
    Vector v = new Vector();

    Student s1 = new Student();
    s1.setId(1);
    s1.setName("Ram");
    s1.setPer(60.5f);

    Student s2 = new Student();
    s2.setId(2);
    s2.setName("dinesh");
    s2.setPer(50.5f);

    Student s3 = new Student();
    s3.setId(3);
    s3.setName("Rajesh");
    s3.setPer(80.5f);
    v.add(s1);
    v.add(s2);
    v.add(s3);
    Iterator i = v.iterator();

    while(i.hasNext())
    {
        Object obj = i.next();
        Student std=(Student)obj;

    }
}
}

```

## 5) Fetch per from student class using getPer() method

```

package org.techhub;
import java.util.*;
public class StudentVectorApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

```

```

Vector v = new Vector();

Student s1 = new Student();
s1.setId(1);
s1.setName("Ram");
s1.setPer(60.5f);

Student s2 = new Student();
s2.setId(2);
s2.setName("dinesh");
s2.setPer(50.5f);

Student s3 = new Student();
s3.setId(3);
s3.setName("Rajesh");
s3.setPer(80.5f);
v.add(s1);
v.add(s2);
v.add(s3);
Iterator i = v.iterator();

while(i.hasNext())
{
    Object obj = i.next();
    Student std=(Student)obj;
    if(std.getPer()>70)
    {
        System.out.println(std.getId()+"\t"+std.getName()+"\t"+std.getPer());
    }
}
}

```

**WAP program to create the class name as Employee with field name id and salary and store 5 employee data in collection and search employee using its id.**

---

**Steps to implement above assignment**

---

**1) Create the class name as Employee with setter and getter method with parameterized constructor and perform constructor overloading**

```
package org.techhub;

public class Employee {

    private int id;
    private String name;
    public Employee()
    {

    }

    public Employee(String name,int id,int sal)
    {
        this.name=name;
        this.id=id;
        this.sal=sal;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getSal() {
        return sal;
    }

    public void setSal(int sal) {
        this.sal = sal;
    }

    private int sal;
}
```

}

## 2) Create the class with main method and create object of vector class in it.

---

```
package org.techhub;
import java.util.*;
public class EmployeeApplication {

    public static void main(String x[])
    {
        Vector v = new Vector();
        Employee emp1=new Employee("Ram",1,1000);
        Employee emp2 = new Employee("Shyam",2,2000);
        Employee emp3 = new Employee("Ghanshyam",3,30000);

    }
}
```

## 3) Add the employee objects in Vector class

---

```
package org.techhub;
import java.util.*;
public class EmployeeApplication {

    public static void main(String x[])
    {
        Vector v = new Vector();
        Employee emp1=new Employee("Ram",1,1000);
        Employee emp2 = new Employee("Shyam",2,2000);
        Employee emp3 = new Employee("Ghanshyam",3,30000);
        v.add(emp1);
        v.add(emp2);
        v.add(emp3);
    }
}
```

## **5) Fetch data from collection using Iterator**

---

```
package org.techhub;
import java.util.*;
public class EmployeeApplication {

    public static void main(String x[])
    {
        Vector v = new Vector();
        Employee emp1=new Employee("Ram",1,1000);
        Employee emp2 = new Employee("Shyam",2,2000);
        Employee emp3 = new Employee("Ghanshyam",3,30000);
        v.add(emp1);
        v.add(emp2);
        v.add(emp3);
        for(Object obj:v) {
            Employee emp=(Employee)obj;
        }
    }
}
```

## **6) Enter the id for search record in collection and fetch id from employee object and compare and setup the flag**

---

```
package org.techhub;

import java.util.*;

public class EmployeeApplication {

    public static void main(String x[]) {
        Vector v = new Vector();
        Employee emp1 = new Employee("Ram", 1, 1000);
        Employee emp2 = new Employee("Shyam", 2, 2000);
        Employee emp3 = new Employee("Ghanshyam", 3, 30000);
        v.add(emp1);
        v.add(emp2);
        v.add(emp3);
        Scanner xyz = new Scanner(System.in);

        System.out.println("Enter the search id for employee");
        int sid = xyz.nextInt();
```

```

boolean flag = false;
for (Object obj : v) {
    Employee emp = (Employee) obj;
    if (emp.getId() == sid) {
        flag = true;
        break;
    }
}
if (flag) {
    System.out.println("Employee found");
} else {
    System.out.println("Employee Not Found");
}
}
}

```

**WAP Program to create the two classes name as Player and Team  
Player class looks like as**

```

class Player
{ private int id;
  private String name;
  private int runs;
  public Player(String name,int id,int runs)
  {
    this.name=name;
    this.id=id;
    this.runs=runs;
  }
  public void setId(int id)
  { this.id=id;
  }
  public int getId()
  { return id;
  }
  public void setName(String name)
  { this.name=name;
  }
  public String getName()
  
```

```
{ return name;
}
public void setRuns(int runs)
{ this.runs=runs;
}
public int getRuns()
{ return runs;
}
}
```

---

### **Your Team class look like as**

---

```
class Team
{
    void addPlayers (List playList)
    {
        // display here playerlist
    }
}
```

In above example Team class is dependent on player list.

### **Steps to Implement the Above Example**

---

#### **1) Create the player class**

---

```
class Player
{ private int id;
    private String name;
    private int runs;
    public Player(String name,int id,int runs)
    {
        this.name=name;
        this.id=id;
        this.runs=runs;
    }
    public void setId(int id)
    { this.id=id;
    }
    public int getId()
    { return id;
    }
}
```

```
public void setName(String name)
    { this.name=name;
    }
    public String getName()
    { return name;
    }
    public void setRuns(int runs)
    { this.runs=runs;
    }
    public int getRuns()
    {return runs;
    }
}
```

**2)Create the Team class and access the reference of List interface  
And fetch data from collection and display it**

```
package org.techhub;
import java.util.*;
public class Team {

    void addPlayers(List playerList)
    {
        for(Object obj:playerList)
        {
            Player p =(Player)obj;

            System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRuns());
        }
    }
}
```

### 3) Create the main class and create the object player class store all objects in List collection

```
package org.techhub;
import java.util.*;
public class PlayerApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
List list = new Vector();

Player p1= new Player("Ram",1,10000);
Player p2 = new Player("Shyam",2,5000);
Player p3 = new Player("Dinesh",3,30000);
list.add(p1);
list.add(p2);
list.add(p3);

    }
}
```

### 4) Create the object of Team class and pass reference of List collection in it

---

```
package org.techhub;
import java.util.*;
public class PlayerApplication {
    public static void main(String[] args) {
List list = new Vector();
Player p1= new Player("Ram",1,10000);
Player p2 = new Player("Shyam",2,5000);
Player p3 = new Player("Dinesh",3,30000);
list.add(p1);
list.add(p2);
list.add(p3);
Team t = new Team();
t.addPlayers (list);
    }
}
```

**1) WAP to create the class name as College with method name as addNewAdmission (List adminList) in addNewAdmission() method of Collection class contain the List of student data we have the class name as Student wit field id , name , per ,contact and we want to store the all student objects in list collection and pass the list collection reference of addNewAdmission() method which is present in College class.**

**2) WAP to create the class name as Company with following methods**  
**void showEmployeeList(List employeeList):** this method can show the all employee list

**Employee getEmployeeRecordById (int id):** this method can return the specified employee record by using its id

**Employee deleteEmployeeRecordById (int id):** this method can remove the employee record using its id

**Employee getEmployeeById (int id):** this method can search the employee record by using its id.

**Void updateEmployee(int id):** this method update the salary of employee using its id from collection

We have the create the new class name as Employee with three fields id,name and salary and store the all employee objects in List Collection and pass the list collection reference of The company class and perform the above mention operation with List collection in Company class

### **ArrayList Collection**

---

ArrayList is same like as Vector and it is also implementor class of List Collection

But there are some major differences between ArrayList and Vector is

1) Vector occupy double memory when its capacity cross and ArrayList occupy half memory than its current capacity when capacity is cross.

### **Logic of Vector capacity increment**

---

newcapacity = size>oldcapacity?(oldcapacity+oldcapacity):oldcapacity;

## Logic of ArrayList capacity increment ?

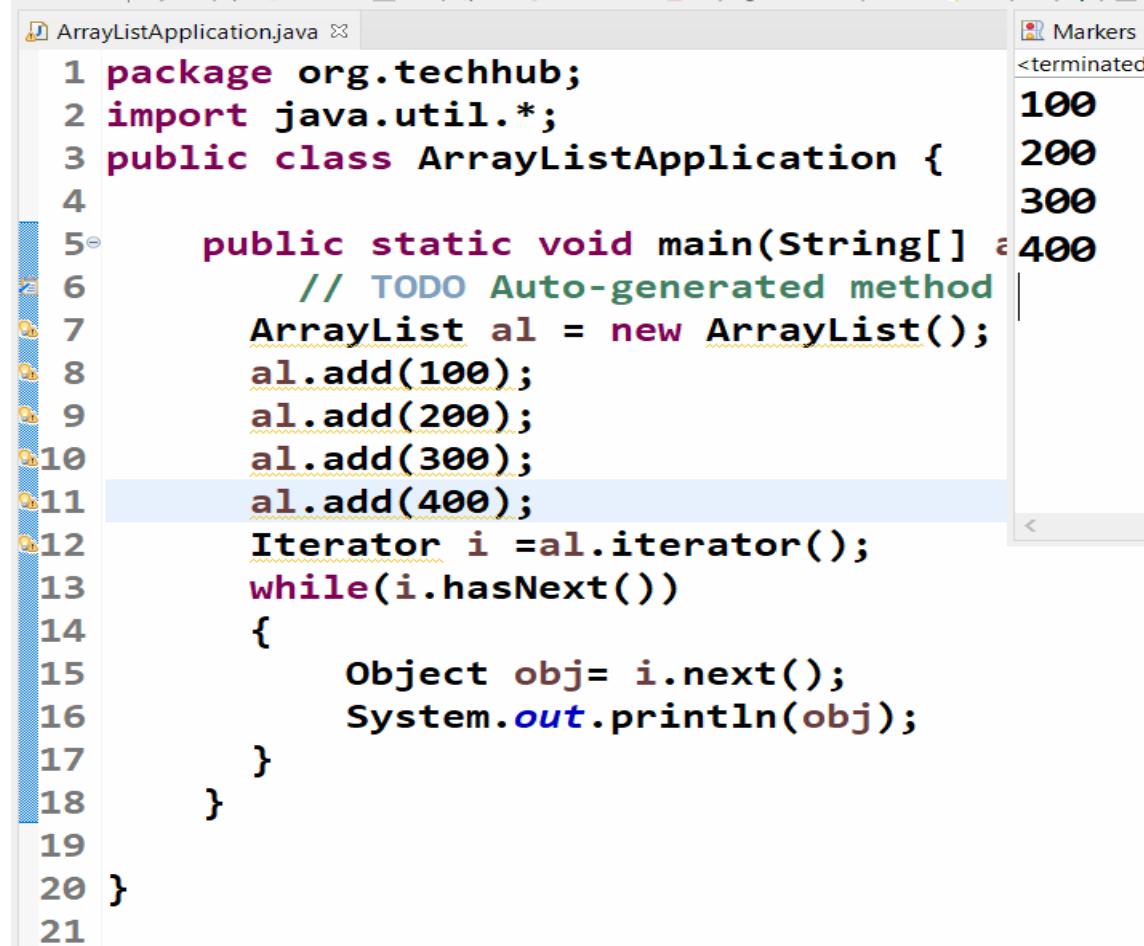
newcapacity = size >oldcapacity? Oldcapacity+oldcapacity >>1: oldcapacity;

- 2) Vector is legacy collection and ArrayList is non legacy collection
- 3) Vector methods are synchronized means Vector is thread safe collection and ArrayList methods are asynchronous means ArrayList is not thread safe

## Similarity between Vector and ArrayList

- 1) Vector default capacity is 10 and ArrayList default capacity is 10
- 2) Vector is part of List implementation and ArrayList also part of list implementation

## Sample code with ArrayList Collection



The screenshot shows a Java code editor with a file named `ArrayListApplication.java`. The code creates an `ArrayList` and adds four elements: 100, 200, 300, and 400. It then iterates over the list using an `Iterator` and prints each element to the console.

```
1 package org.techhub;
2 import java.util.*;
3 public class ArrayListApplication {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         ArrayList al = new ArrayList();
8         al.add(100);
9         al.add(200);
10        al.add(300);
11        al.add(400);
12        Iterator i = al.iterator();
13        while(i.hasNext())
14        {
15            Object obj= i.next();
16            System.out.println(obj);
17        }
18    }
19
20 }
```

On the right side of the editor, there is a 'Markers' panel showing a single entry: '<terminated>' with a value of '100'.

WAP to create the class name as Product with production information like as Product name, quantity, price with setter and getter methods and Create the new class name as Bill with calBill(List list) method and calBill() method contain the reference of List interface or collection and we can store the all Product class object in List Collection and pass list reference to callBill and calculate the bill of products

Output should like as

---

<b>Product Name</b>	<b>Quantity</b>	<b>Rate</b>	<b>Total</b>
Parle-G	5	10	50
Cadbury	10	10	100
Grand Total is			: 150

### **Source Code**

---

```
package org.techhub.billing;

public class Product {

    private String name;
    private int price;
    private int quantity;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getPrice() {
        return price;
    }
}
```

```

public void setPrice(int price) {
    this.price = price;
}

public int getQuantity() {
    return quantity;
}

public void setQuantity(int quantity) {
    this.quantity = quantity;
}

```

## Billing

---

```

package org.techhub.billing;
import java.util.*;
public class Billing {
    int grandTotal=0;
    void calBill(List list)
    {
        System.out.println("Product Name\tQuantity\tRate\tTotal");

        System.out.println("_____");
        for(Object obj:list)
        {
            Product p =(Product)obj;
            int total = p.getQuantity()*p.getPrice();
            grandTotal = grandTotal + total;

            System.out.println(p.getName()+"\t\t\t"+p.getQuantity()+"\t"+p.getPrice()+"\t"+total);
        }
        System.out.println("_____");
        System.out.println("\t\t\t\tTotal Bill :" +grandTotal);
    }
}

```

## BillingApplication

---

```
package org.techhub.billing;

import java.util.*;

public class BillingApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        List al = new ArrayList();
        System.out.println("Enter the Product count");
        Scanner xyz = new Scanner(System.in);
        int size = xyz.nextInt();
        Product p[] = new Product[size];
        for(int i=0; i<p.length;i++)
        {
            p[i]=new Product();
            System.out.println("Enter product name price and quantity");
            xyz = new Scanner(System.in);
            String name=xyz.nextLine();
            int price=xyz.nextInt();
            int quantity=xyz.nextInt();
            p[i].setName(name);
            p[i].setPrice(price);
            p[i].setQuantity(quantity);
            al.add(p[i]);
        }
        Billing b = new Billing();
        b.calBill(al);
    }
}
```

## **Output**

---

```
Enter the Product count
2
Enter product name price and quantity
Parle-G
5
10
Enter product name price and quantity
Cadbury
10
10
Product Name      Quantity      Rate      Total


---


Parle-G           10            5          50
Cadbury          10            10         100


---


Total Bill :150
```

### **1) WAP to create the Application for OnlineExam Using Collection**

Create the pojo class Question with field

question,id,option1,option2,option3,option4,option5,answer

Create the class OnlineExamHelper which contain the following methods

**void addNewQuestion(Question question):** this method is used for add the new question in ArrayList collection

**void removeQuestion(int questionId):** this method is used for remove the question from collection

**void viewAllQuestions():** this method shows the all question from collection

**boolean searchQuestion(String questionname):** this method is used for search question using question name

**void attemptQuestion(int questionid, String answer):** this method can search question using question from collection and check its answer if answer is correct then count the marks otherwise not

**void showResult():** this method show the result means in this method we have to display number of question , count correct question and count incorrect question and display the result in percentage

## Steps to implement Above Assignment

---

- a) create the project in eclipse
- b) create the package name as org.techhub.question
- c) create the pojo Question under the org.techhub.question package.

```
package org.techhub.question;
```

```
public class Question {  
    private int id;  
    private String name;  
    private String option1;  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public String getOption1() {  
        return option1;  
    }
```

```

public void setOption1(String option1) {
    this.option1 = option1;
}
public String getOption2() {
    return option2;
}
public void setOption2(String option2) {
    this.option2 = option2;
}
public String getOption3() {
    return option3;
}
public void setOption3(String option3) {
    this.option3 = option3;
}
public String getOption4() {
    return option4;
}
public void setOption4(String option4) {
    this.option4 = option4;
}
public String getAnswer() {
    return answer;
}
public void setAnswer(String answer) {
    this.answer = answer;
}
private String option2;
private String option3;
private String option4;
private String answer;
}

```

- d) create the one more package name as org.techhub.helper
- e) create the OnlineExamHelper class under the org.techhub.helper package

and create one method name as addNewQuestion(Question question) in OnlineExamHelper class.

```
package org.techhub.helper;
import org.techhub.question.*;
import java.util.*;
public class OnlineExamHelper {

    void addNewQuestion(Question question)
    {

    }
}
```

e) create the object of ArrayList Collection in OnlineExamHelper class and add the question objects in ArrayList object under the addNewQuestion method.

```
package org.techhub.helper;
import org.techhub.question.*;
import java.util.*;
public class OnlineExamHelper {

    List list = new ArrayList();
    void addNewQuestion(Question question)
    {
        list.add(question);
    }
}
```

f) create the new package name as org.techhub.onlineexam

g) create the class OnlineExamClientApp under the org.techhub.onlineexam package and it contain the main method.

```
package org.techhub.onlineexam;

public class OnlineExamClientApp {

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

```
// TODO Auto-generated method stub  
}  
}
```

g) create the object of OnlineExamHelper class and call its method addNewQuestion() but in addNewQuestion() method contain reference of Question so we have to create the object of Question class and store data in it like as question ,id,options and answer and pass to addNewQuestion method

```
package org.techhub.onlineexam;  
  
import java.util.*;  
  
import org.techhub.helper.OnlineExamHelper;  
import org.techhub.question.Question;  
  
public class OnlineExamClientApp {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        OnlineExamHelper helper = new OnlineExamHelper();  
  
        do {  
            Scanner xyz = new Scanner(System.in);  
            System.out.println("Enter your choice");  
            int choice = xyz.nextInt();  
  
            switch (choice) {  
                case 1:  
                    xyz.nextLine();  
                    System.out.println("Enter the question id");  
                    int qid=xyz.nextInt();  
                    xyz.nextLine();  
                    System.out.println("Enter the question");  
                    String question=xyz.nextLine();  
                    System.out.println("Enter the option1 value");  
            }  
        } while (choice != 0);  
    }  
}
```

```

String option1 =xyz.nextLine();
System.out.println("Enter the option2 value");
String option2 = xyz.nextLine();
System.out.println("Enter the option3 value");
String option3 =xyz.nextLine();
System.out.println("Enter the option4 value");
String option4=xyz.nextLine();
System.out.println("Enter the answer");
String ans=xyz.nextLine();
Question q = new Question();
q.setId(qid);
q.setName(question);
q.setOption1(option1);
q.setOption2(option2);
q.setOption1(option3);
q.setOption4(option4);
q.setAnswer(ans);
helper.addNewQuestion(q);
break;
default:
    System.out.println("wrong choice");
}
}while(true);
}
}

```

h) create the new function name as viewAllQuestions() under the OnlineExamHelper class and fetch data from collection created in OnlineExamHelper class

```

package org.techhub.helper;
import org.techhub.question.*;
import java.util.*;
public class OnlineExamHelper {

    List list = new ArrayList();
    public void addNewQuestion(Question question)
}

```

```

{
    list.add(question);
}
public void viewAllQuestions()
{
    for(Object obj:list)
    {
        Question q=(Question)obj;

        System.out.println(q.getId()+"\t"+q.getName()+"\t"+q.getOption1()+"\t"+q.getOption2()+"\t"+q.getOption3()+"\t"+q.getOption4()+"\t"+q.getAnswer());
    }
}
}

```

i) call the ViewAllStudents() function from main on choice number 3

```

package org.techhub.onlineexam;
import java.util.*;
import org.techhub.helper.OnlineExamHelper;
import org.techhub.question.Question;

public class OnlineExamClientApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        OnlineExamHelper helper = new OnlineExamHelper();

        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("Enter your choice");
            int choice = xyz.nextInt();

            switch (choice) {
                case 1:
                    xyz.nextLine();
                    System.out.println("Enter the question id");
                    int qid=xyz.nextInt();
                    xyz.nextLine();
            }
        }
    }
}

```

```

System.out.println("Enter the question");
String question=xyz.nextLine();
System.out.println("Enter the option1 value");
String option1 =xyz.nextLine();
System.out.println("Enter the option2 value");
String option2 = xyz.nextLine();
System.out.println("Enter the option3 value");
String option3 =xyz.nextLine();
System.out.println("Enter the option4 value");
String option4=xyz.nextLine();
System.out.println("Enter the answer");
String ans=xyz.nextLine();
Question q = new Question();
q.setId(qid);
q.setName(question);
q.setOption1(option1);
q.setOption2(option2);
q.setOption1(option3);
q.setOption4(option4);
q.setAnswer(ans);
helper.addNewQuestion(q);
break;
case 3:
    helper.viewAllQuestions();
    break;
default:
    System.out.println("wrong choice");
}
}while(true);
}
}

```

j) create new method under the class OnlineExamHelper name as void removeQuestion(int questionId) and remove the question using question id  
**Note:** first we have to travel the whole collection and extract the single single object of question from collection and extract the question id from Question object and compare the your input question id with object question object id

and fine the index of Question object if question id match then remove the question object by using index of that object.

### Example

---

```
package org.techhub.helper;
import org.techhub.question.*;
import java.util.*;
public class OnlineExamHelper {

    List list = new ArrayList();
    public void addNewQuestion(Question question)
    {
        list.add(question);
    }
    public void viewAllQuestions()
    {
        for(Object obj:list)
        {
            Question q=(Question)obj;

System.out.println(q.getId()+"\t"+q.getName()+"\t"+q.getOption1()+"\t"+q.getOption2()+"\t"+q.getOption3()+"\t"+q.getOption4()+"\t"+q.getAnswer());
        }
    }
    public void removeQuestion(int questionId)
    {
        for(Object obj:list)
        {
            Question q=(Question)obj;
            int qid=q.getId();
            if(qid==questionId)
            {
                int index=list.indexOf(q);
                if(index!=-1)
                { list.remove(index);
                }
            }
        }
    }
}
```

```
    }  
}
```

k) in main method class input the question id under the choice 2 and call the removeQuestion() method of OnlineExamHelper class object and pass the question id in removeQuestion() method

```
package org.techhub.onlineexam;  
  
import java.util.*;  
  
import org.techhub.helper.OnlineExamHelper;  
import org.techhub.question.Question;  
  
public class OnlineExamClientApp {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        OnlineExamHelper helper = new OnlineExamHelper();  
  
        do {  
            Scanner xyz = new Scanner(System.in);  
            System.out.println("Enter your choice");  
            int choice = xyz.nextInt();  
  
            switch (choice) {  
                case 1:  
                    xyz.nextLine();  
                    System.out.println("Enter the question id");  
                    int qid=xyz.nextInt();  
                    xyz.nextLine();  
                    System.out.println("Enter the question");  
                    String question=xyz.nextLine();  
                    System.out.println("Enter the option1 value");  
                    String option1 =xyz.nextLine();  
                    System.out.println("Enter the option2 value");  
                    String option2 = xyz.nextLine();  
                    System.out.println("Enter the option3 value");  
                    String option3 =xyz.nextLine();  
            }  
        } while (choice != 3);  
    }  
}
```

```

System.out.println("Enter the option4 value");
String option4=xyz.nextLine();
System.out.println("Enter the answer");
String ans=xyz.nextLine();
Question q = new Question();
q.setId(qid);
q.setName(question);
q.setOption1(option1);
q.setOption2(option2);
q.setOption1(option3);
q.setOption4(option4);
q.setAnswer(ans);
helper.addNewQuestion(q);
break;
case 2:
System.out.println("Enter the question id which want to remove");
int questionId=xyz.nextInt();
helper.removeQuestion(questionId);
break;
case 3:
    helper.viewAllQuestions();
break;
default:
    System.out.println("wrong choice");
}
}while(true);
}

```

- I) create the searchQuestion(String questionName) under the OnlineExamHelper class and iterate the List of question and fetch single signle object of Question class from collection and extract the question name using getName() method of Question class and compare the question with user input value if found return true otherwise return false

## Example

---

```
package org.techhub.helper;

import org.techhub.question.*;
import java.util.*;

public class OnlineExamHelper {

    List list = new ArrayList();

    public void addNewQuestion(Question question) {
        list.add(question);
    }

    public void viewAllQuestions() {
        for (Object obj : list) {
            Question q = (Question) obj;
            System.out.println(q.getId() + "\t" + q.getName() + "\t" +
q.getOption1() + "\t" + q.getOption2() + "\t" +
                + q.getOption3() + "\t" + q.getOption4() + "\t" +
q.getAnswer());
        }
    }

    public void removeQuestion(int questionId) {
        for (Object obj : list) {
            Question q = (Question) obj;
            int qid = q.getId();
            if (qid == questionId) {
                int index = list.indexOf(q);
                if (index != -1) {
                    list.remove(index);
                }
            }
        }
    }

    public boolean searchQuestion(String questionName) {
        Iterator i = list.iterator();
```

```

boolean b = false;
while (i.hasNext()) {
    Object obj = i.next();
    Question q = (Question) obj;
    String question = q.getName();
    if (question.equals(questionName)) {
        b = true;
        break;
    }
}
return b;
}
}

```

m) call the searchQuestion() on case number 4 I main method and pass question from user input to searchQuestion() and catch the result return by searchQuestion() method and compare using if and if return true by searchQuestion() then display message quesiton found otherwise display message question not found.

```

package org.techhub.onlineexam;

import java.util.*;
import org.techhub.helper.OnlineExamHelper;
import org.techhub.question.Question;
public class OnlineExamClientApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        OnlineExamHelper helper = new OnlineExamHelper();

        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("Enter your choice");
            int choice = xyz.nextInt();

            switch (choice) {
                case 1:
                    xyz.nextLine();

```

```
System.out.println("Enter the question id");
int qid=xyz.nextInt();
xyz.nextLine();
System.out.println("Enter the question");
String question=xyz.nextLine();
System.out.println("Enter the option1 value");
String option1 =xyz.nextLine();
System.out.println("Enter the option2 value");
String option2 = xyz.nextLine();
System.out.println("Enter the option3 value");
String option3 =xyz.nextLine();
System.out.println("Enter the option4 value");
String option4=xyz.nextLine();
System.out.println("Enter the answer");
String ans=xyz.nextLine();
Question q = new Question();
q.setId(qid);
q.setName(question);
q.setOption1(option1);
q.setOption2(option2);
q.setOption1(option3);
q.setOption4(option4);
q.setAnswer(ans);
helper.addNewQuestion(q);
break;

case 2:
System.out.println("Enter the question id which want to remove");
int questionId=xyz.nextInt();
helper.removeQuestion(questionId);
break;

case 3:
helper.viewAllQuestions();
break;

case 4:
System.out.println("Enter the question for searching purpose");
xyz.nextLine();
question=xyz.nextLine();
boolean b=helper.searchQuestion(question);
if(b)
```

```

        {
            System.out.println("Question Found");
        }
    else
    {
        System.out.println("Question not found");
    }
    break;
default:
    System.out.println("wrong choice");
}
}while(true);
}
}

```

N) attempt question logic : we have to create method name as attemptQuestion(int questionid,String ans) in OnlineExamHelper class and we have the fetch single single Question object from collection class and fetch question id and question answer and compare with given input by user and if found then increment couter and initialize the counter by default with 0 value Example

---

```

package org.techhub.helper;

import org.techhub.question.*;
import java.util.*;

public class OnlineExamHelper {

    List list = new ArrayList();
    int count=0;
    public void addNewQuestion(Question question) {
        list.add(question);
    }

    public void viewAllQuestions() {
        for (Object obj : list) {
            Question q = (Question) obj;
        }
    }
}

```

```

        System.out.println(q.getId() + "\t" + q.getName() + "\t" +
q.getOption1() + "\t" + q.getOption2() + "\t"
                    + q.getOption3() + "\t" + q.getOption4() + "\t" +
q.getAnswer());
    }
}

public void removeQuestion(int questionId) {
    for (Object obj : list) {
        Question q = (Question) obj;
        int qid = q.getId();
        if (qid == questionId) {
            int index = list.indexOf(q);
            if (index != -1) {
                list.remove(index);
            }
        }
    }
}

public boolean searchQuestion(String questionName) {
    Iterator i = list.iterator();
    boolean b = false;
    while (i.hasNext()) {
        Object obj = i.next();
        Question q = (Question) obj;
        String question = q.getName();
        if (question.equals(questionName)) {
            b = true;
            break;
        }
    }
    return b;
}

public void attemptQuestion(int questionId, String answer)
{
    for(Object obj:list)
    {
        Question q=(Question)obj;

```

```

        int qid=q.getId();
        String ans=q.getAnswer();
        if(qid==questionId && ans.equals(answer))
        {
            ++count;
        }
    }
}

```

O) call the attemptQuestion() function on choice number 5 in main method and we provide the question id and answer as input to attemptQuestion function

Example

---

```

package org.techhub.onlineexam;

import java.util.*;

import org.techhub.helper.OnlineExamHelper;
import org.techhub.question.Question;

public class OnlineExamClientApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        OnlineExamHelper helper = new OnlineExamHelper();

        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("Enter your choice");
            int choice = xyz.nextInt();

            switch (choice) {
                case 1:
                    xyz.nextLine();
                    System.out.println("Enter the question id");
                    int qid=xyz.nextInt();

```

```

xyz.nextLine();
System.out.println("Enter the question");
String question=xyz.nextLine();
System.out.println("Enter the option1 value");
String option1 =xyz.nextLine();
System.out.println("Enter the option2 value");
String option2 = xyz.nextLine();
System.out.println("Enter the option3 value");
String option3 =xyz.nextLine();
System.out.println("Enter the option4 value");
String option4=xyz.nextLine();
System.out.println("Enter the answer");
String ans=xyz.nextLine();
Question q = new Question();
q.setId(qid);
q.setName(question);
q.setOption1(option1);
q.setOption2(option2);
q.setOption1(option3);
q.setOption4(option4);
q.setAnswer(ans);
helper.addNewQuestion(q);
break;
case 2:
    System.out.println("Enter the question id which want to
remove");
    int questionId=xyz.nextInt();
    helper.removeQuestion(questionId);
    break;
case 3:
    helper.viewAllQuestions();
    break;
case 4:
    System.out.println("Enter the question for searching
purpose");
    xyz.nextLine();
    question=xyz.nextLine();
    boolean b=helper.searchQuestion(question);
    if(b)

```

```

    {
        System.out.println("Question Found");
    }
    else
    {
        System.out.println("Question not found");
    }
    break;
case 5:
    xyz.nextLine();
    System.out.println("Enter the answer");
    ans=xyz.nextLine();
    System.out.println("Enter the question number");
    qid=xyz.nextInt();
    helper.attemptQuestion(qid, ans);
default:
    System.out.println("wrong choice");
}
}while(true);
}
}

```

P) create the showResult() method in OnlineExamHelper class and in this method we have the calculate the size of collection means we have to see the total number of question and we have to perform subtraction operation means we have to subtract correct question count from size of collection means we get incorrect question automatically and we have to calculate its percentage .

## **Example**

---

```

package org.techhub.helper;

import org.techhub.question.*;
import java.util.*;

public class OnlineExamHelper {
    List list = new ArrayList();
}

```

```
int count=0;
public void addNewQuestion(Question question) {
    list.add(question);
}

public void viewAllQuestions() {
    for (Object obj : list) {
        Question q = (Question) obj;
        System.out.println(q.getId() + "\t" + q.getName() + "\t" +
q.getOption1() + "\t" + q.getOption2() + "\t" +
                + q.getOption3() + "\t" + q.getOption4() + "\t" +
q.getAnswer());
    }
}

public void removeQuestion(int questionId) {
    for (Object obj : list) {
        Question q = (Question) obj;
        int qid = q.getId();
        if (qid == questionId) {
            int index = list.indexOf(q);
            if (index != -1) {
                list.remove(index);
            }
        }
    }
}

public boolean searchQuestion(String questionName) {
    Iterator i = list.iterator();
    boolean b = false;
    while (i.hasNext()) {
        Object obj = i.next();
        Question q = (Question) obj;
        String question = q.getName();
        if (question.equals(questionName)) {
            b = true;
            break;
        }
    }
}
```

```

        }
        return b;
    }
    public void attemptQuestion(int questionId, String answer)
    {
        for(Object obj:list)
        {
            Question q=(Question)obj;
            int qid=q.getId();
            String ans=q.getAnswer();
            if(qid==questionId && ans.equals(answer))
            {
                ++count;
            }
        }
    }
    public void showResult()
    {
        float totalQuestion = (float)list.size();
        float incorrectQuestion=totalQuestion-count;
        float per=count/totalQuestion;
        System.out.println("Percentage acheive by student "+(per*100));
    }
}

```

Q) create the case number six in main method and call the showresult() function of OnlineExamHelper class on case number 6

---

### Example

---

```

package org.techhub.onlineexam;

import java.util.*;

import org.techhub.helper.OnlineExamHelper;
import org.techhub.question.Question;

public class OnlineExamClientApp {

    public static void main(String[] args) {

```

```
// TODO Auto-generated method stub
OnlineExamHelper helper = new OnlineExamHelper();

do {
    Scanner xyz = new Scanner(System.in);
    System.out.println("Enter your choice");
    int choice = xyz.nextInt();

    switch (choice) {
        case 1:
            xyz.nextLine();
            System.out.println("Enter the question id");
            int qid=xyz.nextInt();
            xyz.nextLine();
            System.out.println("Enter the question");
            String question=xyz.nextLine();
            System.out.println("Enter the option1 value");
            String option1 =xyz.nextLine();
            System.out.println("Enter the option2 value");
            String option2 = xyz.nextLine();
            System.out.println("Enter the option3 value");
            String option3 =xyz.nextLine();
            System.out.println("Enter the option4 value");
            String option4=xyz.nextLine();
            System.out.println("Enter the answer");
            String ans=xyz.nextLine();
            Question q = new Question();
            q.setId(qid);
            q.setName(question);
            q.setOption1(option1);
            q.setOption2(option2);
            q.setOption3(option3);
            q.setOption4(option4);
            q.setAnswer(ans);
            helper.addNewQuestion(q);
        break;
        case 2:
            System.out.println("Enter the question id which want to remove");
            int questionId=xyz.nextInt();
```

```

        helper.removeQuestion(questionId);
        break;
case 3:
    helper.viewAllQuestions();
    break;
case 4:
    System.out.println("Enter the question for searching
purpose");
    xyz.nextLine();
    question=xyz.nextLine();
    boolean b=helper.searchQuestion(question);
    if(b)
    {
        System.out.println("Question Found");
    }
    else
    {
        System.out.println("Question not found");
    }
    break;
case 5:
    xyz.nextLine();
    System.out.println("Enter the answer");
    ans=xyz.nextLine();
    System.out.println("Enter the question number");
    qid=xyz.nextInt();
    helper.attemptQuestion(qid, ans);
    break;
case 6:
    helper.showResult();
default:
    System.out.println("wrong choice");
}
}while(true);
}

```

## 2) WAP to create the Program For Training And Placement Activity

Create pojo class name as Student with field name , id,contact,email, gradutionPer and one more class TrainingAndPlacementFilter with following methods .

**void addNewStudent(Student student):** this method accept the student information and store in collection or ArrayList

**void debarStudent(String email):** this accept the email of student and remove the record from collection if email found if email not show the message student is not found.

**void searchStudent(String email ,String contact):** this method search record of student using its email and its contact if found then display the record of student if not found then show the message student is not found.

**void arrangeStudent():** this method can show the all students in descending order using their graduationPercentage

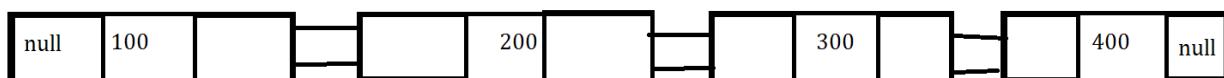
## **LinkedList**

---

LinkedList is by default doubly Linked list in java collection. Double linked list means which having two pointers to every node previous pointer and next pointers. The Previous pointer points to previous node and next pointer points to next node. LinkedList is data structure where the elements are not stored in continuous location and every element is separate object with data part and address part .the elements are linked using pointers and address. Each element is known as node

**Following Diagram shows the double linkedlist**

---



## **Constructor of LinkedList in java**

---

**There are two constructor in LinkedList**

---

**1) LinkedList():** this is used for create the default linked list object with no any size.

**2) LinkedList(Collection):** this constructor is used for create the linked by copying data from another collection .

## **Performing various operation on LinkedList**

---

**1. Adding elements :** In order to add() an element to ArrayList we can use the add()method . This method is overloaded to perform multiple operation based on different parameters

**void add(Object):** this method is used to add an element at the end of the LinkedList

**void add(int index, Object):** this method is used to add an element at specific index in the LinkedList.

## **Changing element or replace the element in LinkedList**

---

After adding the element if we wish to change the element ,it can be done using set() method .Since LinkedList is indexed the element which we wish to change is referenced by the index of an element therefore this method takes and index and update the which need to inserted at the index.

**void set(int index, Object value):** this method can update or change the element on specified index in LinkedList.

## **Removing elements**

---

In order to remove an element from a LinkedList,we can use the remove() method. This method is also overloaded to perform multiple operations based on different parameter.

**void remove(Object):** this method is used to simply remove an object from the LinkedList. If there are multiple such objects,then the first occurrence of the object is removed.

**void remove(int index):** since a LinkedList is indexed ,this method takes an integer value which simply removes the element present at the specific index in the LinkedList. After removing the element all the elements are moved to the left to fill the space and the indices of the objects are updated.

## Program for LinkedList using Default constructor

---

### Example

---

```
package org.techhub;
import java.util.*;
public class LinkedListApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LinkedList lst = new LinkedList();
        lst.add(100);
        lst.add(200);
        lst.add(300);
        lst.add(400);
        Iterator i=lst.iterator();
        while(i.hasNext())
        {
            Object obj = i.next();
            System.out.println(obj);
        }
    }
}
```

## **Program for LinkedList using second constructor**

---

In second constructor we have to copy the content of one collection in to the another collection

```
package org.techhub;
import java.util.*;
public class LinkedListApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList al = new ArrayList();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        LinkedList lst = new LinkedList(al);
        for(Object obj:lst)
        {
            System.out.println(obj);
        }
    }
}
```

## **Q.what is the diff between ArrayList and LinkedList**

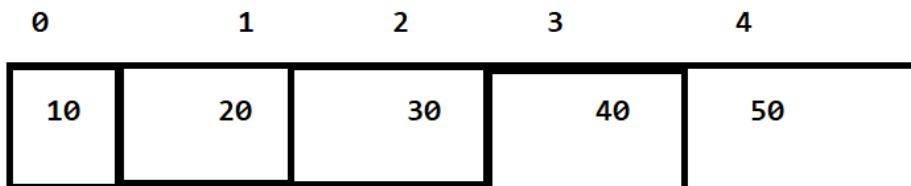
---

- 1) ArrayList is dynamic array means it store data in continuous allocation and LinkedList is by default doubly LinkedList.
- 2) ArrayList use the O(n) and LinkedList use the O(1)  
O(bigo notation) is used for calculate the time complexity of application means the time of complexity of ArrayList is O(n) and the time of complexity of LinkedList is O(1) means if we remove the element from array list or insert the element in ArrayList then it will number of iteration of shifting element means it will take more time for these two operations but if we use the LinkedList then remove and insert the element it will never perform iteration so LinkedList take less time for inserting and removing operation

Means in short we can say LinkedList is faster than ArrayList as for deletion and insertion of element as compare with ArrayList.

### Following Example Demonstrate the above concept

Suppose consider we have the ArrayList given below.

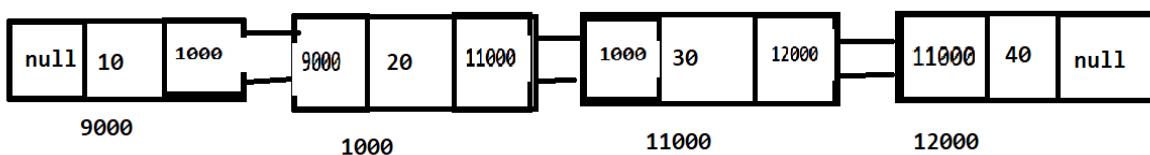


ArrayList

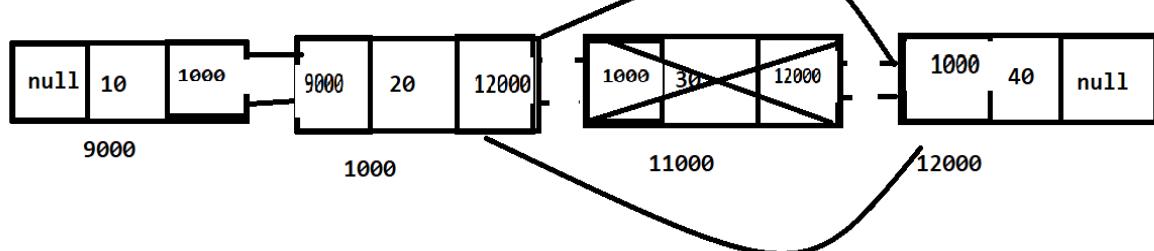
suppose consider we want to delete the element from ArrayList of index 1 then we have to shift the element 2 index on 1 index 3 index on 2 index and fourth index and 3 index and remove the location of last element so it will take time means as per our example we have the perform 3 iteration

suppose consider we have the 1 lakh element in ArrayList and we want to remove the 1 index then there is n number of iteration and it will take more time so we can say time complexity ArrayList is  $O(n)$  so ArrayList is not recommended for deletion and updation purpose.

Suppose consider we have the linkedlist given below.



suppose consider we want to remove the element 30 from LinkedList in this we shift the next address of 20 element node to 12000 and 12000 previous element to 1000



here we not need to perform iteration using linkedlist so the time complexity of LinkedList is  $O(1)$

3) ArrayList is faster than linked for data fetching purpose because ArrayList store in memory in continuous manner and LinkedList is slower than ArrayList for data fetching but LinkedList store randomly memory so in LinkedList for data fetching first we have to search the address of the node and then value it will take more

## **Example**

---

**WAP for review management system here we have to perform following task.**

Case 1: Add New Teacher

Case 2: Rate To Teacher

Case 3: View Number of Rating to Teacher

Case 4: View Positive Rating

Case 5: view Negative Rating

Caes 6: View Average Rating.

Create the pojo class name as Teacher with fields tid,name,rating , count , textreview and create the one more class name as RatingCounter with method name as addTeacher(Teacher) in this method contain the Teacher reference and RatingCounter container the following methods for perform different operation

**boolean isAddTeacher(Teacher teacher):** this method is used for add the new teacher in Collection

**void rateToTeacher(int rating, String teacherName, int teacherId, String textReview):** this method is used for rating to particular Teacher

**void viewTeacherRating():** this method show the rating of every teacher.

**void viewPositiveRating():** this method shows the positive to teacher

**void viewNeativeRating():** this method return the negative rating

**void showAverateRating():** this method show the average rating to every teacher.

## **Steps to implementation of above project**

---

**1) create the java project using eclipse or any other tool.**

**2) create the new package name as org.techhub.teacher and under the package create the new class name as Teacher**

### **Example**

---

```
package org.techhub.teacher;
import java.util.*;
public class Teacher {
    private String name;
    private int id;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public ArrayList getRating() {
        return rating;
    }

    public void setRating(ArrayList rating) {
        this.rating = rating;
    }

    public int getCount() {
        return count;
    }
```

```

public void setCount(int count) {
    this.count = count;
}

public ArrayList getTextReview() {
    return textReview;
}

public void setTextReview(ArrayList textReview) {
    this.textReview = textReview;
}

private ArrayList rating;
private int count;
private ArrayList textReview;
}

```

**3) create the new package name as org.techhub.rating and create the new class under this package name as RatingCounter**

```
package org.techhub.rating;
```

```

public class RatingCounter {

}

```

**4) add the new method addTeacher(Teacher teacher) under the RatingCounter class.**

```

package org.techhub.rating;
import java.util.*;
import org.techhub.teacher.Teacher;
public class RatingCounter {

    LinkedList teacherList = new LinkedList();
    public boolean isAddTeacher(Teacher teacher)
    {
        boolean b=teacherList.add(teacher);
        return b;
    }
}

```

}

**5) create the new package org.techhub.clientapp and under this package we have to create class name as RatingClientApplication with main method**

```
package org.techhub.clientapp;
```

```
import java.util.*;
```

```
import org.techhub.rating.RatingCounter;
```

```
import org.techhub.teacher.Teacher;
```

```
public class RatingClientApplication {
```

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

```
        // TODO Auto-generated method stub
```

```
        RatingCounter r = new RatingCounter();
```

```
        do {
```

```
            Scanner xyz = new Scanner(System.in);
```

```
            System.out.println("1:Add New Teacher");
```

```
            System.out.println("2:Rate To Teacher");
```

```
            System.out.println("3:View Teacher Rating");
```

```
            System.out.println("4:View Positive Rating");
```

```
            System.out.println("5:View Negative Rating");
```

```
            System.out.println("6:View Average Rating");
```

```
            System.out.println("Enter Your Choice");
```

```
            int choice = xyz.nextInt();
```

```
            switch (choice) {
```

```
                case 1:
```

```
                    xyz.nextLine();
```

```
                    System.out.println("enter the name of Teacher");
```

```
                    String name=xyz.nextLine();
```

```
                    System.out.println("Enter the id of Teacher");
```

```
                    int id=xyz.nextInt();
```

```
                    Teacher t = new Teacher();
```

```
                    t.setId(id);
```

```
                    t.setName(name);
```

```
                    boolean b = r.isAddTeacher(t);
```

```
                    if(b)//if(true)
```

```
                {
```

```

        System.out.println("Teacher Added Successfully.....");
    }
    else
    {
        System.out.println("Teacher Not Added .....");
    }
    break;
case 2:
    break;
case 3:
    break;
case 4:
    break;
case 5:
    break;
case 6:
    break;
case 7:
    System.exit(0);
    break;
default:
    System.out.println("wrong choice");
}
} while (true);

}

```

**6) Rating to Teacher** we have the function name as **rateToTeacher()** with parameters rating and teacher name but we have the **LinkedList** name as **teacherList** which contain the n number of teacher so first we have to search teacher name and teacher id for rating if teacher found then we can give rating to teacher otherwise we have to give message teacher not found

### Source Code

---

```

package org.techhub.rating;
import java.util.*;

```

```
import org.techhub.teacher.Teacher;
public class RatingCounter {

    LinkedList teacherList = new LinkedList();

    public boolean isAddTeacher(Teacher teacher)
    {
        boolean b=teacherList.add(teacher);
        return b;
    }
    public void rateToTeacher(int rating,String name,int id,String
textReview)
    { boolean flag=false;
        Teacher t=null;
        for(Object obj:teacherList)
        {
            t=(Teacher)obj;
            String n=t.getName();
            int i=t.getId();
            if(n.equals(name) && i==id)
            {
                flag=true;
                break;
            }
        }
        if(flag)
        {
            ArrayList ratingArrayList=t.getRating();
            if(ratingArrayList==null)
            {
                ratingArrayList = new ArrayList();
            }
            ratingArrayList.add(rating);
            ArrayList textReviewList=t.getTextReview();
            if(textReviewList==null)
            {
                textReviewList=new ArrayList();
            }
        }
    }
}
```

```

        textReviewList.add(textReview);
        t.setRating(ratingArrayList);
        t.setTextReview(textReviewList);
    }
} else {
{
    System.out.println("Teacher Not Found for Rating");
}
}

public void viewAllRating()
{
    for(Object obj:teacherList)
    {
        Teacher t=(Teacher)obj;
        System.out.println(t.getId()+"\t"+t.getName()+"\t"+t.getRating()+"\t"+
t.getTextReview());
    }
}
}

```

## Main method class

---

```

package org.techhub.clientapp;

import java.util.*;
import org.techhub.rating.RatingCounter;
import org.techhub.teacher.Teacher;

public class RatingClientApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        RatingCounter r = new RatingCounter();
        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("1:Add New Teacher");
            System.out.println("2:Rate To Teacher");
            System.out.println("3:View Teacher Rating");

```

```
System.out.println("4:View Positive Rating");
System.out.println("5:View Negative Rating");
System.out.println("6:View Average Rating");
System.out.println("Enter Your Choice");
int choice = xyz.nextInt();
switch (choice) {
case 1:
    xyz.nextLine();
    System.out.println("enter the name of Teacher");
    String name=xyz.nextLine();
    System.out.println("Enter the id of Teacher");
    int id=xyz.nextInt();
    Teacher t = new Teacher();
    t.setId(id);
    t.setName(name);
    boolean b = r.isAddTeacher(t);
    if(b)//if(true)
    {
        System.out.println("Teacher Added Successfully.....");
    }
    else
    {
        System.out.println("Teacher Not Added .....");
    }
    break;
case 2:
    xyz.nextLine();
    System.out.println("Enter the Teacher name");
    name=xyz.nextLine();
    System.out.println("enter the text review for teacher");
    String textReview=xyz.nextLine();
    System.out.println("Enter the id of teacher");
    id =xyz.nextInt();
    System.out.println("Enter the rating to teacher");
    int rating=xyz.nextInt();
    r.rateToTeacher(rating, name, id, textReview);
    break;
case 3:
    r.viewAllRating();
```

```

        break;
    case 4:
        break;
    case 5:
        break;
    case 6:
        break;
    case 7:
        System.exit(0);
        break;
    default:
        System.out.println("wrong choice");
    }
} while (true);

}

```

**7) view positive review :** here we have to create the `viewReviewRating()` function under the `RatingCounter` class and write the following logics

```

package org.techhub.rating;
import java.util.*;

```

```

import org.techhub.teacher.Teacher;
public class RatingCounter {

```

```

LinkedList teacherList = new LinkedList();

public boolean isAddTeacher(Teacher teacher)
{
    boolean b=teacherList.add(teacher);
    return b;
}
public void rateToTeacher(int rating,String name,int id,String
textReview)
{ boolean flag=false;
    Teacher t=null;
    for(Object obj:teacherList)
    {

```

```
t=(Teacher)obj;
String n=t.getName();
int i=t.getId();
if(n.equals(name) && i==id)
{
    flag=true;
    break;
}

}
if(flag)
{
    ArrayList ratingArrayList=t.getRating();
    if(ratingArrayList==null)
    {
        ratingArrayList = new ArrayList();
    }
    ratingArrayList.add(rating);
    ArrayList textReviewList=t.getTextReview();
    if(textReviewList==null)
    {
        textReviewList=new ArrayList();
    }
    textReviewList.add(textReview);
    t.setRating(ratingArrayList);
    t.setTextReview(textReviewList);
}
else
{
    System.out.println("Teacher Not Found for Rating");
}
}

public void viewAllRating()
{
    for(Object obj:teacherList)
    {
        Teacher t=(Teacher)obj;
```

```

        System.out.println(t.getId()+"\t"+t.getName()+"\t"+t.getRating()+"\t"+
t.getTextReview());
    }
}
public void viewPositiveReview()
{
    for(Object obj:teacherList)
    {
        Teacher t=(Teacher)obj;
        ArrayList reviewList=t.getTextReview();
        Iterator i=reviewList.iterator();
        System.out.println("View Positive Reviews");

        while(i.hasNext())
        {
            String review=(String)i.next();
            if(review.equals("good") ||review.equals("excellent")
||review.equals("better"))
            {
                System.out.println(review);
            }
        }
    }
}

```

## **Main Method**

---

```

package org.techhub.clientapp;
import java.util.*;
import org.techhub.rating.RatingCounter;
import org.techhub.teacher.Teacher;

public class RatingClientApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}

```

```
RatingCounter r = new RatingCounter();
do {
    Scanner xyz = new Scanner(System.in);
    System.out.println("1:Add New Teacher");
    System.out.println("2:Rate To Teacher");
    System.out.println("3:View Teacher Rating");
    System.out.println("4:View Positive Rating");
    System.out.println("5:View Negative Rating");
    System.out.println("6:View Average Rating");
    System.out.println("Enter Your Choice");
    int choice = xyz.nextInt();
    switch (choice) {
        case 1:
            xyz.nextLine();
            System.out.println("enter the name of Teacher");
            String name=xyz.nextLine();
            System.out.println("Enter the id of Teacher");
            int id=xyz.nextInt();
            Teacher t = new Teacher();
            t.setId(id);
            t.setName(name);
            boolean b = r.isAddTeacher(t);
            if(b)//if(true)
            {
                System.out.println("Teacher Added Successfully.....");
            }
            else
            {
                System.out.println("Teacher Not Added .....");
            }
            break;
        case 2:
            xyz.nextLine();
            System.out.println("Enter the Teacher name");
            name=xyz.nextLine();
            System.out.println("enter the text review for teacher");
            String textReview=xyz.nextLine();
            System.out.println("Enter the id of teacher");
            id =xyz.nextInt();
```

```

        System.out.println("Enter the rating to teacher");
        int rating=xyz.nextInt();
        r.rateToTeacher(rating, name, id, textReview);
        break;
    case 3:
        r.viewAllRating();
        break;
    case 4:
        r.viewPositiveReview();
        break;
    case 5:
        break;
    case 6:
        break;
    case 7:
        System.exit(0);
        break;
    default:
        System.out.println("wrong choice");
    }
} while (true);
}
}

```

## **Stack**

---

Stack is also implementer class of List Collection. This class work on principal of last in first out format and it is also child class of Vector class

```

List
|
Vector
|
Stack

```

Stack class having only one constructor

**Stack()**: create the object of stack class using default constructor

## **Operation with stack**

---

**void push (E/Object):** this method can push the element in stack the initially top position is -1 when we push the element then top increment by 1

**Object pop():** this method can remove the last element from stack collection means this method can remove the top most element from stack collection

**Objet peek():** this method only show the last or top most element from stack not remove.

**boolean isEmpty():** this method check wheather statck is empty or not if empty return true otherwise return false.

### **WAP to perform basic operation on stack given below**

- 1. Push the element in Stack**
- 2. Pop or Remove element From Stack**
- 3. Display All Element from stack in Last in first out order**

### **Source code given below**

---

```
package org.techhub;
import java.util.*;
public class StackApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Stack s = new Stack();
        int choice;
        do {
            Scanner xyz = new Scanner(System.in);
            System.out.println("Enter your choice");
            choice = xyz.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Enter the element in stack");
                    int ele = xyz.nextInt();
                    s.push(ele);
                    break;
                case 2:
                    boolean b = s.isEmpty();
                    if (b) {
                        System.out.println("Stack is empty");
                    } else {
```

```
        System.out.println("Removed element is " + s.pop());
    }
    break;
case 3:
    ListIterator li = s.listIterator(s.size());
    while (li.hasPrevious()) {
        Object obj = li.previous();
        System.out.println(obj);
    }
    break;
default:
    System.out.println("wrong choice");
}
} while (true);
}
}
```

## Output

---

```
Enter your choice
1
Enter the element in stack
10
Enter your choice
1
Enter the element in stack
20
Enter your choice
1
Enter the element in stack
30
Enter your choice
1
Enter the element in stack
40
Enter your choice
3
40
30
20
10
```

## **Mini Project using Stack**

---

WAP to create the passing booking Application

In this application we have the classname as Passenger with field id ,name,contact,source and destination and one more class name as

TravelAgency which is depend on Passenger

The Policy the of TravelAgency is first passing allocate last seat which is number 1 then second passenger allocat next second from back side of bus seat number 2 and so on when bus reach destination then last passenger remove first .

Here we have to create the POJO class name as Passenger and we have to create the one class name as TravelAgency with following methods

**TravelAgency():** this constructor initialize the seat number by default 0  
And when new passenger added in bus then it will increment by 1 and so on

**void addNewPassenger(Passenger passenger):** this method accept the detail of passenger and add in stack

**void removePassenger():** this method remove the last passenger added in bus by default .

**int getPassengerCount():** this method return exact of number of passenger in bus.

**ListIterator getAllPassengerDetail():** this method return detail of all passenger available in bus.

**Steps to Implement above project(refer video from topic Mini Project using Collection Framework)**

---

- 1) create the new project in eclipse by name SeatBookingApplication**
- 2) create the new package in project name as org.techhub.passenger**  
**And create the pojo class under the org.techhub.passenger package.**

## **Source code given below**

---

```
package org.techhub.passenger;

public class Passenger {

    private int seatNo;
    private String name;
    private String contact;
    private String source;
    private String dest;

    public int getSeatNo() {
        return seatNo;
    }

    public void setSeatNo(int seatNo) {
        this.seatNo = seatNo;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getContact() {
        return contact;
    }

    public void setContact(String contact) {
        this.contact = contact;
    }

    public String getSource() {
        return source;
    }
}
```

```

public void setSource(String source) {
    this.source = source;
}

public String getDest() {
    return dest;
}

public void setDest(String dest) {
    this.dest = dest;
}

}

```

**3) Create the package name as org.techhub.travelagency under this package we have to create the class name as TravelAgency**

```

package org.techhub.travelagency;

public class TravelAgency {
}

```

**4) create the new function/method under the TravelAgency class name as addNewPassenger(Passenger passenger)**

---

This method accept the detail of passenger from main method class and store the passenger detail in Stack.

```

package org.techhub.travelagency;
import java.util.*;
import org.techhub.passenger.Passenger;
public class TravelAgency {
    Stack s = new Stack();
    private static int seatNo;
    public TravelAgency() {
        seatNo = 0;
    }
}

```

```

public void addNewPassenger(Passenger p) {
    ++seatNo;
    p.setSeatNo(seatNo);
    s.push(p);
}
}

```

**5) create the class name as TravelAgencyApplication with main method and write create the object of TravelAgency and Passenger class and store detail of the passenger in main method by pressing choice one and pass to TravelAgency class by using addNewPassenger() method.**

### **TravelAgency class source code**

---

```

package org.techhub.travelagency;

import java.util.*;

import org.techhub.passenger.Passenger;

public class TravelAgency {

    Stack s = new Stack();
    private static int seatNo;
    public TravelAgency() {
        seatNo = 0;
    }

    public void addNewPassenger(Passenger p) {
        ++seatNo;

        p.setSeatNo(seatNo);
        s.push(p);
    }
    public ListIterator getAllPassengers()
    {
        ListIterator li=s.listIterator(s.size());
        return li;
    }
}

```

```

public void removePassenger()
{
    boolean b = s.isEmpty();
    if(b)
    {
        System.out.println("No passenger in bus");
    }
    else
    {
        s.pop();
    }
}
public int getPassengerCount() {
    //int count=s.size();
    //return count;
    return s.size();
}
}

```

Main method class logic

---

```

package org.techhub.clientapp;

import java.util.*;

import org.techhub.passenger.Passenger;
import org.techhub.travelagency.TravelAgency;

public class TravelAgencyApplication {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        TravelAgency agency = new TravelAgency();
        Scanner xyz = new Scanner(System.in);
        int choice;
        do {
            System.out.println("1:Add New Passenger");
            System.out.println("2:View All Passenger");
            System.out.println("3:Remove Passengers");
            System.out.println("4:Count the Number of Passengers");
        }
    }
}

```

```

System.out.println("enter your choice");
choice = xyz.nextInt();
switch (choice) {
case 1:
    xyz.nextLine();
    Passenger p = new Passenger();
    System.out.println("enter the name of passenger");
    String name = xyz.nextLine();
    System.out.println("Enter the contact of passenger");
    String contact = xyz.nextLine();
    System.out.println("Enter the source location of
passenger");
    String source = xyz.nextLine();
    System.out.println("Enter the destination location of
passenger");
    String dest = xyz.nextLine();
    p.setName(name);
    p.setContact(contact);
    p.setSource(source);
    p.setDest(dest);
    agency.addNewPassenger(p);
    break;
case 2:
    ListIterator li = agency.getAllPassengers();
    while (li.hasPrevious()) {
        Object obj = li.previous();
        p = (Passenger) obj;
        System.out.println(p.getSeatNo() + "\t" +
p.getName() + "\t" + p.getContact() + "\t" + p.getSource()
+ "\t" + p.getDest());
    }
    break;
case 3:
    agency.removePassenger();
    break;
case 4:
    //int count=agency.getPassengerCount();
    //System.out.println("Now passenger in bus "+count);
    System.out.println("Now passenger in bus "+agency.getPassengerCount());
}

```

```
        default:
            System.out.println("Wrong choice");
    }

} while (true);

}
```

## **Object class and its methods**

---

Object class is parent of every class in java and it is present in `java.lang` package and `java.lang` is default package of java means we not need to import in your project.

### **Why Object class is parent of every class in java?**

---

Because Object class provide the some inbuilt method to us which is used for perform the operation on any class object like as check the equality of different object, convert the object to string, perform object cloning, perform garbage collection on object etc

**Example:**

<b>Programmer</b>	<b>intenrally meaning</b>
<code>class A { } }</code>	<code>class A extends java.lang.Object { } }</code>

Object class provide the some inbuilt method to us to perform operation on object just we have to override this methods in class on which object we want perform operations.

### **Object class methods given below**

---

**boolean equals(Object):** this method is used for perform comparison between two different object using hash code and if objects equal then return true otherwise return false.

**String toString():** this method is used for convert the object in to the string format.

**Object clone()**: this method is used for create the duplicated copy or shadow copy of object

**void finalize()**: this method normally we use for resource cleaning purpose then method call when we perform garbage collection on object.

**void wait()**: this method is used for perform unconditional wait with object in multi threading.

**void wait(int milliseconds)**: this method perform the conditional wait with object in multi threading.

**void notify()**: this method is used for call the waited object in first in first out format or in queue format.

**void notifyAll()**: this method is used for call the all waited threads in last in first out format .

**static block** : static block get executed in program very first even before main method of class and static block call only once when we create the object of class.

**Class getClass()**: this method is used for create the run time instance of class using reflection api

### **Now we will discuss about the equals() method of Object class**

---

Before learn the equals() method we will see the one example and we will use the equals method for cover the limitation of given example.

```
package org.techhub;
class Test
{
    int no;
    Test(int x)
    {
        no=x;
    }
}
```

```
}

public class ObjCompApplication {

    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(100);

        if(t1==t2)
        {
            System.out.println("Objects are equal");
        }
        else
        {
            System.out.println("Objects are not equal");
        }

    }
}
```

If we think about above code we have the two objects t1 and t2 we provide the same value to t1 as 100 and t2 as 100 but program generate the output to us Objects are not equal.

**Q. why program generate the output to us objects are not equal even values of both objects are same?**

---

Because in the case of java Objects not compare on basis of its value object get compare on basis of its hash code.

**Q. what is the Hash Code ?**

---

Hashcode is like as address object or hashCode is unique integer number provided by JVM to every object in Ram and JVM not generate the same hashCode for different objects.

If we want to see the hashCode of object we have the method name as System.identityHashCode(object)

---

## Example

---

```
package org.techhub;
class Test{
    int no;
    Test(int x)
    {no=x;
    }
}
public class ObjCompApplication {
    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(100);
        System.out.println("Hash code of t1 "+System.identityHashCode(t1));
        System.out.println("Hash code of t2 "+System.identityHashCode(t2));
        if(t1==t2)
            {System.out.println("Objects are equal");
        }
        else
            {System.out.println("Objects are not equal");
        }
    }
}
```

```

class Test
{
    int no;
    Test(int x)
    {
        no=x;
    }
}
public class ObjCompApplication {
    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(100);

        System.out.println("Hash code of t1 "+System.identityHashCode(t1));
        System.out.println("Hash code of t2 "+System.identityHashCode(t2));
        if(t1==t2) 1227229563 == 971848845
        {
            System.out.println("Objects are equal");
        }
        else
        {
            System.out.println("Objects are not equal");
        }
    }
}

```

The diagram illustrates the state of two objects, `t1` and `t2`, both initialized with the value `100`. The output of their `toString()` methods is shown in boxes:

- `t1.toString()` outputs `no = 100` and `1227229563`.
- `t2.toString()` outputs `no = 100` and `971848845`.

If we think about above diagram we have the object `t1` with hash code `1227229563` and `t2` with `971848845` and when we execute the statement `t1 == t2` then JVM perform comparison between hash code of object means JVM compare `1227229563` with `971848845` and these hash code are not equal so we get output Objects are not equal If we want to solve this problem Object class provide the two methods to us

`boolean equals(Object)` and `int hashCode()`

## How to override the `equals()` method of Object and Perform Object Comparison

---

Steps to work with `equals()` method

---

### 1) Declare the class and override the `equals()` method from Object class

---

```

class Test
{
    int no;
    Test(int x)
    {

```

```
    no=x;
}
public boolean equals(Object obj)
{ Test t =(Test)obj;
  if(this.no==t.no)
  {
    return true;
  }
  else
  {
    return false;
  }
}
```

## **2) Call the equals() method for Object Comparison purpose**

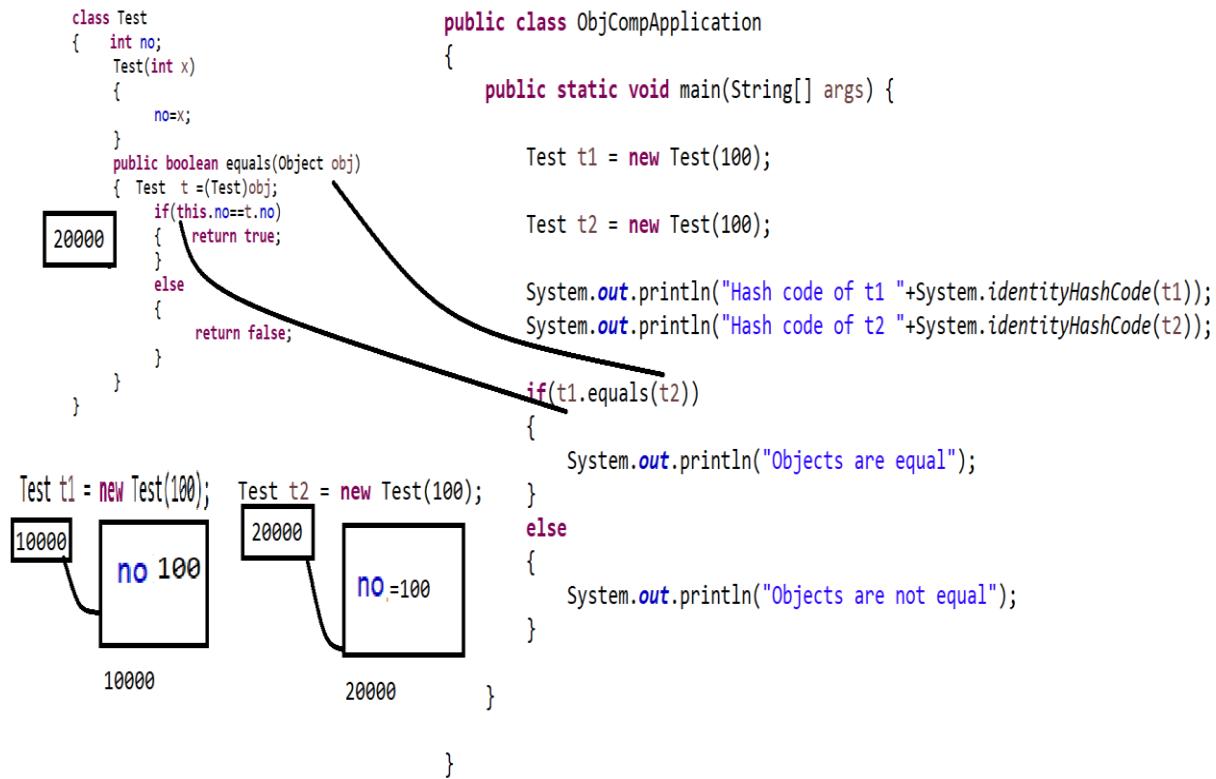
---

**Syntax:** objectname1.equals (objectname2): when we call this method then objectname2 pass as parameter to equals() method and objectname1 is calling object means objectname1 is work as this reference.

---

**We will discuss above concept with following diagram**

---



If we think about above diagram then we have the two objects t1 and t2 so t1 and t2 contain same value i.e no=100 when we call the statement t1.equals(t2) then equals() method get executed then left hand object i.e calling object points to this reference and second object pass as parameter means as per our example t1 points to this and t2 points to t so here t1 and t2 having same value i.e 100 and 100 so equals() method return true value in if statement so we get output objects are equals.

## Rules of object comparison in java

- 1) If two objects are equals then their hash code should be equal
- 2) If values two objects are equal then hash code should be equal and object should be equal.

## Following example demonstrate the above concept

---

```
package org.techhub;
class Test
{ int no;
    Test(int x)
    {
        no=x;
    }
    public boolean equals(Object obj)
    { Test t =(Test)obj;
        if(this.no==t.no)
        { return true;
        }
        else
        {
            return false;
        }
    }
}
public class ObjCompApplication
{
    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(100);
        System.out.println("Hash code of t1 "+System.identityHashCode(t1));
        System.out.println("Hash code of t2 "+System.identityHashCode(t2));
        if(t1.equals(t2))
        {
            System.out.println("Objects are equal");
        }
        else
        {
            System.out.println("Objects are not equal");
        }
    }
}
```

}

**Hash code of t1 1227229563**

**Hash code of t2 971848845**

**Objects are equal**

If we think about above code and output we have two objects are equals but their hashcode are different means above mention rules by java not satisfy here so if we want to satisfy this rule we have to override the hashCode method.

**Note:** JVM not generate the same hashCode for different object internally so if we want to satisfy above mention we have override the hashCode method and generate the hash code when two objects are equal and generate the two different hashCode when have two different objects.

Example

---

```
package org.techhub;
class Test {
    int no;

    Test(int x) {
        no = x;
    }

    public boolean equals(Object obj) {
        Test t = (Test) obj;
        if (this.no == t.no) {
            return true;
        } else {
            return false;
        }
    }
}
```

```

public int hashCode() {
    return no * 1000;
}
}

public class ObjCompApplication {
    public static void main(String[] args) {
        Test t1 = new Test(100);
        Test t2 = new Test(200);
        if (t1.equals(t2)) {
            System.out.println("Objects are equal");
            System.out.println("Hash code of t1 " + t1.hashCode());
            System.out.println("Hash code of t2 " + t2.hashCode());
        } else {
            System.out.println("Objects are not equal");
            System.out.println("Hash code of t1 " + t1.hashCode());
            System.out.println("Hash code of t2 " + t2.hashCode());
        }
    }
}

```

## **Q. Why we need to override hashCode for generate the dummy hashCode when we perform Object Comparision**

---

Some time when we store the user defined objects in Set collection and the rule of Set Collection is cannot store the duplicated data but when we add the new element in Set Collection then Set Collection cross verify hashCode of object and in java every object having different hashCode generated by JVM even objects values are same in this case your set Collection may be store the duplicated data and if we want to avoid this problem we have to override the equals() and hashCode in class whose object want to store in Set Collection.

**WAP to create the Set Collection and in Set Collection we want to store the Employee class objects but we want to employee detail should be unique cannot duplicate employee record.**

---

```
package org.techhub;

public class Employee {

    private int id;
    private String name;
    private int sal;

    public Employee(String name,int id,int sal)
    {
        this.name=name;
        this.id=id;
        this.sal=sal;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getSal() {
```

```

        return sal;
    }

    public void setSal(int sal) {
        this.sal = sal;
    }
    public boolean equals(Object obj)
    {
        Employee e=(Employee)obj;
        if(e.id==this.id &&e.name.equals(this.name) && e.sal==this.sal)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public int hashCode()
    {
        return id*10;
    }
}

```

Mainmethod

---

```

package org.techhub;
import java.util.*;
public class EmployeeSetApplication {
    public static void main(String[] args) {
        LinkedHashSet employeeSet = new LinkedHashSet();
        Employee emp1 = new Employee("Ram",1,1000);
        Employee emp2 = new Employee("Shyam",2,1000);
        Employee emp3 = new Employee("Dinesh",3,1000);
        Employee emp4 = new Employee("Ram",1,1000);
        employeeSet.add(emp1);
        employeeSet.add(emp2);
        employeeSet.add(emp3);
        employeeSet.add(emp4);
    }
}

```

```

for(Object obj:employeeSet)
{
    Employee e=(Employee)obj;
    System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal()+"\t"+e.hashCode());
}
}

```

### **toString() method of Object class**

---

toString() method is used for convert the user defined object in to the string format

Normally we use the toString() in three cases

- 1) Convert the user objects in to the string format
- 2) When we want to display the object content using System.out.println()
- 3) Some time when we store the user defined objects in Collection and when print object then object data not visible in collection so we have to override the toString() and display it.

#### **Example**

---

```
package org.techhub;
```

```

class Student {
    private int id;
    private String name;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {

```

```
this.name = name;
}
public float getPer() {
    return per;
}
public void setPer(float per) {
    this.per = per;
}
private float per;

}

public class StudentApplication {
    public static void main(String[] args) {

        Student s1 = new Student(); //s1 contain id name and percentage
        s1.setId(1);
        s1.setName("Ram");
        s1.setPer(50.5f);
        System.out.println(s1);
    }
}
```

If we think about the above code then we print the s1 object using System.out.println() then s1 not print its content we get output in following fashion.

**org.techhub.Student@4926097b**

But we want to display the content of s1 using System.out.println() then we have to convert the s1 in to the string by using toString() method of Object class means we have to override the toString() of Object class in Student class and return the student content in the form of String and capture and display it.

When we want to display the object content using System.out.println()

Example

```
package org.techhub;
```

```

class Student {
    private int id;
    private String name;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPer() {
        return per;
    }
    public void setPer(float per) {
        this.per = per;
    }
    private float per;

    public String toString()
    {
        return "["+name+","+id+","+per+"]";
    }
}

public class StudentApplication {
    public static void main(String[] args) {

        Student s1 = new Student(); //s1 contain id name and percentage
        s1.setId(1);
        s1.setName("Ram");
    }
}

```

```

    s1.setPer(50.5f);

    System.out.println(s1); //s1.toString()

}

}

```

Some time when we store the user defined objects in Collection and when print object then object data not visible in collection so we have to override the `toString()` and display it

Example with Collection

---

```

package org.techhub;
import java.util.*;
class Student {
    private int id;
    private String name;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPer() {
        return per;
    }
    public void setPer(float per) {
        this.per = per;
    }
    private float per;
}

```

```
public String toString()
{
    return "["+name+","+id+","+per+"]";
}
}

public class StudentApplication {
    public static void main(String[] args) {

        Student s1 = new Student(); //s1 contain id name and percentage
        s1.setId(1);
        s1.setName("Ram");
        s1.setPer(50.5f);

        Student s2 = new Student(); //s1 contain id name and percentage
        s2.setId(2);
        s2.setName("Shyam");
        s2.setPer(60.5f);

        ArrayList al = new ArrayList();
        al.add(s1);
        al.add(s2);
        System.out.println(al);

    }
}
```

**Object clone ()**: this method is used for create the duplicated or clone copy of object.

### Q. Why we need to perform Object cloning in Java?

Some time if we apply more than one reference on single object and if we try to change the object by using any reference then object original state or variables or values get change and if we want to persist the object previous values in this case we can use the object cloning concept.

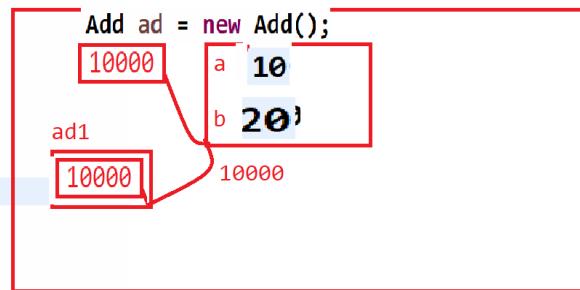
Given Diagram shows the need of object cloning concept

```

class Add
{
    int a, b;           10, 20      Addition is 30
    void setValue(int x, int y) {
        a = x;
        b = y;
    }
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
}

public class AdditionApplication {
    public static void main(String[] args) {
        Add ad = new Add();
        ad.setValue(100, 200);
        Add ad1 = ad;
        ad1.setValue(10, 20);
        ad.showAdd();
    }
}

```



If we think about above diagram we create the statement name as Add  
`ad=new Add()` means we create the object in memory and we store the address of object in ad reference we consider the address of object is 10000 means when we call the method name as `ad.setValue(100,200)` then value stored on 10000 address space means 100 and 200 stored on 10000 address space when we call the statement `Add ad1=ad` means we assign the reference of ad or address of ad in ad1 means address of ad1 is 10000 means ad1 points to the location where ad points means the location or address space of ad and ad1 is same means when we call the method name as `ad1.setValue(10,20)` then 10 and 20 get override in a and b on 10000 address space and when we call the `ad.showAdd()` function then we get answer Addition is 30.  
If we want to solve this problem we have to use the Object cloning concept And we want to perform object cloning concept we have the following steps.

## Steps to work with Object cloning

### 1) Create the class and implement the Cloneable interface

---

```

class Add implements Cloneable
{
    int a, b;
    void setValue(int x, int y) {
        a = x;
        b = y;
    }
}

```

---

```

    }
void showAdd() {
    System.out.printf("Addition is %d\n", a + b);
}
}

```

**2) Create the one user defined method and return the clone of object using clone() method and handle the CloneNotSupportedException**

**class Add implements Cloneable**

```

{   int a, b;
    void setValue(int x, int y) {
        a = x;
        b = y;
    }
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
    public Add getInstance()throws CloneNotSupportedException {
        Object obj = this.clone();
        Add ad=(Add)obj;
        return ad;
    }
}

```

**3) Call the method from object clone get return**

---

**class Add implements Cloneable**

```

{
    int a, b;
    void setValue(int x, int y) {
        a = x;
        b = y;
    }
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
    public Add getInstance()throws CloneNotSupportedException {
        Object obj = this.clone();
        Add ad=(Add)obj;
        return ad;
    }
}

```

```

        }
    }

public class AdditionApplication {
    public static void main(String[] args)throws Exception {
        Add ad = new Add();
        ad.setValue(100, 200);
        Add ad1 = ad.getInstance(); //we get object clone
        ad1.setValue(10, 20);
        ad.showAdd();
    }
}

```

In the case of object cloning we have the two copies of object shadow copy and deep copy. Deep copy means original object created by using new keyword and shadow copy means duplicated copy of object created by clone method.

### **Source Code Example**

---

```

package org.techhub;
class Add implements Cloneable
{ int a, b;
    void setValue(int x, int y) {
        a = x;
        b = y;
    }
    void showAdd() {
        System.out.printf("Addition is %d\n", a + b);
    }
    public Add getInstance()throws CloneNotSupportedException {
        Object obj = this.clone();
        Add ad=(Add)obj;
        return ad;
    }
}
public class AdditionApplication {
    public static void main(String[] args)throws Exception {
        Add ad = new Add();

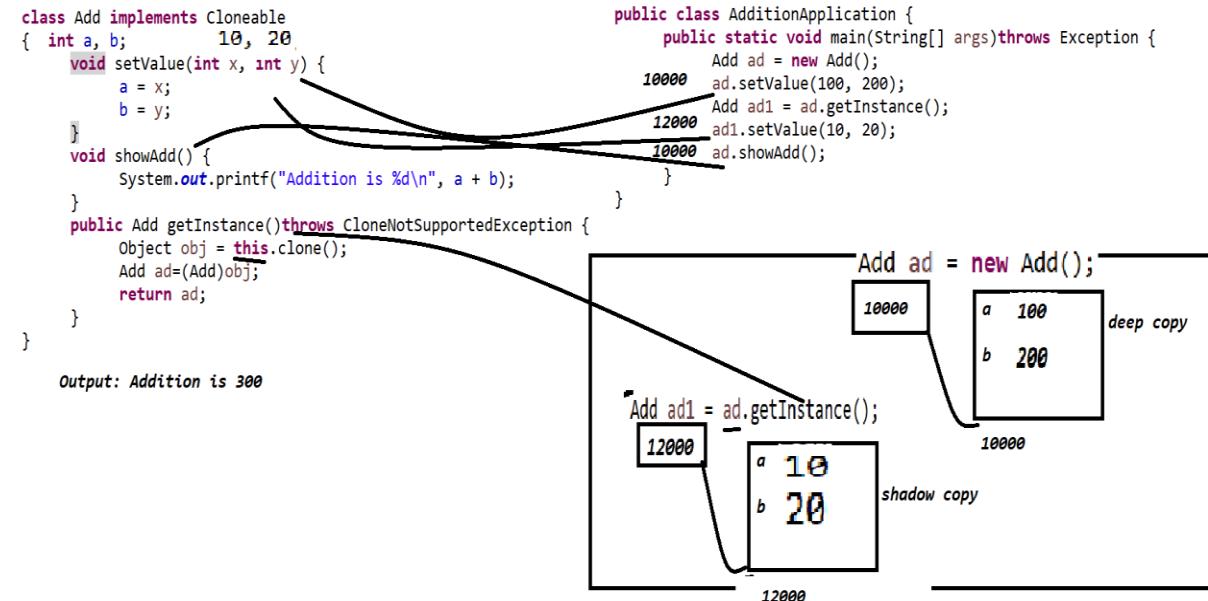
```

```

        ad.setValue(100, 200);
        Add ad1 = ad.getInstance();
        ad1.setValue(10, 20);
        ad.showAdd();
    }
}

}

```



If we think about above code then we have the Statement `Add ad=new Add()` means we create object by using new keyword as per our diagram we have the address of object is 10000 and we store this object in ad reference when we call the method `ad.setValue(100,200)` these values on stored on 10000 address which is created by using `new Add()` means deep copy of object.when we call the statement `Add ad1=ad.getInstance()` so using this method JVM return the duplicated copy of object whose address stored in Add and return the new object address in ad1 means as per our example `this.clone()` statement return the duplicated of object whose address is 10000 and give new address to new object in our example 12000 means ad1 points to 12000 object called as shadow copy or object clone copy so when we call the statement `ad1.setValue(10,20)` it will override on 12000 address so when we execute statement `ad.showAdd()` then we get answer Addtion is 300 because we not perform change on address space 10000

## **Q. what is the diff between object creation by using new keyword and by using clone() method.**

When we create the object by using new keyword then constructor get executed and when we create the object by using clone () then constructor not executed. New object create the new constructed block in heap area of memory and clone() method create the object using previous constructor memory by using new keyword in heap.

### **Static block in Object class**

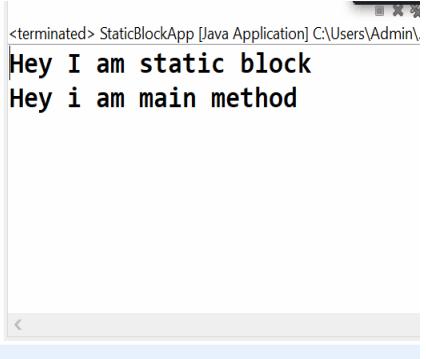
We can write the two blocks or initializer in class

**1) Static block or static initialize:** static block is member of object class  
If we want to work with static block in java we have the some important points.

a) static block call very first in class even before main method of class

```

1 package org.techhub;
2
3 public class StaticBlockApp {
4
5     static
6     {
7         System.out.println("Hey I am static block");
8     }
9     public static void main(String[] args) {
10        // TODO Auto-generated method stub
11        System.out.println("Hey i am main method");
12    }
13
14 }
```



Static block call once means in program means if we declare the static block in particular class and if we create the multiple object of that class but static block run only once in program.

```

class A
{
    A()
    {
        System.out.println("I am constructor");
    }
    static
    {
        System.out.println("I am static block");
    }
}
public class StaticBlockApp {
    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new A();
        A a3 = new A();
    }
}

```

```

I am static block
I am constructor
I am constructor
I am constructor

```

## 2) no static block or non static initializer :

Non static block call at the time of object creation means non static block call before constructor of class and non static block call every time when we create the object of class.

```

class A
{
    A()
    {
        System.out.println("I am constructor");
    }
    static //static initializer or static block
    {
        System.out.println("I am static block");
    }
    { //instance initializer or non static block
        System.out.println("I am non static initializer");
    }
}
public class StaticBlockApp {
    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new A();
        A a3 = new A();
    }
}

```

```

I am static block
I am non static initializer
I am constructor
I am non static initializer
I am constructor
I am non static initializer
I am constructor

```

**void finalize () method :** this method is used for perform garbage collection or resource cleaning purpose this method call automatically when we

initialize the reference of class as null and call System.gc() method .  
System.gc() method destroy the object from memory which is not use by any  
reference called as garbage collection and when we call the System.gc() then  
finalize() method get executed means we can write the logic in finalize() for  
release the resource at the time of object destruction.

## Source Code

---

```
package org.techhub;
class A
{
    A()
    { System.out.println("I am constructor");
    }

    static //static initializer or static block
    {System.out.println("I am static block");
    }

    { //instance initializer or non static block
        System.out.println("I am non static initializer");
    }

    public void finalize()
    { System.out.println("I can release resource");
    }
}

public class StaticBlockApp {
    public static void main(String[] args) {
        A a1 = new A();
        a1=null;
        System.gc();
    }
}
```

## Output

---

I am static block  
I am non static initializer  
I am constructor  
I can release resource

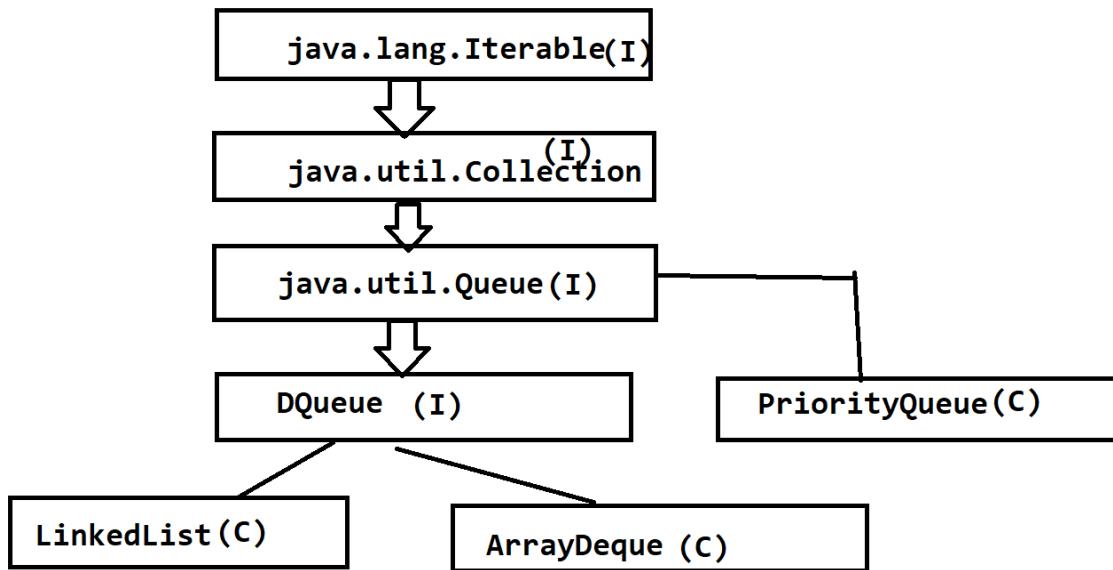
## Queue Collection

---

The Queue interface present in the `java.util` package and extends the Collection interface is used to hold the elements about to be processed in FIFO (First In First Out) order. It is an ordered list of objects with its use limited to insert element at the end of the list and delete the element from start of the list i.e. It follows the FIFO or the First in First out Approach.

### Following Hierarchy Represent the Queue

---



## PriorityQueue

---

PriorityQueue class which is implemented in the Collection Framework provides us a way to process the objects on the priority. It is known that Queue follow the First in First Out algoritham but sometimes the elements of the queue are needed to be processed according to the priority that's when the PriorityQueue comes in to play .

## Example

---

```
package org.techhub;
import java.util.*;
public class QueueCollectionApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Queue<Integer> pQueue=new PriorityQueue<Integer>();
        pQueue.add(10);
        pQueue.add(20);
        pQueue.add(15);
        //print the top element of the priority queue
        System.out.println("Peek eleemnt "+pQueue.peek());
        //printing the top element of the queue and removing
        System.out.println("Before Remove "+pQueue);
        System.out.println(pQueue.poll());
        System.out.println("After Remove "+pQueue);
    }
}
```

## **Generics**

Generics are concept launch by java in JDK 1.5 version and the major goal of generic is to avoid the ClassCastException at run time.

### **Q. What is the ClassCastException**

---

ClassCastException means when we perform type casting between two different objects with each other but when object type or class type not matches with each other then ClassCastException occur at program time.

### **Q. Can you Give RealTime Example Where ClassCastException occur?**

---

If we use the Collection then we have the benefit of Collection is we can store the any type of data in it but sometime this behavior of collection may be generate the ClassCastException at run time . Because Collection return every element in the form Object class and in java Object class is only use for store the data we cannot perform any operation on Object class and if we want to perform any Operation on Object class we have to type the cast the Object in its original type so in this case there is possibility to generate the ClassCastException at run time and if we want to resolve this problem we have the manage the generics with Collection.

### **Example**

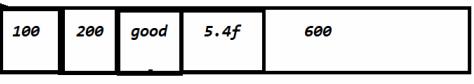
---

```

import java.util.*;
public class CollectionTestApp {
    public static void main(String[] args) {
        ArrayList al= new ArrayList();
        al.add(100);
        al.add(200);
        al.add("good");
        al.add(5.4f);
        al.add(600);
        Integer sum=0;
        for(Object obj:al) {
            Integer val=(Integer)obj;
            System.out.println(val);
            sum = sum + val;
        }
        System.out.printf("Sum is %d\n",sum);
    }
}

```

100  
200  
**Exception in thread "main" java.lang.ClassCastException: class java.lang.String cannot be cast to class java.lang.Integer (java.lang.String and java.lang.Integer are in different packages)**



If we think about above code we store the different type of data in `ArrayList` collection and we try to fetch it and try to convert in `Integer` type from `Object` type so first two values i.e. 100 and 200 get converted because their original type is `Integer` so we can case `Object` class to `Integer` property we have the third element value is good and its original type `String` and we cannot cast `String` type to `Integer` type directly so we get error at run time called as `ClassCastException`

## How to use the Generics in java

---

If we want to use the generics in java we have the following syntax `< >` this is the notation of Generics

**Example:** `ArrayList<Integer> ref = new ArrayList<Integer>();`

We apply the compile time restriction on `ArrayList` as per our Example means if we try to store the other data in `ArrayList` than `integer` then it will generate the error to us at compile time.

## Source code

---

```

package org.techhub;
import java.util.*;
public class GenApplication {

```

```

public static void main(String[] args) {
    ArrayList <Integer>al = new ArrayList<Integer>();
        al.add(100);
        al.add(200);
        al.add(300);
        al.add(400);
        int sum=0;
        for(Integer val: al)
        {
            sum = sum + val;
        }
        System.out.println("Sum is "+sum);
    }
}

```

If we think about above code then there is limitation ArrayList hold only integer value but if we required the String, integer and Float type in ArrayList. In this case we can use the POJO class store the all data in POJO class and store the POJO class objects in collection.

### Example

---

```

package org.techhub;
import java.util.*;
class Data
{
    private String name;
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getId() {
        return id;
    }
}

```

```
}

public void setId(int id) {
    this.id = id;
}

public float getPer() {
    return per;
}

public void setPer(float per) {
    this.per = per;
}

private int id;
private float per;

public Data(String name,int id,float per)
{
    this.name=name;
    this.id=id;
    this.per=per;
}

}

public class GenApplication {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList <Data>al = new ArrayList<Data>();
        Data d1=new Data("Ram",1,90);
        Data d2 =new Data("Shyam",2,80);
        Data d3=new Data("Dinesh",3,70);
        Data d4=new Data("Rajesh",5,60);
        al.add(d1);
        al.add(d2);
        al.add(d3);
        al.add(d4);
```

```
for(Data d:al)
{
    System.out.println(d.getId()+"\t"+d.getName()+"\t"+d.getPer());
}
}
```

## Output

---

1	Ram	90.0
2	Shyam	80.0
3	Dinesh	70.0
5	Rajesh	60.0

## Types of Generics

---

- 1) Class with Generics
- 2) Interface with Generics
- 3) Wild card generics
  - a) Bounded generics
  - b) Unbounded generics

If we want to work with user defined generics we have the some important notation provided by java to us.

**E:** Generic Element normally we refer this notation when we want to use the single value as Generic

**T:** Generic Type element this is normally use for when we want to use the Generic Array

**K:** Generic Key value normally this notation is used for map with key

**V:** Generic value normally this notation is used for map with value.

All notations are inbuilt classes in java

## **Class With generic concept**

---

We can create your own class as generic class in java so if we want to declare your own class as generic class we have following syntax.

```
class classname<GenericNotation>{  
    returntype functionname(GenericNotation ref) {  
    }  
}
```

## **Example**

---

```
package org.techhub;  
class A<E>  
{  
    void setValue(E e)//E can store any type of data but we can apply  
    restriction on using generics  
    { System.out.println("E is "+e);  
    }  
}  
public class OwnGenApplication {  
    public static void main(String[] args) {  
        A <Integer>a1 =new A<Integer>();  
        a1.setValue(100);  
        a1.setValue(600);  
        a1.setValue(500);  
    }  
}
```

## **Generic with Interface**

---

If we use the generics with interface we have the major benefit is we can customize the method parameter type or change method parameter type in every implementer class.

## Example

---

```
package org.techhub;
interface Area<E>
{
    void setRadius(E e);
}
class Circle<E> implements Area<Integer>
{
    @Override
    public void setRadius(Integer e) {
        // TODO Auto-generated method stub
        System.out.println("I am integer radius "+e);
    }
}
class Cirm<E> implements Area<Float>
{
    @Override
    public void setRadius(Float e) {
        System.out.println("I am float Radius "+e);
    }
}
public class GenericWithInterfaceApplication {
    public static void main(String[] args) {
        Circle <Integer> c = new Circle<Integer>();
        c.setRadius(5);
        Cirm cm = new Cirm();
        cm.setRadius(5.6f);

    }
}
```

## Wildcard Generics

---

### There are three types of wild card generics

---

**A) Upper bounded:** To declare an upper bounded wildcard use the wildcard character <?> followed by extends keyword by its upper bound.

**Syntax:** <? extends classname>

### Source code without upper bounded

---

```
package org.techhub;
import java.util.*;
public class UpperBoundedApp {

    public static void main(String[] args) {
        List <Integer>intList=new ArrayList<Integer>();
        intList.add(10);
        intList.add(20);
        intList.add(30);
        calSum(intList); //call calSum function
        List <Double>dblList= new ArrayList<Double>();
        dblList.add(5.4);
        dblList.add(6.5);
        dblList.add(7.6);
        dblList.add(3.4);
        calSum(dblList);
    }
    public static void calSum(List<Integer> list)
    {   for(Integer val:list)
        {
            System.out.println(val);
        }
    }
}
```

If we think about above code then it will generate the compile time error because in calSum(List<Integer> list) this method can accept the only list of

integer but from main method we pass the list of double in calSum() from calling point so it is not acceptable here because strictly specify in calSum() List only works with Integer parameter so if we want to pass any kind of Number means if we want to pass Integer in list, float in list,double list etc then we can use the upper bounded generics in java. So as per our example we need to use the Number class with List using upper bounded generics Because in java Wrapper classes Number class is parent of Integer ,Float and Double etc means when we use the Number with upper bounded in List then any child of Number class is acceptable in List Collection

## Source Code with Upper Bounded Generics

---

```

package org.techhub;
import java.util.*;
public class UpperBoundedApp {

    public static void main(String[] args) {
        List <Integer>intList=new ArrayList<Integer>();
        intList.add(10);
        intList.add(20);
        intList.add(30);
        System.out.println("Integer list is ");
        calSum(intList); //call calSum function
        List <Double>dblList= new ArrayList<Double>();
        dblList.add(5.4);
        dblList.add(6.5);
        dblList.add(7.6);
        dblList.add(3.4);
        System.out.println("Float list is");
        calSum(dblList);
    }
    public static void calSum(List<? extends Number> list)
    {
        for(Number val:list)
        {System.out.println(val);
        }
    }
}

```

}

**b) Lower bounded:** The way upper bounded wildcard restricts the unknown type to be a specific type or subtype of that type same way lower bounded wildcard restrict the unknown type to be a specific type or super type of that type.

**Syntax: <? Super classname>:** here we can use the all super classes of specified child mention in Lower bounded generics.

### Example

---

```
package org.techhub;
import java.util.*;
public class LowerBoundedApp {
    public static void main(String[] args) {
        List <Number> numList=Arrays.asList(10,20,30,40,50);
        System.out.println("Number with list");
        calSum(numList);
        System.out.println("Object with list");
        List<Object> objList=Arrays.asList(5.6,6.5);
        calSum(objList);
    }
    public static void calSum(List<? super Integer> list)
    {
        for(Object val:list)
        {
            System.out.println (val);
        }
    }
}
```

### c) Unbounded wild cards

---

Unbounded wild cards as name suggest wild card without any upper or lower bound. Unbounded wildcard type is specific using the wildcard character <?> for Example List<?> This is called of List of unknown type

There are two scenarios where an unbounded wildcard are useful

- 1) If you are writing method that can be implemented using any functionality provided by in Object class.
- 2) When the code using methods in Generics class That don't depend on the type parameter e.g. List.clear (), List.isEmpty ()

Source code Example

---

```
package org.techhub;
import java.util.*;
public class WildCardGenApp {
    public static void main(String[] args) {
        List <Integer> numList=Arrays.asList(10,20,30,40,50);
        System.out.println("Number with list");
        calSum(numList);
        System.out.println("Object with list");
        List<String> objList=Arrays.asList("good","bad","test");
        calSum(objList);
    }
    public static void calSum(List<?> list)
    {
        for(Object val:list)
        {
            System.out.println(val);
        }
    }
}
```

### **Question 1**

---

WAP to retrieve words from Following String and Store in Vector and remove the stop words from Vector and display the Vector before remove and after remove.

**String message=**" good morning india India is my country I am Indian this is the India "

We have to separate the words from above mention message string and store in Vector and remove the stop words from Vector

**Stop word list:** as per our example we have the stop words: my, I am this is

Means we have to remove these words from Vector Collection

### **Question 2**

---

WAP to store the 10 values in Vector and find the occurrence of every element in Vector.

e.g. suppose if we store the 10 values like as 10 20 30 10 20 30 10 10 20 10

10 ---- 4

20 ---- 3

30 ---- 3

### **Question 3**

---

WAP to store the Player data in Vector using POJO class with three fields like as name, id and runs and perform the following operation

1. Add New Player in Vector
2. View All Player
3. Delete the player using id

4. Search player by using id
- 5) Find the player list whose runs are same.

#### **Question 4**

---

WAP to store the 10 strings in Vector and count the words of every string

#### **Example**

---

Suppose Vector contain the following string

Good morning India

Good Afternoon India

I am Indian I like India

#### **Output**

---

First Statement Word count: 3

Second Statement Word count: 3

Third Statement work count: 6

#### **Question 5**

---

WAP to store the 5 words in Vector and perform the steaming operation on every word and restore in Vector and display it.

#### **Example**

---

Store following five words Vector:

marker: remove the er from marker means store mark in Vector

Morning: remove the ing from morning and store the morn in Vector

Evening: remove the ing from evening and store even in Vector

Testing: remove the ing from testing and store test in Vector

Shocked: remove the ed from shocked and store shock in Vector

So output like as

Mark, morn, even, test, shock

## **Important Question on Vector**

---

---

### **Q. what is the default capacity of Vector?**

---

The default capacity of Vector is 10.

### **Q. what is the load factor in Collection?**

---

The load factor is the measure that decides when to increase the capacity collection.

**Example:** the load factor of Vector is 1 means 100% means when the capacity of Vector is 100% cross then Vector allocate double memory than its current capacity means Vector increase its capacity 100%.

### **Q. Is Vector is Thread Safe?**

---

Vector is a Thread safe collection because methods of Vector are synchronized so Vector is thread safe.

### **Q. Is Vector legacy Collection?**

---

Yes Vector is legacy Collection Because Vector is not added in Collection framework before JDK 1.1 but later JDK 1.2 it is added in Collection Framework.

### **Q. Explain the all constructor of Vector?**

---

**Vector contain 4 types of Constructor**

**1) Vector():**

**2) Vector(int initialCapacity)**

**3) Vector(int initialCapacity,int incrementalSize);**

**4) Vector(Collection collection);**

**Vector ()**: this constructor is used for create the Vector with default capacity provided by Vector and the default capacity of Vector is 10.

**Vector (int initialCapacity)**: this constructor is used for create a vector with a initial capacity provided by user.

**Note:** if user uses these two constructors and Vector cross its capacity then Vector allocate double memory then its current capacity if user want to set the increment capacity as per his choice then Vector contain third constructor.

**Vector (int initialCapacity, int incrementalCapacity)**: this constructor contain two parameter

**int initialCapacity**: this parameter is used for set the initialCapacity

**int incrementCapacity**: this parameter decide how much capacity increment if current capacity cross.

### Example

---

```
import java.util.*;

public class TestVectorApp

{ public static void main(String x[])

    {   Vector  v = new Vector(5,2);

        v.add(100);

        v.add(200);

        v.add(300);

        v.add(400);

        v.add(500);

        v.add(600);

        v.add(700);
```

```
        System.out.println("Now Capacity is "+v.capacity());  
    }  
}
```

## **Q.what is the time complexity?**

---

Time Complexity of an algorithm qualifies amount of time taken by an algorithm to runs a function of the length of input.

## **Q. what is the time complexity of Vector in Java?**

---

- 1) Retrieving element from a specific position O(1)
- 2) Adding and Removing element from the end constant time complexity O(1)

**Note:** Adding and Removing element from other position is expensive leaner O( $n-1$ ) where  $n$  is the number of elements and  $i$  is the index of element added or removed these operations are more expensive because you have to shift all element at index  $i$  and higher over by one element.

## **Q. how does Vector grow internally in Java ?**

---

A Vector will grow based on its capacity increment value new length of Vector is calculated as given below.

```
newlength = capacityIncrement==0? existingSize+existingSize:  
existingSize+capacityIncrement;
```

Once new size of Vector is calculated `Arrays.copyOf (existingArray, newlength)` is called the existing array and new length of array passed to the method. This method will create a new array of given length and copy data in to the new array.

## **Practical Assignment on Vector**

---

**Q. WAP to create Vector of integer values and display it using its index.**

**Output should like as**

---

<b>Index</b>	<b>Value</b>
0	100
1	200
3	300

**Q. WAP to create Vector with 10 values and remove value using its index from vector?**

---

e.g. Vector v= new Vector ();

```
v.add (10);  
v.add (20);  
v.add (30);
```

**Output like as**

---

Enter the index

2

If index found then show the message value is deleted successfully and if index not found the show message value not deleted

**Q. WAP to create a Vector with a 10 floating values and calculate its sum?**

---

Example: Vector v = new Vector ();

```
v.add(5.4);  
v.add(2.5);
```

**Output:** Sum of Vector is 7.9

**Q. WAP to create the vector and store the 10 string values in it and count the character on every index?**

---

```
Vector v =new Vector ();
```

```
    v.add("Good");
```

```
    v.add("Bad");
```

**Output should look like**

---

Index	Character Count
-------	-----------------

---

0	----- > 4
---	-----------

1	----- > 3
---	-----------

**Q. WAP program to create a Vector with Employee Collection with id , name and salary and perform following operation**

1. Add New Employee in Vector
2. View All Employee
3. Search Employee by id
4. Deleted Employee by using its id

**Q. WAP to create a Vector and Store the sentences in Vector and count the word on every index?**

---

**Example**

---

```
Vector v = new Vector ();
```

```
    v.add ("Good Morning India");
```

```
    v.add ("Good Afternoon India");
```

```
v.add ("Good evening India");
```

---

Index	Word
-------	------

---

0	3
1	3
2	3

---

**Q. WAP to create Vector with a sentence and calculate its word on every index as well as count the number character on every index.**

---

```
Vector v= new Vector ();  
v.add("Good Morning India");  
v.add ("Good Morning Pune");
```

---

**Output**

---

Index	Word Count	Character Count
0	3	18
1	3	18

## Differentiate between Iterator and Enumeration?

---

### Iterator

- It can traverse both legacy as well as non legacy elements.
- It is slower than Enumeration.
- It can perform `remove` operations while traversing the collection.
- It is fail-fast.

### Enumeration

- It can traverse only legacy elements.
- It is faster than an Iterator.
- It can perform only `traverse` operations on the collection.
- It is not a fail-fast.

## What is Fail-fast and Fail-safe iterations in Java Collections?

---

Using iterations we can traverse over the collections objects. The iterators can be either fail-safe or fail-fast. *Fail-safe* iterators means they will not throw any exception even if the collection is modified while iterating over it. Whereas *Fail-fast* iterators throw an exception(*ConcurrentModificationException*) if the collection is modified while iterating over it.

Consider an example:

```
ArrayList<Integer> integers = new ArrayList<>();
integers.add(1);
integers.add(2);
integers.add(3);
Iterator<Integer> itr = integers.iterator();
while (itr.hasNext()) {
    Integer a = itr.next();
    integers.remove(a);
}
```

As arrays are fail-fast above code will throw an exception.

First a will have value = 1, and then 1 will be removed in same iteration.

Next when a will try to get next(), as the modification is made to the list, it will throw an exception here.

**However if we use an fail-safe collection e.g. `CopyOnWriteArrayList` then no exception will occur:**

```
List integers = new CopyOnWriteArrayList();
integers.add(1);
integers.add(2);
```

```
integers.add(3);
Iterator itr = integers.iterator();
while (itr.hasNext()) {
    Integer a = itr.next();
    integers.remove(a);
}
```

Here if we print the element a, then all the elements will be printed.

#### **Fail-Fast Iterators internal working:**

Every fail fast collection has a **modCount** field, to represent how many times the collection has changed/modified.

So at every modification of this collection we increment the modCount value. For example the modCount is incremented in below cases:

1. When one or more elements are removed.
2. When one or more elements are added.
3. When the collection is replaced with other collection.
4. When the collection is sorted.

So everytime there is some change in the collection structure, the mod count is incremented.

Now the iterator stores the modCount value in the initialization as below:

```
int expectedModCount = modCount;
```

Now while the iteration is going on, expectedModCount will have old value of modCount.

If there is any change made in the collection, the modCount will change and then an exception is thrown using:

```
if (modCount != expectedModCount)
    throw new ConcurrentModificationException();
```

This code is used in most of the iterator methods e.g.

1. next()
2. remove()
3. add()

So if we make any changes to the collection, the modCount will change, and expectedModCount will not be hence equal to the modCount. Then if we use any of the above methods of iterator, the ConcurrentModificationException will be thrown.

**Note:** If we remove/add the element using the remove() or add() of iterator instead of collection, then in that case no exception will occur. It is because the remove/add methods of iterators call the

*remove/add method of collection internally, and also it reassigns the expectedModCount to new modCount value*

```
ArrayList.this.remove(lastRet);
cursor = lastRet;
lastRet = -1;
expectedModCount = modCount; and ArrayList.this.add(i, e);
expectedModCount = modCount;
```

So the below code is safe as we are removing the element from the iterator here:

```
Iterator<Integer> itr = integers.iterator();
while (itr.hasNext()) {
if (itr.next() == 2) {
// will not throw Exception
itr.remove();
}
}
```

Whereas the below code will throw an exception as we are removing the element from the collection here:

```
Iterator<Integer> itr = integers.iterator();
while (itr.hasNext()) {
if (itr.next() == 3) {
// will throw Exception on
// next call of next() method
integers.remove(3);
}
}
```

### Fail-Safe Iterators internal working:

Unlike the fail-fast iterators, these iterators traverse over the clone of the collection. So even if the original collection gets structurally modified, no exception will be thrown. E.g. in case of CopyOnWriteArrayList the original collections is passed and is stored in the iterator:

```
public Iterator<E> iterator() {  
    return new COWIterator<E>(getArray(), 0);  
}
```

here iterator() method returns the iterator of the CopyOnWriteArrayList. As we can see, it passes the getArray() in the constructor of the iterator. This getArray() has all the collection elements.

Now the iterator(COWIterator here) will save this to traverse upon as:

```
COWIterator(Object[] elements, int initialCursor) {  
    cursor = initialCursor;  
    snapshot = elements;  
}
```

So the original collection elements are saved in the snapshot field variable. So all the iterator methods will work on this snapshot method. So even if there is any change in the original collection, no exception will be thrown. But note the the iterator will not reflect the latest state of the collection

```
Iterator<Integer> itr = integers.iterator();  
while (itr.hasNext()) {  
    int a = itr.next();  
    if (a == 1) {  
        integers.remove(Integer.valueOf(a));  
    }  
    System.out.print(a);  
}
```

So as we are removing the element from the collection. the collection now has only elements **2 and 3** in it. But the iterator will print all the elements **1,2,3** because it traverses over the snapshot of the collection elements. We can print the collection elements after the above code. It will print only **2,3** as:

```
Iterator<Integer> itr = integers.iterator();  
while (itr.hasNext()) {  
    int a = itr.next();  
    System.out.print(a);  
}
```

**Q.What is the diff between Iterator and ListIterator in java**

---

Iterator	ListIterator
Can traverse elements present in Collection only in the forward direction.	Can traverse elements present in Collection both in forward and backward directions.
Helps to traverse Map, List and Set.	Can only traverse List and not the other two.
Cannot modify or replace elements present in Collection	We can modify or replace elements with the help of set(E e)
Cannot add elements and it throws ConcurrentModificationException.	Can easily add elements to a collection at any time.
Certain methods of Iterator are next(), remove() and hasNext().	Certain methods of ListIterator are next(), previous(), hasNext(), hasPrevious(), add(E e).

## **Q. How to Remove Duplicates from ArrayList in Java?**

---

ArrayList is the most popular implementation of the List interface from Java's Collection framework, but it allows duplicates. Though there is another collection called Set which is primarily designed to store unique elements, there are situations when you receive a List like ArrayList in your code and you need to ensure that it doesn't contain any duplicate before processing. Since with ArrayList you cannot guarantee uniqueness, there is no other choice but to remove repeated elements from ArrayList. The simplest approach to remove repeated objects from ArrayList is to copy them to a Set e.g. HashSet and then copy it back to ArrayList. This will remove all duplicates without writing any more code. One thing to noted is that, if original order of elements in ArrayList is important for you, as List maintains insertion order, you should use LinkedHashSet because HashSet doesn't provide any ordering guarantee.

**Java Program to Remove duplicates from ArrayList**

Here is our sample program to learn how to remove duplicates from ArrayList. The steps followed in the below example are:

- Copying all the elements of ArrayList to LinkedHashSet. Why we choose LinkedHashSet? Because it removes duplicates and maintains the insertion order.
- Emptying the ArrayList, you can use clear() method to remove all elements of ArrayList and start fresh.
- Copying all the elements of LinkedHashSet (non-duplicate elements) to the ArrayList.

Please find below the complete code:

```
import java.util.ArrayList;  
  
import java.util.LinkedHashSet;  
  
import java.util.List;  
  
import java.util.Set;
```

```
public class ArrayListDuplicateDemo
{
    public static void main(String args[])
    {
        List<Integer> primes = new ArrayList<Integer>();
        primes.add(2);
        primes.add(3);
        primes.add(5);
        primes.add(7); //duplicate
        primes.add(7);
        primes.add(11);

        System.out.println("list of prime numbers : " + primes);

        Set<Integer> primesWithoutDuplicates = new
        LinkedHashSet<Integer>(primes);

        primes.clear();

        primes.addAll(primesWithoutDuplicates);

        System.out.println("list of primes without duplicates : " + primes);
    }
}
```

Output list of prime numbers: [2, 3, 5, 7, 7, 11]

List of primes without duplicates: [2, 3, 5, 7, 11]

## **Q. How to reverse ArrayList in Java with Example ?**

---

You can reverse ArrayList in Java by using the reverse() method of java.util.Collections class. This is one of the many utility methods provided by the Collections class e.g. sort () method for sorting ArrayList. The Collections. reverse () method also accepts a List, so you not only can reverse ArrayList but also any other implementation of List interface e.g. LinkedList or Vector or even a custom implementation. This method has a time complexity of O(n) i.e. it runs on linear time because it uses ListIterator of the given list. It reverses the order of an element in the specified list.

By the way, you cannot reverse an ArrayList using this method if the specified ArrayList or its ListIterator doesn't support set() operation. It switches between two algorithms depending upon the size of List or if List implements RandomAccess interface like ArrayList.

If a number of elements in List are less than REVERSE\_THRESHOLD, which is equal to 18 then it uses for loop for swapping elements otherwise it uses list iterator. If you want to learn more about *how the reverse() method of Collections works*, you can see it's code from JDK itself or in the next section

By the way, this is a typesafe generic method and you can use it to reverse Integer, String, Float or any kind of List in Java. You can also see the classic book Core Java Volume 1 - Fundamentals by Cay S. Horstmann to learn more about key classes of the Java Development Kit.

### **Java Program to reverse ArrayList in Java**

You can see that we have added numbers on increasing order but after calling reverse() method, it prints them on decreasing order. Unlike popular misconception, Comparable or Comparator is not used while reversing ArrayList in Java. Though they are used if you want to sort Array in Java.

```
import java.util.ArrayList;  
  
import java.util.Collections;  
  
public class ArrayListReverseDemo  
{ public static void main(String args[])  
  
{ ArrayList<String> listOfInts = new ArrayList<>();  
  
listOfInts.add("1");  
  
listOfInts.add("2");  
  
listOfInts.add("3");  
  
listOfInts.add("4");  
  
listOfInts.add("5");  
  
System.out.println("Before Reversing : " + listOfInts);  
  
Collections.reverse(listOfInts);  
  
System.out.println("After Reversing : " + listOfInts);  
}  
}
```

### How Collections.reverse() method works in Java

Here is the code snippet from `java.util.Collections` class which you can use to *reverse an ArrayList* or any kind of List in Java. You can see that it uses `set()` method of List interface for swapping elements and that's why you cannot reverse a read-only ArrayList because it doesn't support `set()` operation.

### Logic of Collections.reverse() method

Here is the logic and explanation of how the Collections.reverse() method works and how it reverse the given collection.

```
public static void reverse(List<?> list)

{ int size = list.size();

    if (size < REVERSE_THRESHOLD || list instanceof RandomAccess)

    {

        for (int i=0, mid=size>>1, j=size-1; i<mid; i++, j--)

            swap(list, i, j);

    }

    else

    { ListIterator fwd = list.listIterator();

        ListIterator rev = list.listIterator(size);

        for (int i=0, mid=list.size()>>1; i<mid; i++)

        { Object tmp = fwd.next();

            fwd.set(rev.previous()); rev.set(tmp);

        }

    }

}
```

## **Q. Difference between Array vs ArrayList in Java ?**

Both Array and Array List are used to store elements, which can be either primitive or objects in case of Array and only objects in case of ArrayList in Java. The main *difference between Array vs ArrayList in Java* is the static nature of the Array and the dynamic nature of ArrayList. Once created you can not change the size of Array but ArrayList can re-size itself when needed. Another notable difference between ArrayList and Array is that Array is part of core Java programming and has special syntax and semantics support in Java, While ArrayList is part of the Collection framework along with other popular classes like Vector, Hashtable, HashMap or LinkedList.

### **Array vs ArrayList in Java**

1) First and Major difference between Array and ArrayList in Java is that Array is a fixed-length data structure while ArrayList is a variable-length Collection class. You can not change the length of Array once created in Java but ArrayList re-size itself when gets full depending upon the capacity and load factor.

Since ArrayList is internally backed by Array in Java, any resize operation in ArrayList will slow down performance as it involves creating a new Array and copying content from the old array to the new array.

2) Another difference between Array and ArrayList in Java is that you can not use Generics along with Array, as Array instance knows about what kind of type it can hold and throws ArrayStoreException if you try to store type which is not convertible into the type of Array. ArrayList allows you to use Generics to ensure type safety.

3) You can also compare Array vs ArrayList on *How to calculate the length of Array or size of ArrayList*. All kinds of Array provides length variable which denotes the length of Array while ArrayList provides size() method to calculate the size of ArrayList in Java.

4) One more major difference between ArrayList and Array is that, you can not store primitives in ArrayList, it can only contain Objects. While Array can contain both primitives and Objects in Java. Though Autoboxing of Java 5 may give you an impression of storing primitives in ArrayList, it actually automatically converts primitives to Object. e.g.

5) Java provides add() method to insert an element into ArrayList and you can simply use the assignment operator to store element into Array e.g. In order to store Object to specified position use

```
Object[] objArray = new Object[10];  
objArray[1] = new Object();
```

6) One more difference on Array vs ArrayList is that you can create an instance of ArrayList without specifying size, Java will create Array List with default size but it's mandatory to provide the size of Array while creating either directly or indirectly by initializing Array while creating it. By the way, you can also initialize ArrayList while creating it.

## **Q. How to Synchronize ArrayList in Java with Example?**

---

ArrayList is a very useful Collection in Java, I guess most used one as well but it is not synchronized. What this mean? It means you cannot share an instance of ArrayList between multiple threads if they are not just reading from it but also

writing or updating elements. So *how can we synchronize ArrayList?* Well, we'll come to that in a second but did you thought why ArrayList is not synchronized in the first place? Since multi-threading is a core strength of Java and almost all Java programs have more than one thread, why Java designer does not make it easy for ArrayList to be used in such an environment?

The answer lies in performance, there is a performance cost associated with synchronization, and making ArrayList synchronized would have made it slower. So, they definitely thought about it and left ArrayList as non-synchronized to keep it fast, but at the same time they have provided easy ways to make it synchronized

The JDK Collections class has several methods to create synchronized List, Set and Map and we will use Collections.synchronizedList() method to make our ArrayList synchronized. This method accepts a List which could be any implementation of the List interface e.g. ArrayList, LinkedList, and returns a synchronized (thread-safe) list backed by the specified list. So you can also use this technique to make LinkedList synchronized and thread-safe in Java.

### Synchronized List and Iteration

One of the main challenges of sharing ArrayList between multiple threads is how to deal with a situation where one thread is trying to access the element which is removed by other. If you are using methods like get(index) or remove(index) method to retrieve or remove elements then it's also possible that other thread may also be removing other elements.

This means you cannot call get(index) or remove(index) reliably without checking the size of the list first and then you also needs to provide extra synchronization between your call to size() and remove(int index).

In order to guarantee serial access, it is critical that all access to the backing list is accomplished through the returned list and it is imperative that the user manually synchronizes on the returned list when iterating over it as shown in the following sample code :

```
List list = Collections.synchronizedList(new ArrayList());  
  
...  
synchronized(list) {  
    Iterator i = list.iterator(); // Must be in synchronized block  
    while (i.hasNext())  
        foo(i.next());  
}
```

## Java Program to Synchronize ArrayList

Here is the complete example of synchronizing an ArrayList in Java. If you look, we have created a List of String and added a couple of elements to it. Then we passed this ArrayList to Collections.synchronizedList() method, which returned a thread-safe, synchronized version of the backed list.

You can now safely share this list among multiple threads, but you need to be a little bit careful while retrieving or removing elements from ArrayList. In order to have safe access, you should use Iterator for getting and removing elements from the list and that too in a synchronized manner as shown in this example.

```
import java.util.ArrayList;  
import java.util.Collections;  
  
import java.util.Iterator;  
  
import java.util.List;  
  
public class SynchronizedArrayListDemo  
{ public static void main(String args[])  
{ // An ArrayList which is not synchronize  
List<String> listOfSymbols = new ArrayList<String>();
```

```
listOfSymbols.add("RELIANCE");

listOfSymbols.add ("TATA");

listOfSymbols.add ("TECHMAH");

listOfSymbols.add ("HDFC");

listOfSymbols.add ("ICICI"); // Synchronizing ArrayList in Java

listOfSymbols = Collections.synchronizedList (listOfSymbols);

synchronized (listOfSymbols)

{ Iterator<String> myIterator = listOfSymbols.iterator ();

  while(myIterator.hasNext())

  { System.out.println(myIterator.next());

    }

  }

}

}
```

## **Q. When to use ArrayList vs LinkedList in Java?**

---

ArrayList and LinkedList are two popular concrete implementations of the List interface from Java's popular Collection framework. Being List implementation both ArrayList and LinkedList are ordered, the index-based and allows duplicate. Despite being from the same type of hierarchy there are a lot of differences between these two classes which makes them popular among Java interviewers. The main difference between ArrayList vs LinkedList is that the former is backed by an array while the latter is based upon the linked list data structure, which makes the performance of add(), remove(), contains(), and iterator() different for both ArrayList and LinkedList. The difference between ArrayList and LinkedList is also an important Java collection interview question, as much popular as Vector

vs ArrayList or HashMap vs HashSet in Java. Sometimes this is also asked as for when to use LinkedList and when to use ArrayList in Java.

In this Java collection tutorial, we will compare LinkedList vs ArrayList on various parameters which will help us to decide when to use ArrayList over LinkedList in Java. We will not focus on the array and linked list data structure much, which is subject to data structure and algorithm, we'll only focus on the Java implementations of these data structures which are ArrayList and LinkedList.

## **When to use ArrayList vs LinkedList in Java**

---

Before comparing differences between ArrayList and LinkedList, let's see What is common between ArrayList and LinkedList in Java:

- 1) Both ArrayList and LinkedList are an implementation of the List interface, which means you can pass either ArrayList or LinkedList if a method accepts the java.util.List interface.
- 2) Both ArrayList and LinkedList are not synchronized, which means you cannot share them between multiple threads without external synchronization. See here to know.
- 3) ArrayList and LinkedList are ordered collection e.g. they maintain insertion order of elements i.e. the first element will be added to the first position.
- 4) ArrayList and LinkedList also allow duplicates and null, unlike any other List implementation e.g. Vector.
- 5) An iterator of both LinkedList and ArrayList are fail-fast which means they will throw ConcurrentModificationException if a collection is modified structurally once the Iterator is created. They are different than CopyOnWriteArrayList whose Iterator is fail-safe.

### Difference between LinkedList and ArrayList in Java

Now let's see some differences between ArrayList and LinkedList and when to use ArrayList and LinkedList in Java.

#### 1. Underlying Data Structure

The first difference between ArrayList and LinkedList comes with the fact that ArrayList is backed by Array while LinkedList is backed by LinkedList. This will lead to further differences in performance.

## 2. LinkedList implements Deque

Another difference between ArrayList and LinkedList is that apart from the List interface, LinkedList also implements the Deque interface, which provides first in first out operations for add() and poll() and several other Deque functions.

## 3. Adding elements in ArrayList

Adding an element in ArrayList is O(1) operation if it doesn't trigger re-size of Array, in which case it becomes O(log(n)), On the other hand, appending an element in LinkedList is O(1) operation, as it doesn't require any navigation.

## 4. Removing an element from a position

In order to remove an element from a particular index e.g. by calling remove(index), ArrayList performs a copy operation which makes it close to O(n) while LinkedList needs to traverse to that point which also makes it O(n/2), as it can traverse from either direction based upon proximity.

## 5. Iterating over ArrayList or LinkedList

Iteration is the O (n) operation for both LinkedList and ArrayList where n is a number of an element.

## 6. Retrieving element from a position

the get (index) operation is O (1) in ArrayList while its O (n/2) in LinkedList, as it needs to traverse till that entry. Though, in Big O notation O (n/2) is just O (n) because we ignore constants there.

## 7. Memory

LinkedList uses a wrapper object, Entry, which is a static nested class for storing data and two nodes next and previous while ArrayList just stores data in Array. So memory requirement seems less in the case of ArrayList than LinkedList except for the case where Array performs the re-size operation when it copies content from one Array to another.

If Array is large enough it may take a lot of memory at that point and trigger Garbage collection, which can slow response time.

From all the above differences between ArrayList vs LinkedList, It looks like ArrayList is the better choice than LinkedList in almost all cases, except when you

do a frequent add () operation than remove (), or get () .

It's easier to modify a linked list than ArrayList, especially if you are adding or removing elements from start or end because the linked list internally keeps references of those positions and they are accessible in O(1) time.

In other words, you don't need to traverse through the linked list to reach the position where you want to add elements, in that case, addition becomes an O(n) operation. For example, inserting or deleting an element in the middle of a linked list

## **Q. Difference between ArrayList and HashSet in Java?**

---

1. First and most important difference between ArrayList and HashSet is that ArrayList implements List interface while HashSet implements Set interface in Java.
2. Another difference between ArrayList and HashSet is that ArrayList allows duplicates while HashSet doesn't allow duplicates. This is the side effect of first difference and property of implementing List and Set interface.
3. The differences between ArrayList and HashSet is that ArrayList is an ordered collection and maintains insertion order of elements while HashSet is an unordered collection and doesn't maintain any order.
4. The difference between ArrayList and HashSet is that ArrayList is backed by an Array while HashSet is backed by a HashMap instance
5. Fifth difference between HashSet and ArrayList is that it's index-based you can retrieve objects by calling get(index) or remove objects by calling remove(index) while HashSet is completely object-based. HashSet also doesn't provide the get() method.

Similarities ArrayList and HashSet

1) Both ArrayList and HashSet are non synchronized collection classes and not meant to be used in multi-threading and concurrent environments. You can make ArrayList and HashSet synchronized by

Using Collections.synchronizedCollection () just like we Make ArrayList and HashSet read-only other days

2) Both ArrayList and HashSet can be traversed using Iterator. This is in fact a preferred way if you want to perform operations on all elements.

3) Iterator of ArrayList and HashSet both are fail-fast, i.e. they will throw ConcurrentModificationException if ArrayList or HashSet is modified structurally once Iterator has been created.

## **Q. Difference between Vector and ArrayList in Java?**

---

ArrayList and Vector are the two most widely used Collection classes in Java and are used to store objects in an ordered fashion. Every Java programmer which is introduced to Java Collection Framework either started with Vector or ArrayList

Vector vs ArrayList in Java

### **1. Synchronization**

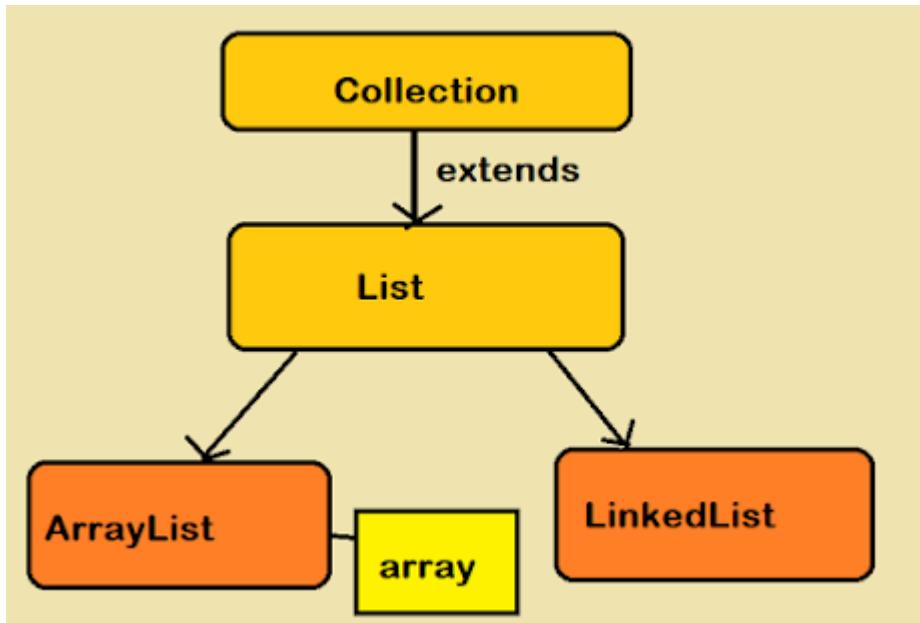
The first and most common *difference between Vector vs ArrayList* is that Vector is synchronized and thread-safe while ArrayList is neither Synchronized nor thread-safe. Now, What does that mean? It means if multiple thread try to access Vector same time they can do that without compromising Vector's internal state. Same is not true in case of ArrayList as methods like add(), remove() or get() is not synchronized.

### **2. Speed**

The second major difference on Vector vs ArrayList is Speed, which is directly related to previous difference. Since Vector is synchronized, its slow and ArrayList is not synchronized its faster than Vector.

### 3. Legacy Class

The third difference on Vector vs ArrayList is that Vector is a legacy class and initially it was not part of the Java Collection Framework. From Java 1.4 Vector was retrofitted to implement List interface and become part of Collection Framework.



## **Q. How to sort ArrayList in Java?**

---

Sorting ArrayList in Ascending Order in Java

First, let's see how to sort an array in ascending order in Java using the `Collections.sort()` method. This method is overloaded, which means you can sort the ArrayList in natural order by leveraging the default comparator, which sorts the list in the natural order, and you can use the `Collections.sort(list, Comparator)` to sort the ArrayList in custom order defined by Comparator. Since we are sorting ArrayList of String which implements a Comparable method, we can use the first method to sort the ArrayList string into the natural order or

ascending order. You can also use this method to sort an ArrayList of Integer into increasing order because that's the default order for an Integer object.

```
import java.util.*;  
  
class Main  
  
{ public static void main(String[] args)  
  
{ List<Integer> listofYears = new ArrayList<Integer>();  
  
    listofYears.add(2021);  
  
    listofYears.add(2019);  
  
    listofYears.add(2018);  
  
    listofYears.add(2020); // print the ArrayList before sorting  
  
    System.out.println("ArrayList before sorting");  
  
    System.out.println(listofYears); // sorting an ArrayList of Integer in ascending  
order  
  
    Collections.sort(listofYears); // print the ArrayList after sorting  
  
    System.out.println("ArrayList after sorting");  
  
    System.out.println(listofYears);  
}  
}
```

## **Q. Difference between ArrayList and HashMap in Java?**

1. The first difference between ArrayList and HashMap is that ArrayList implements a List interface while HashMap implements Map interface in Java.

2. The second difference between ArrayList and HashMap is that ArrayList only stores one object while HashMap stores two objects key and value.
  3. The third difference between HashMap and ArrayList is that keys of HashMap must implement equals and hashCode method correctly, ArrayList doesn't have that requirement but its good to have that because contains() method of ArrayList will use the equals() method to see if that object already exists or not.
  4. The fourth difference between HashMap and ArrayList is that ArrayList maintains the order of objects, in which they are inserted while HashMap doesn't provide any ordering guarantee.
  5. Another difference between ArrayList and HashMap is that ArrayList allows duplicates but HashMap doesn't allow duplicates key though it allows duplicate values.
  6. ArrayList get(index) method always gives an O(1) performance but HashMap get(key) can be O(1) in the best case and O(n) in the worst case.
- 

#### **Q. Java program to get SubList from ArrayList – Example?**

Sometimes we need *subList from ArrayList in Java*. For example, we have an ArrayList of 10 objects and we only need 5 objects or we need an object from index 2 to 6, these are called subList in Java. Java collection API provides a method to *get SubList from ArrayList*. In this Java tutorial, we will see an example of getting SubList from ArrayList in Java. In this program, we have an ArrayList which contains 4 String objects. Later we call ArrayList.subList () method to get part of that List

Example with Source code

```
import java.util.ArrayList;  
  
import java.util.List;  
  
public class GetSubListExample {
```

```
public static void main(String[] args) {  
    ArrayList arrayList = new ArrayList();  
    //Add elements to Arraylist  
    arrayList.add("Java");  
    arrayList.add("C++");  
    arrayList.add("PHP");  
    arrayList.add("Scala");  
    lst = arrayList.subList(1,3);  
    System.out.println("Sub list contains : ");  
    for(int i=0; i< lst.size() ; i++)  
        System.out.println(lst.get(i));  
    //remove one element from sub list  
    Object obj = lst.remove(0);  
    System.out.println(obj + " is removed from sub list");  
    //print original ArrayList  
    System.out.println("After removing " + obj + " from sub list, original ArrayList  
contains : ");  
    for(int i=0; i< arrayList.size() ; i++)  
        System.out.println(arrayList.get(i));  
}  
}
```

## **Q. Array length vs ArrayList Size in Java?**

---

One of the confusing parts in learning Java for a beginner to understand how to find the length of array and ArrayList in Java? The main reason for the confusion is an inconsistent way of calculating the length between two. Calling size() method on arrays and length, or even length() on ArrayList is a common programming error made by beginners. The main reason for the confusion is the special handling of an array in Java. Java native arrays have built-in length attribute but no size() method while the Java library containers, known as Collection classes like ArrayList<>, Vector<>, etc, all have a size() method.

There is one more thing which adds to this confusion, that is capacity, at any point capacity of any collection class is the maximum number of elements collection can hold. the size of collection must be less than or equal to its capacity.

Though in reality, collection resize themselves even before it reaches its capacity, controlled by load factor. I have mentioned this before on my post difference between ArrayList and Array in Java, if you not read it already, you may find some useful detail there as well.

So, use length attribute to get number of elements in a array, also known as length, and for same thing in Collection classes e.g. ArrayList, Vector, use size() method. To give you more context, consider following lines of code, can you spot the error, which is bothering our beginner friend:

```
import java.util.*;

public class ArrayListTest

{ public static void main(String[] args)

{   ArrayList<String> arrList = new ArrayList<String>();

    String[] items = { "One", "Two", "Three", "Four", "Five" };

    for(String str: items){

        arrList.add(str);
```

```
    }

    int size = items.size();

    System.out.println (size);

}

}

Arrays.asList("First", "Second", "Third", "Fourth", "Fifth");
```

It's handy way to declare and initialize ArrayList in same place, similar to array in Java. So next time don't confuse between length and size(), array are special object in Java and has attribute called length, which specifies number of buckets in array, while ArrayList is a Collection class, which inherit size() method, which returns number of elements inside Collection. Remember, size is different than capacity of collection. At any point, size <= capacity of collection.

## **Q. What is CopyOnWriteArrayList in Java ?**

---

CopyOnWriteArrayList is a concurrent Collection class introduced in Java 5 Concurrency API along with its popular cousin ConcurrentHashMap in Java. CopyOnWriteArrayList implements List interface like ArrayList, Vector, and LinkedList but its a thread-safe collection and it achieves its thread-safety in a slightly different way than Vector or other thread-safe collection class. As the name suggests CopyOnWriteArrayList creates a copy of underlying ArrayList with every mutation operation e.g. add, remove, or when you set values. That's why it is only suitable for a small list of values which are read frequently but modified rarely e.g. a list of configurations.

Normally CopyOnWriteArrayList is very expensive because it involves costly Array copy with every writes operation but it's very efficient if you have a List where Iteration outnumbers mutation e.g. you mostly need to iterate the ArrayList and don't modify it too often.

Iterator of CopyOnWriteArrayList is fail-safe and doesn't throw ConcurrentModificationException even if underlying CopyOnWriteArrayList is modified once Iteration begins because

Iterator is operating on a separate copy of ArrayList. Consequently, all the updates made on CopyOnWriteArrayList is not available to Iterator

### **Difference between CopyOnWriteArrayList and ArrayList in Java.**

- 1) First and foremost difference between CopyOnWriteArrayList and ArrayList in Java is that CopyOnWriteArrayList is a thread-safe collection while ArrayList is not thread-safe and cannot be used in the multi-threaded environment.
- 2) The second difference between ArrayList and CopyOnWriteArrayList is that Iterator of ArrayList is fail-fast and throw ConcurrentModificationException once detect any modification in List once iteration begins but Iterator of CopyOnWriteArrayList is fail-safe and doesn't throw ConcurrentModificationException.
- 3) The third difference between CopyOnWriteArrayList vs ArrayList is that Iterator of former doesn't support remove operation while Iterator of later supports remove() operation. If you want to learn more about collections, I suggest you go through Complete Java MasterClass, one of the best Java course on Udemy.

```
import java.util.Iterator;  
  
import java.util.concurrent.CopyOnWriteArrayList;  
  
public class CopyOnWriteArrayListExample  
{  
    public static void main(String args[])  
    {  
        CopyOnWriteArrayList<String> threadSafeList = new  
        CopyOnWriteArrayList<String>();
```

```
threadSafeList.add("Java");
threadSafeList.add("J2EE");
threadSafeList.add("Collection");

Iterator<String> failSafeIterator = threadSafeList.iterator();

while(failSafeIterator.hasNext()){
    System.out.printf("Read from CopyOnWriteArrayList : %s %n",
failSafeIterator.next());

    failSafeIterator.remove(); //not supported in CopyOnWriteArrayList in Java
}

}
```

## **Q. Ways to Remove Elements/Objects From ArrayList in Java ?**

---

There are *two ways to remove objects from ArrayList in Java*, first, by using the `remove()` method, and second by using `Iterator`. `ArrayList` provides overloaded `remove ()` method, one accepts the index of the object to be removed i.e. `remove(int index)`, and the other accept objects to be removed, i.e. `remove (Object obj)`. The rule of thumb is, If you know the index of the object, then use the first method, otherwise use the second method. By the way, you must remember to use `ArrayList` `remove` methods, only when you are not iterating over `ArrayList` if you are iterating then use `Iterator`. `Remove()` method, failing to do so may result in `ConcurrentModificationException` in Java

Another gotcha can have occurred due to autoboxing. If you look closely that two `remove` methods, `remove (int index)` and `remove (Object obj)` are indistinguishable if you are trying to remove from an `ArrayList` of `Integers`.

### **Code Example To Remove Elements from ArrayList**

Let's test the above theory with a simple code example of ArrayList with Integers. The following program has an ArrayList of Integers containing 1, 2, and 3 i.e. [1, 2, 3], which corresponds exactly to the index.

Suppose you have three objects in ArrayList i.e. [1,2,3] and you want to remove the second object, which is 2. You may call remove(2), which is actually a call to remove(Object) if consider autoboxing, but will be interpreted as a call to remove 3rd element, by interpreting as remove(index).

## **Q. How to Create Read Only, Unmodifiable ArrayList in Java?**

---

You can create read-only Collection by using Collections.unmodifiableCollection() utility method. it returns an unmodifiable or read-only view of Collection in which you cannot perform any operation which will change the collection like add(), remove() and set() either directly or while iterating using Iterator or ListIterator. It will throw UnsupportedOperationException whenever you try to modify the List.

One of the common misconceptions around read-only ArrayList is that you can create read-only ArrayList by using Arrays.asList(String[]), which is apparently not true as this method only return a fixed-size list.

You cannot perform add() and remove() but the set() method is still allowed which can change the contents of ArrayList. Collections class also provides a different method to make List and Set read-only. In this Java tutorial, we will learn *How to make any collection read only* and How to create a fixed size List as well.

### **Example**

---

```
import java.util.ArrayList;  
  
import java.util.Arrays;  
  
import java.util.Collection;  
  
import java.util.Collections;  
  
import java.util.List;
```

```
public class ReadOnlyCollection {  
  
    public static void main(String args[])  
    {  
        Collection readOnlyCollection = Collections.unmodifiableCollection(new  
ArrayList<String>());  
  
        ArrayList readableList = new ArrayList();  
  
        readableList.add("Jeffrey Archer");  
  
        readableList.add("Khalid Hussain");  
  
        List unmodifiableList = Collections.unmodifiableList(readableList);  
  
        unmodifiableList.add("R.K. Narayan");  
  
        unmodifiableList.remove(0);  
  
        unmodifiableList.set(0, "Anurag Kashyap");  
  
        fixedLengthList.set(0, "J.K. Rowling");  
  
        System.out.println(fixedLengthList.get(0));  
    }  
}
```

**Q. WAP to create the LinkedList and store the 10 values in it and find the max value from LinkedList without using Collections.max () method**

---

**Example**

---

```
LinkedList lst = new LinkedList ();  
lst.add (100);  
lst.add (200);  
lst.add (50);  
lst.add (30);  
lst.add (24);  
lst.add (56);
```

**Output:** Maximum value is 200

**Q. WAP to create the LinkedList and store the 10 values in it and sort the LinkedList values in Ascending order without using Collections.sort() method and write your own logic for sorting**

---

```
LinkedList lst = new LinkedList();  
lst.add (100);  
lst.add (200);  
lst.add (50);  
lst.add (30);  
lst.add (24);  
lst.add (56);
```

**Output:**

**Before Sorting**

---

```
100  
200  
50  
30  
24  
56
```

## **After Sorting**

---

24

30

50

56

100

200

**Q. WAP to Create the LinkedList and with a Student Details with a following details**

**name id and per and store the 10 student details in LinkedList and find the topper students from collection**

**Note:** In this program we have to Create the POJO class name as Student with id, name and per and store its object in LinkedList.

**Q. WAP to Create the LinkedList and perform the following Operation On It ?**

---

Case 1. Add New Element in LinkedList at beginning

Case 2. Add New Element At the end of LinkedList.

Case 3: Search The Value from LinkedList

Case 4: Delete the element from LinkedList

Case 5: Display the Total size of LinkedList

Case 6: Check LinkedList is empty or not if empty then display message LinkedList empty otherwise show message LinkedList is not empty

Case 7: Travel the LinkedList in forward as well as in backward direction

## **Q. what is the Set and Why Use Set Collection in java?**

---

Basically, **Set** is a type of collection that does not allow duplicate elements. That means an element can only exist once in a Set. It models the set abstraction in mathematics.

### **Characteristics of a Set collection:**

The following characteristics differentiate a Set collection from others in the Java Collections framework:

- Duplicate elements are not allowed.
- Elements are not stored in order. That means you cannot expect elements sorted in any order when iterating over elements of a Set.

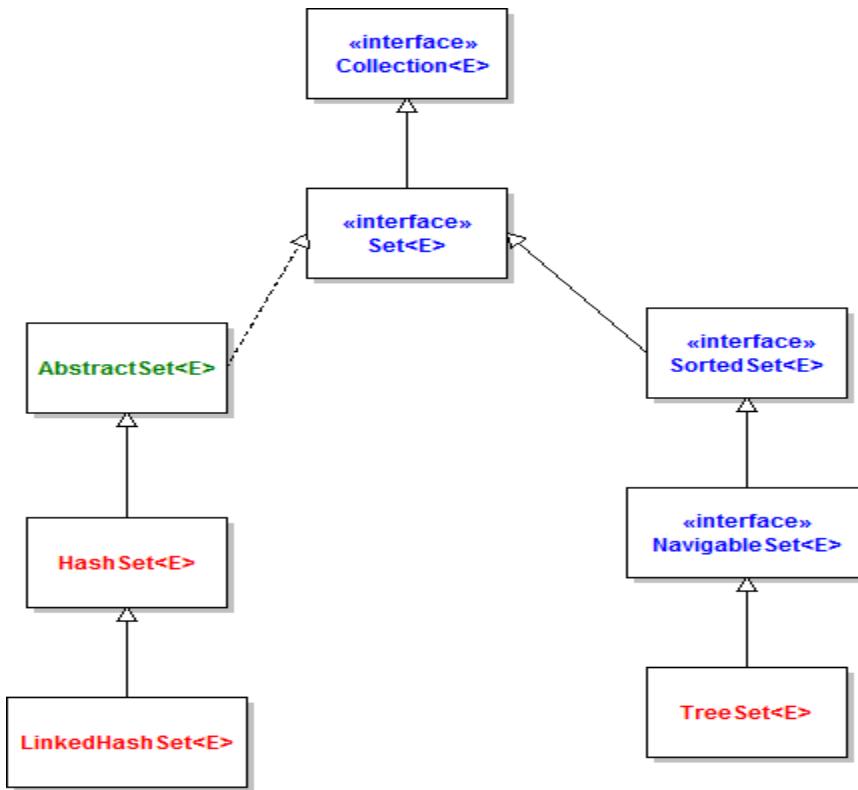
### **Why and When Use Sets?**

Based on the characteristics, consider using a Set collection when:

- You want to store elements distinctly without duplication, or unique elements.
- You don't care about the order of elements.

For example, you can use a Set to store unique integer numbers; you can use a Set to store cards randomly in a card game; you can use a Set to store numbers in random order, etc.

The Java Collections Framework provides three major implementations of the Set interface: HashSet, LinkedHashSet and TreeSet. The Set API is described in the following diagram:



Let's look at the characteristics of each implementation in details:

- **HashSet**: is the best-performing implementation and is a widely-used Set implementation. It represents the core characteristics of sets: no duplication and unordered.
- **LinkedHashSet**: This implementation orders its elements based on insertion order. So consider using a `LinkedHashSet` when you want to store unique elements in order.
- **TreeSet**: This implementation orders its elements based on their values, either by their natural ordering, or by a Comparator provided at creation time.

Therefore, besides the uniqueness of elements that a Set guarantees, consider using `HashSet` when ordering does not matter; using `LinkedHashSet` when you want to order elements by their insertion order; using `TreeSet` when you want to order elements by their values.

## **Q. What is HashSet? What are the properties of HashSet object?**

1. HashSet ensures uniqueness, In other words , every object in the HashSet presents only once
2. Unlike LinkedList, HashSet does not store the objects in orderly manner. It is possible that the object which we added first in the HashSet , may appear last in the output .
3. It offers constant time performance for basic operations like add, remove , contains and size assuming the hash function disperses the element properly among the buckets.
4. HashSet implementation is not synchronized.
  - \* If multiple threads access a hash set concurrently, and at least one of the threads modifies the set, it must be synchronized externally.
  - \*This is typically accomplished by synchronizing on some object that naturally encapsulates the set.

## **Q. What is the default initial capacity and initial load factor of HashSet object?**

As we already discussed that HashSet internally uses HashMap. So the default initial capacity and initial load factor of HashSet is same as that of HashMap, that is

Default Initial Capacity of HashSet Object: 16

Initial Load Factor of HashSet Object: 0.75

Iteration performance of the HashSet object depends on the above two factors that is initial capacity and load factor:

- a. It is very important not to set the initial capacity too high or the load factor too low if iteration performance is important.

## **Q. What is the difference between HashSet and TreeSet?**

**Ordering:** HashSet stores the object in random order . There is no guarantee that the element we inserted first in the HashSet will be printed first in the output .

For example

```
import java.util.HashSet;
public class HashSetExample {
```

```
public static void main(String[] args) {  
    HashSet<String> obj1= new HashSet<String>();  
    obj1.add("Alive");  
    obj1.add("is");  
    obj1.add("Awesome");  
    System.out.println(obj1);  
}  
}
```

Elements are sorted according to the natural ordering of its elements in TreeSet. If the objects can not be sorted in natural order than use compareTo() method to sort the elements of TreeSet object

```
import java.util.TreeSet;  
public class TreeSetExample {  
    public static void main(String[] args) {  
        TreeSet<String> obj1= new TreeSet<String>();  
        obj1.add("Alive");  
        obj1.add("is");  
        obj1.add("Awesome");  
        System.out.println(obj1);  
    }  
}
```

**2. Null value:** HashSet can store null object while TreeSet does not allow null object. If one try to store null object in TreeSet object , it will throw Null Pointer Exception.

**3. Performance :** HashSet take constant time performance for the basic operations like add, remove contains and size. While TreeSet guarantees  $\log(n)$  time cost for the basic operations (add,remove,contains).

**4. Speed:** HashSet is much faster than TreeSet, as performance time of HashSet is constant against the log time of TreeSet for most operations (add,remove ,contains and size) . Iteration performance of HashSet mainly depends on the load factor and initial capacity parameters.

**5. Internal implementation :** As we have already discussed How hashset internally works in java thus, in one line HashSet are internally backed by hashmap. While TreeSet is backed by a Navigable TreeMap.

**6. Functionality :** TreeSet is rich in functionality as compare to HashSet. Functions like pollFirst(),pollLast(),first(),last(),ceiling(),lower() etc. makes TreeSet easier to use than HashSet.

**7. Comparision :** HashSet uses equals() method for comparison in java while TreeSet uses compareTo() method for maintaining ordering .

## **Q. What Are The Similarities Between HashSet and TreeSet**

**1. Unique Elements :** Since HashSet and TreeSet both implements Set interface . Both are allowed to store only unique elements in their objects. Thus there can never be any duplicate elements inside the HashSet and TreeSet objects.

**2. Not Thread Safe :** HashSet and TreeSet both are not synchronized or not thread safe.HashSet and TreeSet, both implementations are not synchronized. If multiple threads access a hash set/ tree set concurrently, and at least one of the threads modifies the set, it must be synchronized externally.

**3. Clone() method copy technique:** Both HashSet and TreeSet uses shallow copy technique to create a clone of their objects .

**4. Fail-fast Iterators :** The iterators returned by this class's method are fail-fast: if the set is modified at any time after the iterator is created, in any way except through the iterator's own remove method, the iterator will throw a ConcurrentModificationException. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

## **Q. When to prefer TreeSet over HashSet**

1. Sorted unique elements are required instead of unique elements. The sorted list given by TreeSet is always in ascending order.

## 2. TreeSet has greater locality than HashSet.

If two entries are nearby in the order , then TreeSet places them near each other in data structure and hence in memory, while HashSet spreads the entries all over memory regardless of the keys they are associated to.

As we know Data reads from the hard drive takes much more latency time than data read from the cache or memory. In case data needs to be read from hard drive than prefer TreeSet as it has greater locality than HashSet.

## 3. TreeSet uses Red- Black tree algorithm underneath to sort out the elements. When one need to perform read/write operations frequently , then TreeSet is a good choice.

### **What is the LinkedHashSet?**

---

LinkedHashSet collection can store the unique data in collection and manage the data sequence as per the user sequence.

The LinkedHashSet class of the Java collections framework provides functionalities of both the hashtable and the linked list data structure.

However, linked hash sets maintain a doubly-linked list internally for all of its elements. The linked list defines the order in which elements are inserted in hash tables.

In order to create a linked hash set, we must import the `java.util.LinkedHashSet` package first.

Once we import the package, here is how we can create linked hash sets in Java.

```
// LinkedHashSet with 8 capacity and 0.75 load factor  
LinkedHashSet numbers = new LinkedHashSet(8, 0.75);
```

Notice, the part `new LinkedHashSet(8, 0.75)`. Here, the first parameter is **capacity** and the second parameter is **loadFactor**.

- **capacity** - The capacity of this hash set is 8. Meaning, it can store 8 elements.

- **loadFactor** - The load factor of this hash set is 0.6. This means, whenever our hash table is filled by 60%, the elements are moved to a new hash table of double the size of the original hash table.

## Q. what is the default capacity and load factor of LinkedHashSet in java Collection?

---

It's possible to create a linked hash set without defining its capacity and load factor. For example,

```
// LinkedHashSet with default capacity and load factor  
LinkedHashSet<Integer> numbers1 = new LinkedHashSet<>();
```

By default,

- the capacity of the linked hash set will be 16
- the load factor will be 0.75

## How to Create the LinkedHashSet by using other collection

---

If we want to create the LinkedHashSet by using other collection framework we have to create the object of collection class and store the element in it and we have to create the object of LinkedHashSet and pass the collection reference in its constructor.

```
import java.util.LinkedHashSet;  
import java.util.ArrayList;  
  
public class Main {  
    public static void main(String[] args) {  
        // Creating an arrayList of even numbers  
        ArrayList<Integer> evenNumbers = new ArrayList<>();  
        evenNumbers.add(2);  
        evenNumbers.add(4);  
        System.out.println("ArrayList: " + evenNumbers);  
        // Creating a LinkedHashSet from an ArrayList
```

```

    LinkedHashSet<Integer> numbers = new
    LinkedHashSet<>(evenNumbers);
    System.out.println("LinkedHashSet: " + numbers);
}
}

```

**Output:**

ArrayList: [2, 4]  
 LinkedHashSet: [2, 4]

## **Q. what is the diff between HashSet and LinkedHashSet in Java?**

---

Internal Working	HashSet internally uses HashMap for storing objects	LinkedHashSet uses LinkedHashMap internally to store objects
When To Use	If you don't want to maintain insertion order but want to store unique objects	If you want to maintain the insertion order of elements then you can use LinkedHashSet
Order	HashSet does not maintain insertion order	LinkedHashSet maintains the insertion order of objects
Complexity of Operations	HashSet gives O(1) complexity for insertion, removing, and retrieving objects	LinkedHashSet gives insertion, removing, and retrieving operations performance in order O(1).
Performance	The performance of HashSet is better when compared to LinkedHashSet and TreeSet.	The performance of LinkedHashSet is slower than TreeSet. It is almost similar to HashSet but slower because LinkedHashSet internally maintains LinkedList to maintain the insertion order of elements

Compare	HashSet uses equals() and hashCode() methods to compare the objects	LinkedHashSet uses equals() and hashCode() methods to compare its objects
Null Elements	HashSet allows only one null value.	LinkedHashSet allows only one null value.

### **Q. what are the similarities between HashSet and LinkedHashSet?**

---

- Duplicates: HashSet, LinkedHashSet and TreeSet are implements Set interface, so they are not allowed to store duplicates objects.
- Thread-safe: If we want to use HashSet, LinkedHashSet, and TreeSet in a multi-threading environment then first we make it externally synchronized because both LinkedHashSet and TreeSet are not thread-safe.
- All three are Cloneable and Serializable

### **Q. When to use HashSet, TreeSet, and LinkedHashSet in Java:**

1. **HashSet:** If you don't want to maintain insertion order but wants to store unique objects.
2. **LinkedHashSet:** If you want to maintain the insertion order of elements then you can use LinkedHashSet.
3. **TreeSet:** If you want to sort the elements according to some Comparator then use TreeSet.

### **Q. Explain the All Set Operations With Examples?**

---

**Union of Sets:** This operation adds all the elements in one set with the other.

To perform the union between two sets, we can use the addAll() method. Now consider we have the two sets and we want to perform the union operation on it.

Let set1 = [1, 3, 2, 4, 8, 9, 0] and set2 = [1, 3, 7, 5, 4, 0, 7, 5]

After union we have the following result

**Union = [0, 1, 2, 3, 4, 5, 7, 8, 9]**

---

### Example

```
import java.util.*;
public class SetExample {
    public static void main(String args[])
    {
        Set<Integer> a = new HashSet<Integer>();
        a.addAll(Arrays.asList( new Integer[] { 1, 3, 2, 4, 8, 9, 0 }));

        // Again declaring object of Set class
        // with reference to HashSet
        Set<Integer> b = new HashSet<Integer>();
        b.addAll (Arrays.asList(new Integer[] { 1, 3, 7, 5, 4, 0, 7, 5 }));
        // To find union
        Set<Integer> union = new HashSet<Integer>(a);
        union.addAll(b);
        System.out.print("Union of the two Set");
        System.out.println(union);
    }
}
```

### Intersection of Sets

This operation returns all the common elements from the given two sets.

For the above two sets, the intersection would be

Let set1 = [1, 3, 2, 4, 8, 9, 0] and set2 = [1, 3, 7, 5, 4, 0, 7, 5]

After Intersection: [0, 1, 3, 4]

---

### Example

```
import java.util.*;
public class SetExample {
    public static void main(String args[])
    { Set<Integer> a = new HashSet<Integer>();
        a.addAll(Arrays.asList( new Integer[] { 1, 3, 2, 4, 8, 9, 0 }));


```

```

// Again declaring object of Set class
// with reference to HashSet
Set<Integer> b = new HashSet<Integer>();
b.addAll(Arrays.asList(new Integer[] { 1, 3, 7, 5, 4, 0, 7, 5 }));
// To find intersection
Set<Integer> intersection = new HashSet<Integer>(a);
intersection.retainAll(b);
System.out.print("Intersection of the two Set");
System.out.println(intersection);
}
}

```

**Difference:** This operation removes all the values present in one set from the other set.

To calculate the difference between the two sets, we can use the removeAll() method.

Let set1 = [1, 3, 2, 4, 8, 9, 0] and set2 = [1, 3, 7, 5, 4, 0, 7, 5]

After Difference: [2, 8, 9]

Example

```

import java.util.*;
public class SetExample {
    public static void main(String args[])
    { Set<Integer> a = new HashSet<Integer>();
        a.addAll(Arrays.asList( new Integer[] { 1, 3, 2, 4, 8, 9, 0 }));
        // Again declaring object of Set class
        // with reference to HashSet
        Set<Integer> b = new HashSet<Integer>();
        b.addAll(Arrays.asList(new Integer[] { 1, 3, 7, 5, 4, 0, 7, 5 }));
        // To find the symmetric difference
        Set<Integer> difference = new HashSet<Integer>(a);
        difference.removeAll(b);
        System.out.print("Difference of the two Set");
        System.out.println(difference);
    }
}

```

## **Q. what is the difference between LinkedHashSet VS HashSet?**

---

### **LinkedHashSet Vs. TreeSet**

Here are the major differences between LinkedHashSet and TreeSet:

- The TreeSet class implements the SortedSet interface. That's why elements in a tree set are sorted. However, the LinkedHashSet class only maintains the insertion order of its elements.
- A TreeSet is usually slower than a LinkedHashSet. It is because whenever an element is added to a TreeSet, it has to perform the sorting operation.
- LinkedHashSet allows the insertion of null values. However, we cannot insert a null value to TreeSet

## **Q. What is TreeSet?**

---

TreeSet is like HashSet which contains the unique elements only but in a sorted manner. The major difference is that TreeSet provides a total ordering of the elements. The elements are ordered using their natural ordering, or by a Comparator typically provided at sorted set creation time. The set's iterator will traverse the set in ascending element order.

## **Q. Why and when we use TreeSet?**

We prefer TreeSet in order to maintain the unique elements in the sorted order .

## **Q. What is natural ordering in TreeSet?**

"Natural" ordering is the ordering implied by the implementation of Comparable interface by the objects in the TreeSet . Essentially RBTree must be able to tell which object is smaller than other object , and there are two ways to supply that logic to the RB Tree implementation :

1. We need to implement the Comparable interface in the class(es) used as objects in TreeSet.
2. Supply an implementation of the Comparator would do comparing outside the class itself.

**Q. Why do we need TreeSet when we already had Sorted Set?**

SortedSet is an interface while TreeSet is the class implementing it. As we know, in java, we can not create the objects of the interface. The class implementing the interface must fulfill the contract of interface, i.e , concrete class must implement all the methods present in the interface. TreeSet is such an implementation.

**Q. What happens if the TreeSet is concurrently modified while iterating the elements?**

The iterator's returned by the TreeSet class iterator method are fail-fast. fail-fast means if the set is modified at any time after the iterator is created , in any way except the iterator's own remove method , the iterator will throw a ConcurrentModificationException. Thus , in the face of concurrent modification , the iterator fails quickly and cleanly

**Q. Which copy technique (deep or shallow ) is used by TreeSet clone() method ?**

According to Oracle docs , clone() method returns the shallow copy of the TreeSet instance. In shallow copy , Both objects A and B shared the same memory location .

**Q. How to convert HashSet to TreeSet object?**

```
Set treeObject = new TreeSet (hashSetObject);
```

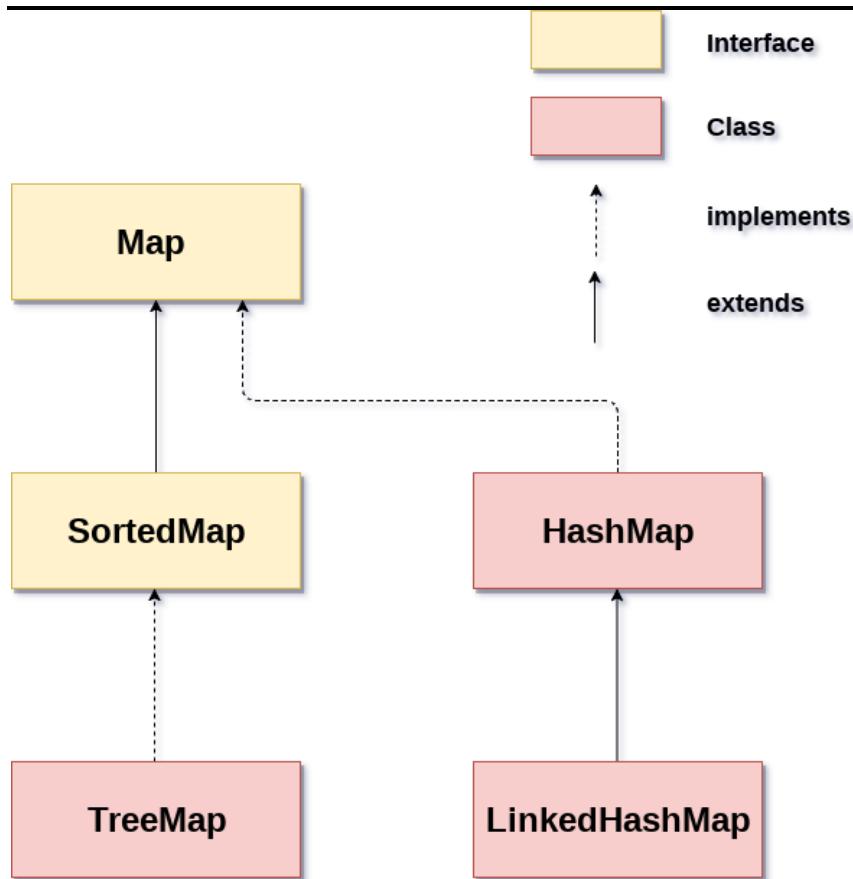
## **Q. What Is the Map interface in java?**

---

Map interface is used for to store the data in the form of key and value pair. Key is unique identity of data and value is actual data present in map means we cannot add the duplicate key in map but we can add the duplicate value in map interface. In short we can say Map is combination of Set and List Collection Set represent key and List represent value.

## **Q. Explain the Map Hierarchy?**

---



In the above Hierarchy we have the Map and SortedMap are the interfaces and HashMap and TreeMap are the implementer classes of above interfaces.

## **Q. Explain the Methods of Map interface**

---

Map interface contain the some methods those help us to store and retrieve, delete, update etc operation on map interface

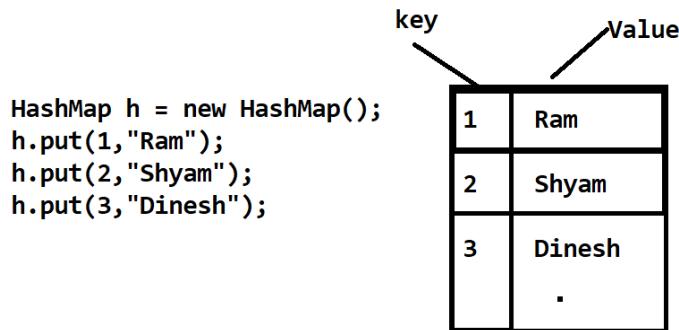
Some of the method given below with its example

---

**void put(Object key, Object value):** this method is used for to store data in map in the form of key and value pair.

### **Example**

---



This method is used for store the data in map using key and value pair shown in diagram

**Object get (Object key):** this method is used for retrieve data from map using its key if key not found return null

**Note:** Normally we use this method for retrieve data using its key as well as this method help us to find or search data from map using its key shown in following example.

### **Example of Fetch Data from Map using its key**

---

key                      Value

1	Ram
2	Shyam
3	Dinesh
.	.

```

HashMap h = new HashMap();
h.put(1,"Ram");
h.put(2,"Shyam");
h.put(3,"Dinesh");

Object obj = h.get(2);
System.out.println(obj); //Shyam

```

Note: in this example we pass the key get() method i.e 2 so we get its value Shyam

### Example of Search Element from Map using get() method

---

key                      Value

1	Ram
2	Shyam
3	Dinesh
.	.

```

Object obj = h.get(2);

if(obj!=null)
{ System.out.println("Element found");
}
else{
    System.out.println("Element not found");
}

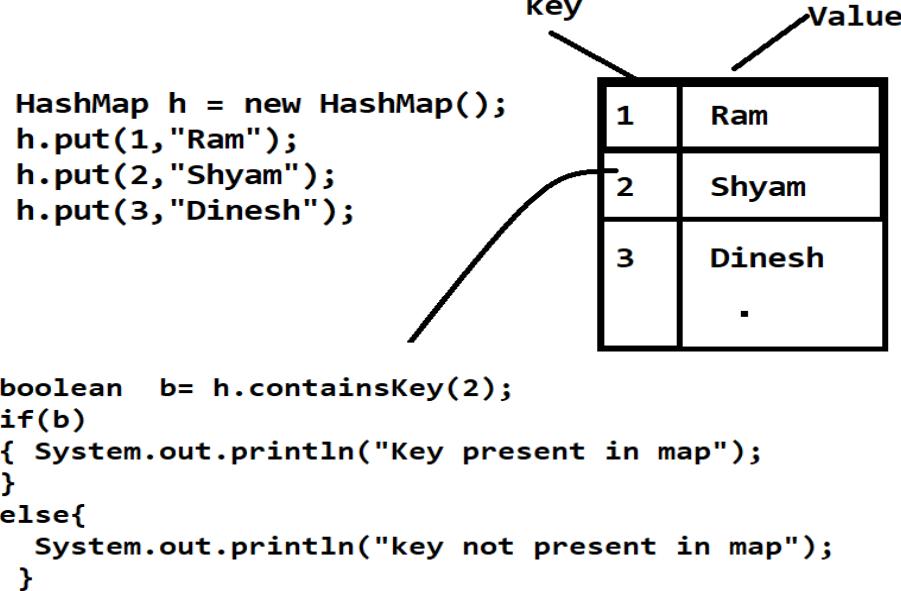
```

Note: get() method return null value if key not present in map and we check the condition obj!=null means if data found

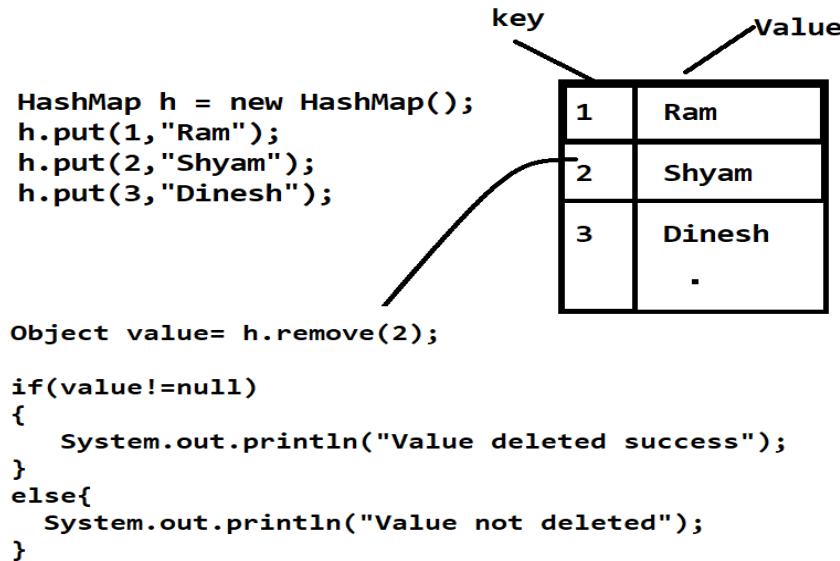
**boolean containsKey(Object key):** this method check key present in map or not if present return true otherwise return false.

### Example

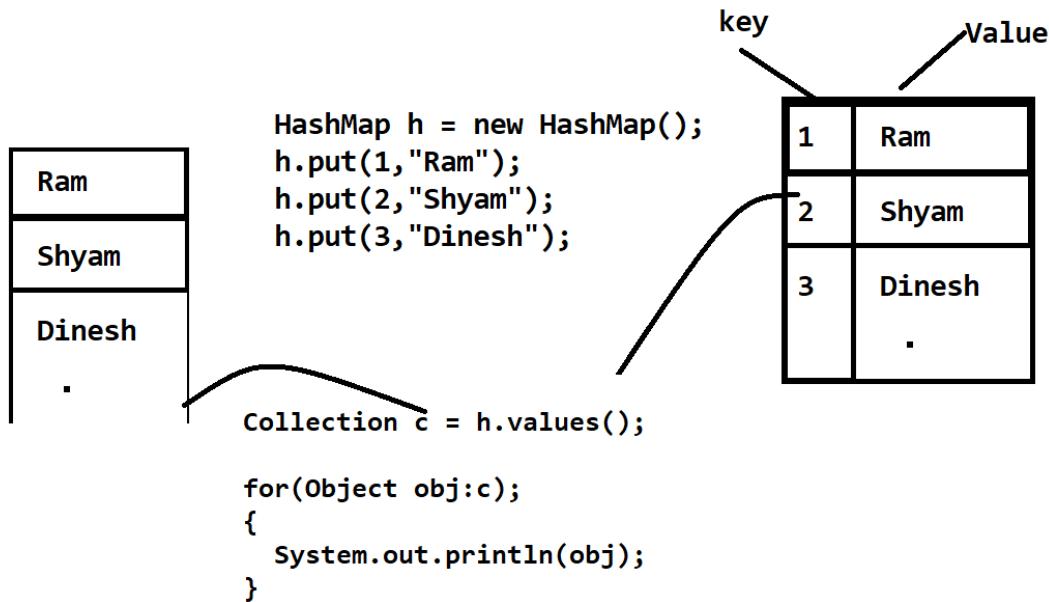
---



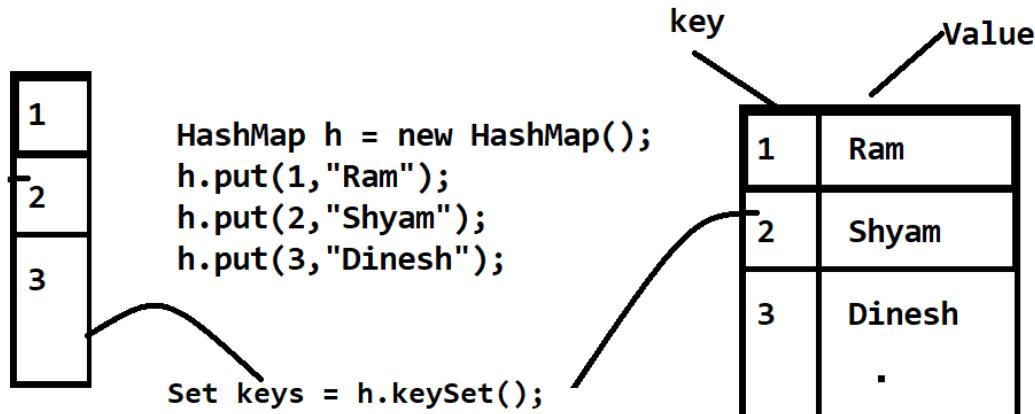
**Object remove (Object key):** this method can delete the data from Map using its key if key found the delete the element and return deleted value if key not found then return null value.



**Collection values():** this method can return only values from Map interface



**Set keyset():** this method return the all keys from Map interface and store in Set Collection.

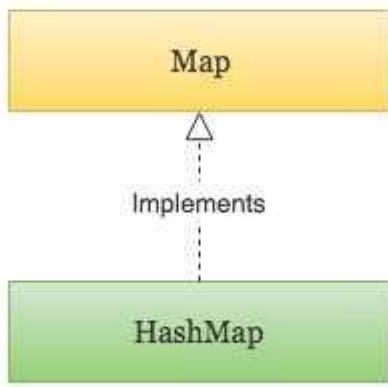


## Q. Explain the HashMap?

---

Java HashMap is a hash table based implementation of Java's Map interface. A Map, as you might know, is a collection of key-value pairs. It maps keys to values.

## Java HashMap Class Hierarchy



Following are few key points to note about HashMap in Java -

- A HashMap cannot contain duplicate keys.
- Java HashMap allows null values and the null key.
- HashMap is an unordered collection. It does not guarantee any specific order of the elements.
- Java HashMap is not thread-safe. You must explicitly synchronize concurrent modifications to the HashMap.

### How to create the HashMap

---

```
import java.util.HashMap;
import java.util.Map;
public class CreateHashMapExample {
    public static void main(String[] args) {
        // Creating a HashMap
        Map numberMapping = new HashMap();
        // Adding key-value pairs to a HashMap
        numberMapping.put("One", 1);
        numberMapping.put("Two", 2);
        numberMapping.put("Three", 3);
```

```

        numberMapping.putIfAbsent("Four", 4);
        System.out.println(numberMapping);
    }
}

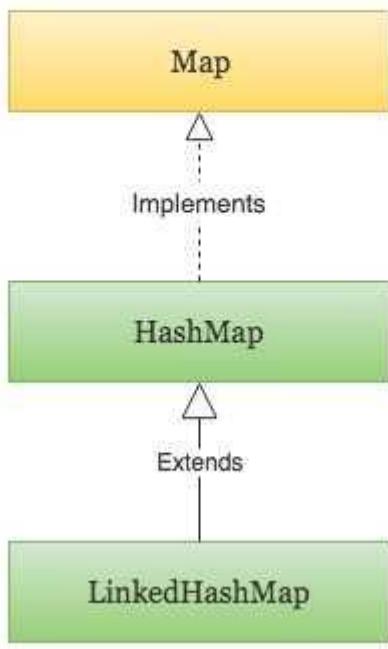
```

## Q. Explain the LinkedHashMap In Java

---

Java LinkedHashMap is a hash table and doubly linked List based implementation of Java's Map interface. It extends the HashMap class which is another very commonly used implementation of the Map interface.

Java LinkedHashMap Class Hierarchy



The iteration order in a LinkedHashMap is normally the order in which the elements are inserted. However, it also provides a special constructor using which you can change the iteration order from the least-recently accessed element to the most-recently accessed element and vice versa. This kind of iteration order can be useful in building LRU caches. In any case, the iteration order is predictable.

**Following are some important points to note about LinkedHashMap in Java -**

- A LinkedHashMap cannot contain duplicate keys.
- LinkedHashMap can have null values and the null key.
- Unlike HashMap, the iteration order of the elements in a LinkedHashMap is predictable.
- Just like HashMap, LinkedHashMap is not thread-safe. You must explicitly synchronize concurrent access to a LinkedHashMap in a multi-threaded environment.

## Creating and Initializing a LinkedHashMap

```
import java.util.LinkedHashMap;
public class CreateLinkedHashMapExample {
    public static void main(String[] args) {
        // Creating a LinkedHashMap
        LinkedHashMap wordNumberMapping = new LinkedHashMap();

        // Adding new key-value pairs to the LinkedHashMap
        wordNumberMapping.put("one", 1);
        wordNumberMapping.put("two", 2);
        wordNumberMapping.put("three", 3);
        wordNumberMapping.put("four", 4);

        wordNumberMapping.putIfAbsent("five", 5);

        System.out.println(wordNumberMapping);
    }
}
```

## **Q. Perform Following Operation on LinkedHashMap**

---

- Check if a key exists in a LinkedHashMap.
- Check if a value exists in the LinkedHashMap.
- Modify the value associated with a given key in the LinkedHashMap.

### **Example**

---

```
import java.util.LinkedHashMap;
public class AccessEntriesFromLinkedHashMapExample {
    public static void main(String[] args) {
        LinkedHashMap<Integer, String> customerIdNameMapping = new
        LinkedHashMap<>();
        customerIdNameMapping.put(1001, "Jack");
        customerIdNameMapping.put(1002, "David");
        customerIdNameMapping.put(1003, "Steve");
        customerIdNameMapping.put(1004, "Alice");
        customerIdNameMapping.put(1005, "Marie");
        System.out.println("customerIdNameMapping : " +
customerIdNameMapping);
        Integer id = 1005;
        if(customerIdNameMapping.containsKey(id)) {
            System.out.println("Found the customer with id " + id + " : " +
customerIdNameMapping.get(id));
        } else {
            System.out.println("Customer with id " + id + " does not exist");
        }
        String name = "David";
        if(customerIdNameMapping.containsValue(name)) {
            System.out.println("A customer named " + name + " exist in the map");
        } else {
            System.out.println("No customer found with name " + name + " in the map");
        }
    }
}
```

```

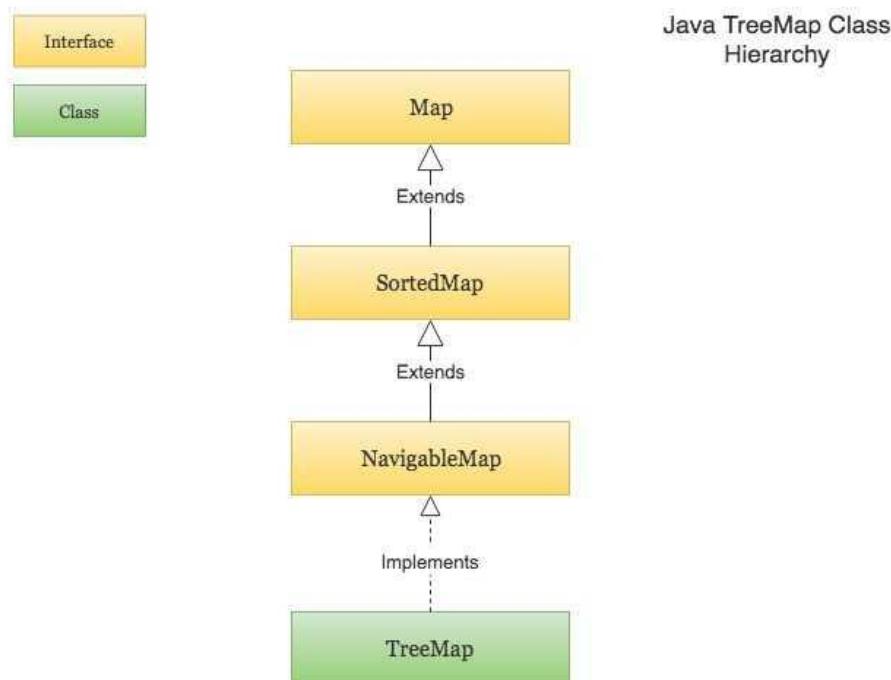
customerIdNameMapping.put(id, "Bob");
System.out.println("Changed the name of customer with id " + id + ", New
mapping : " + customerIdNameMapping);
}
}

```

## **Q. Explain the TreeMap in Detail**

---

Java TreeMap is a Red-Black tree based implementation of Java's Map interface. The entries in a TreeMap are always sorted based on the natural ordering of the keys, or based on a custom Comparator that you can provide at the time of creation of the TreeMap. The TreeMap class is part of Java's collection framework. It implements the NavigableMap interface, which in turn extends the SortedMap interface. Following is the class hierarchy of TreeMap.



The SortedMap interface provides functionalities to maintain the ordering of keys. And the NavigableMap interface provides functionalities to navigate through the

map. For example, finding the entry just greater than or just less than the given key, finding the first and last entry in the TreeMap etc.

Since a TreeMap implements NavigableMap interface, it has the functionalities of both the NavigableMap as well as the SortedMap.

### **Following are few key points to note about TreeMap in Java -**

- A TreeMap is always sorted based on keys. The sorting order follows the natural ordering of keys. You may also provide a custom Comparator to the TreeMap at the time of creation to let it sort the keys using the supplied Comparator.
- A TreeMap cannot contain duplicate keys.
- TreeMap cannot contain the null key. However, It can have null values.
- TreeMap is not synchronized. Access to TreeMaps must be synchronized explicitly in a multi-threaded environment.

## **Creating a TreeMap**

---

```
import java.util.SortedMap;
import java.util.TreeMap;
public class CreateTreeMapExample {
    public static void main(String[] args) {
        // Creating a TreeMap
        SortedMap fileExtensions = new TreeMap();
        // Adding new key-value pairs to a TreeMap
        fileExtensions.put("python", ".py");
        fileExtensions.put("c++", ".cpp");
        fileExtensions.put("kotlin", ".kt");
        fileExtensions.put("golang", ".go");
        fileExtensions.put("java", ".java");

        // Printing the TreeMap (Output will be sorted based on keys)
        System.out.println(fileExtensions);
```

## Package

---

### **Q What is the Package?**

---

Package is collection of classes and interfaces in java

### **Q Why Use Package or what is the benefit of package?**

---

The major use of package is reuse the bundle of classes as well as it is used for create the distribution or reusable component using java.

### **Q how to create the package and Reuse the Package using java?**

---

If we want to create the package in java we have the some important steps.

**i) Declare the package:** if we want to declare the package in java we have the package keyword.

**syntax:**

```
package packagename;
```

**e.g package org.techhub;**

### **ii) Declare the classes under the package:**

---

Once we declare the package we can declare the number of classes in package as well as declare the interface as member under the package.

**syntax:**

```
package packagename;  
access specifier class classname  
{  
    access specifier returntype functionname(datatype variablename)  
    {  
    }  
}  
}
```

```
package org.techhub;  
public class Table  
{  
    public void showTable(int x)  
    {  
        for(int i=1; i<=10; i++)  
        {  
            System.out.printf("%d X %d = %d\n",i,x,i*x);  
        }  
    }  
}
```

Once we create package and declare the classes in package then we can compile the package

If we want to compile class in which we declare the package we have the command like as

## **javac -d . filename.java**

-d means create the folder according to package name mention in file and store the all .class files in it.

**Note: when we want to compile your package or java program from any another location except the jdk\bin then we need to set the jdk\bin folder path in my computer enviourmental variable.**

### **How to set the jdk\bin folder in My Computer Enviourmental variable**

---

**If we want to set the path of jdk\bin folder in mycomputer enviourmental variable we have the some steps.**

Right click on my computer → click on advanced system setting → choose advance option → click on enviourmental variable → system variable → select path variable → click on edit button → click on new button → paste the jdk\bin folde where it is present.

Once we set the jdk\bin folder path then we can compile the java program from any location in your machine

**3)Compile the package program:** if we want to compile the package program then we have the command given below

**javac –d . filename.java**

e.g javac -d . Table.java

as per our example when we execute this command then java automatically create the package name as org.techhub and store them all .class file in it.

Once we create the package then we can reuse the package more than one time.

## **How to Reuse the Package**

---

If we want to reuse the package in java we have the some important steps.

i) **import the created package:** if we want to import package we have the import keyword.

### **syntax**

---

**import packagename;**

**e.g import org.techhub;**

ii) **Create the object of class which we want use:**

### **Example**

---

```
import org.techhub.Table;
public class UseTableApplication
{
    public static void main(String x[])
    {
        Table t = new Table();
    }
}
```

**iii) Call the member from package which we want to use**

---

```
import org.techhub.Table;
public class UseTableApplication
{
    public static void main(String x[])
    {
        Table t = new Table();

        t.showTable(5);

    }
}
```

```
D:\batchdevyani>javac TableApplication.java

D:\batchdevyani>java TableApplication
1 X 5 = 5
2 X 5 = 10
3 X 5 = 15
4 X 5 = 20
5 X 5 = 25
6 X 5 = 30
7 X 5 = 35
8 X 5 = 40
9 X 5 = 45
10 X 5 = 50
```

## Rules of Access Specifier With package

---

**public:** if we declare the member as public in package then member can access inside of package as well as outside of package.

**default:** if we not specify any access specifier with member then java use the default access specifier and if declare any member as default then member can access only side of package not outside of package.

**private :** if we declare any member as private then member cannot access outside class as well as outside of package also.

**protected** : protected member can access within package as well as outside of package but only within child class.

## Example

---

```
package org.techhub;
public class Table
{
    protected void showTable(int x)
    {
        for(int i=1; i<=10; i++)
        {
            System.out.printf("%d X %d = %d\n",i,x,i*x);
        }
    }
}
```

```
import org.techhub.Table;
class MyTable extends Table
{
    void display()
    {
        showTable(5);
    }
}
public class TableApplication
{
    public static void main(String x[])
    {
        MyTable m = new MyTable();
        m.display();
    }
}
```

## Package

---

### **Q What is the Package?**

---

Package is collection of classes and interfaces in java

### **Q Why Use Package or what is the benefit of package?**

---

The major use of package is reuse the bundle of classes as well as it is used for create the distribution or reusable component using java.

### **Q how to create the package and Reuse the Package using java?**

---

If we want to create the package in java we have the some important steps.

**i) Declare the package:** if we want to declare the package in java we have the package keyword.

**syntax:**

```
package packagename;
```

**e.g package org.techhub;**

### **ii) Declare the classes under the package:**

---

Once we declare the package we can declare the number of classes in package as well as declare the interface as member under the package.

**syntax:**

```
package packagename;  
access specifier class classname  
{  
    access specifier returntype functionname(datatype variablename)  
    {  
    }  
}  
}
```

```
package org.techhub;  
public class Table  
{  
    public void showTable(int x)  
    {  
        for(int i=1; i<=10; i++)  
        {  
            System.out.printf("%d X %d = %d\n",i,x,i*x);  
        }  
    }  
}
```

Once we create package and declare the classes in package then we can compile the package

If we want to compile class in which we declare the package we have the command like as

## **javac -d . filename.java**

-d means create the folder according to package name mention in file and store the all .class files in it.

**Note: when we want to compile your package or java program from any another location except the jdk\bin then we need to set the jdk\bin folder path in my computer enviourmental variable.**

### **How to set the jdk\bin folder in My Computer Enviourmental variable**

---

**If we want to set the path of jdk\bin folder in mycomputer enviourmental variable we have the some steps.**

Right click on my computer → click on advanced system setting → choose advance option → click on enviourmental variable → system variable → select path variable → click on edit button → click on new button → paste the jdk\bin folde where it is present.

Once we set the jdk\bin folder path then we can compile the java program from any location in your machine

**3)Compile the package program:** if we want to compile the package program then we have the command given below

**javac -d . filename.java**

e.g javac -d . Table.java

as per our example when we execute this command then java automatically create the package name as org.techhub and store them all .class file in it.

Once we create the package then we can reuse the package more than one time.

## **How to Reuse the Package**

---

If we want to reuse the package in java we have the some important steps.

i) **import the created package:** if we want to import package we have the import keyword.

### **syntax**

---

**import packagename;**

**e.g import org.techhub;**

ii) **Create the object of class which we want use:**

### **Example**

---

```
import org.techhub.Table;
public class UseTableApplication
{
    public static void main(String x[])
    {
        Table t = new Table();
    }
}
```

**iii) Call the member from package which we want to use**

---

```
import org.techhub.Table;
public class UseTableApplication
{
    public static void main(String x[])
    {
        Table t = new Table();

        t.showTable(5);

    }
}
```

```
D:\batchdevyani>javac TableApplication.java  
D:\batchdevyani>java TableApplication  
1 X 5 = 5  
2 X 5 = 10  
3 X 5 = 15  
4 X 5 = 20  
5 X 5 = 25  
6 X 5 = 30  
7 X 5 = 35  
8 X 5 = 40  
9 X 5 = 45  
10 X 5 = 50
```

## Rules of Access Specifier With package

---

**public:** if we declare the member as public in package then member can access inside of package as well as outside of package.

**default:** if we not specify any access specifier with member then java use the default access specifier and if declare any member as default then member can access only side of package not outside of package.

**private :** if we declare any member as private then member cannot access outside class as well as outside of package also.

**protected** : protected member can access within package as well as outside of package but only within child class.

## Example

---

```
package org.techhub;
public class Table
{
    protected void showTable(int x)
    {
        for(int i=1; i<=10; i++)
        {
            System.out.printf("%d X %d = %d\n",i,x,i*x);
        }
    }
}
```

```
import org.techhub.Table;
class MyTable extends Table
{
    void display()
    {
        showTable(5);
    }
}
public class TableApplication
{
    public static void main(String x[])
    {
        MyTable m = new MyTable();
        m.display();
    }
}
```

## Exception Handling In Java

### Q. What Is The Exception?

Exception is events which occur at program run time and which is responsible for disturb the normal flow of application called as Exception.

```
import java.util.*;
public class DivApplication
{
    public static void main(String x[])
    {
        Scanner xyz =new Scanner(System.in);
        int a,b,c;
        System.out.println("Enter the two values");
        a=xyz.nextInt(); //9
        b=xyz.nextInt(); //0
        c=a/b;
        System.out.print("Division is %d\n",c);
        System.out.println("Logic1");
        System.out.println("Logic2");
        System.out.println("Logic3");
    }
}
```

The diagram illustrates a division operation where the numerator is 9 and the denominator is 0. A curved arrow points from the text '9/0' to a circle containing three small circles, representing infinity. Another curved arrow points from the word 'infinity' to the same circle. A line also connects the variable 'c' in the code to this infinity symbol, indicating that the division results in infinity.

In above code if input the value a is 9 and value of b is 0 and perform the operation  $c=a/b$  means  $c=9/0$  it is called as infinity. So JVM cannot perform the calculation with infinity so it will raise error at program run time and break the execution of complete program called as exception.

### Q. Why use the Exception Handling?

- 1) Exception handling help program to detect the problem at run time
- 2) It Help us to skip the code which is responsible for generate the exception and execute the remaining program safe zone.

## There Are Three Types of Exception in Java

---

**1) Checked Exception:** Those exceptions occur at program compile time called as exception. Normally in Java checked Exception is used for to generate the exception warning at program compile time.

```
import java.io.*;
public class DivApplication
{
    public static void main(String x[])
    {
        DataInputStream xyz= new DataInputStream(System.in);
        int a,b,c;
        System.out.println("Enter the two values");
        a=Integer.parseInt(xyz.readLine());
        b=Integer.parseInt(xyz.readLine());
        c=a/b;
        System.out.printf("Division is %d\n",c);
        System.out.println("Logic1");
        System.out.println("Logic2");
        System.out.println("Logic3");
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac DivApplication.java
DivApplication.java:9: error: unreported exception IOException; must be caught or declared to be thrown
        a=Integer.parseInt(xyz.readLine());
                           ^
DivApplication.java:10: error: unreported exception IOException; must be caught or declared to be thrown
        b=Integer.parseInt(xyz.readLine());
                           ^
Note: DivApplication.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
2 errors
```

If we think about the above code then we get the exception error at program compile time it will generate the error to us at program compile time IOException must be caught or declared to be thrown

It Will Generate the Warning Message to Us at Program Compile time for handle the IOException in program.

**2) Unchecked Exception:** That Exception Occur at program run time called as Unchecked Error

```
import java.util.*;
public class DivApplication
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in);
        int a,b,c;
        System.out.println("Enter the two values");
        a=xyz.nextInt();
        b=xyz.nextInt();
        c=a/b;
        System.out.printf("Division is %d\n",c);
        System.out.println("Logic1");
        System.out.println("Logic2");
        System.out.println("Logic3");
    }
}
```

## Output

---

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac DivApplication.java

C:\Program Files\Java\jdk1.8.0_291\bin>java DivApplication
Enter the two values
8
0
Exception in thread "main" java.lang.ArithmetricException: / by zero
        at DivApplication.main(DivApplication.java:11)

C:\Program Files\Java\jdk1.8.0_291\bin>
```

If we think about above code then we compile program successfully but we get the exception at program run time when we input the values 8 and 0

**3) Error:** Error means those exceptions not handle by programmer called as error.

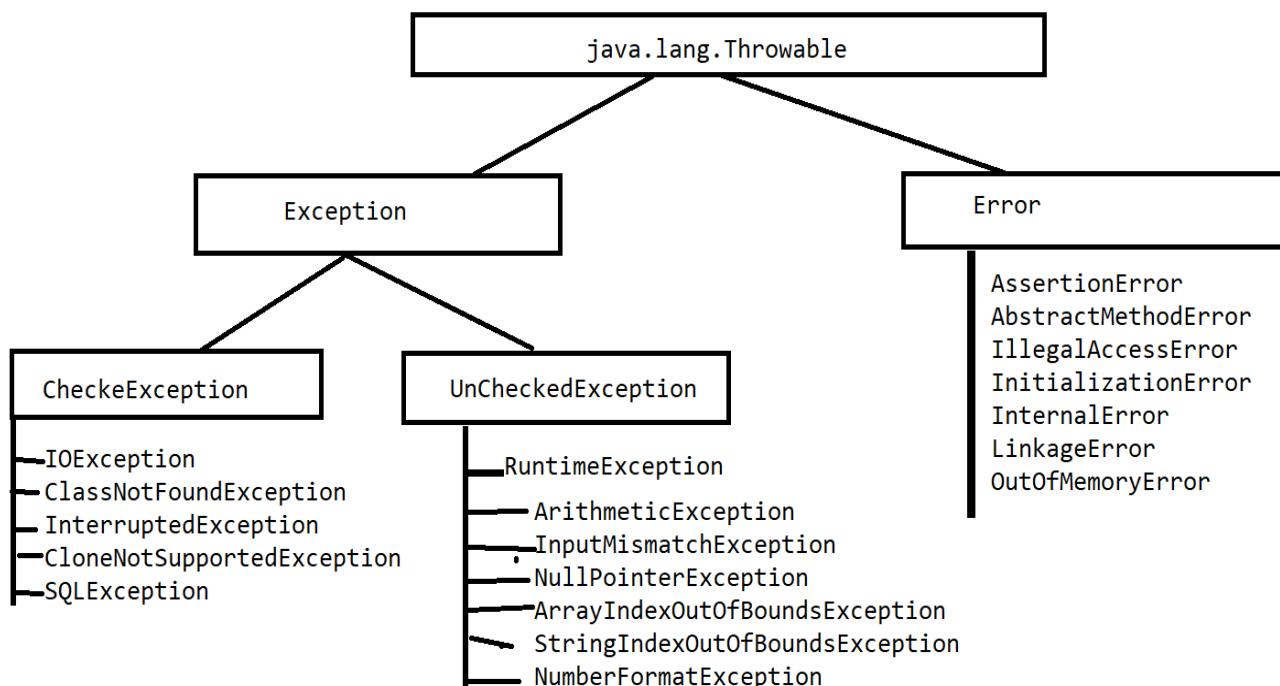
### **Q. what is the diff between Exception and Error?**

---

Exception may be checked and unchecked and handle by programmer but Error is part of Exception but never handle by programmer

## Hierarchy of Exception Handling

---



## Keywords for Exception Handling

---

If we want to work with exception in java we have the major five keywords provided by java to us.

**try :** try is block in java which is used for write the code in which exception may be occur or normally we write code in try block which is responsible for generate the exception in program. When exception generate in try block then JVM create the one error object and hand over to catch for further execution.

**catch:** catch is block which always execute when exception generate in try block Normally Catch block is used for execute the logic those want to execute after exception or write the logic those want to execute after exception as well as we can show the exception message using

catch block. Catch block hold the error object thrown by try block and display it.

syntax:

```
try
{
    write here logics which is responsible for exception
}
catch(exceptiontype ref)
{
    write here logic those want to execute after exception
}
```

**Note: single try can have more than one catch block.**

syntax:

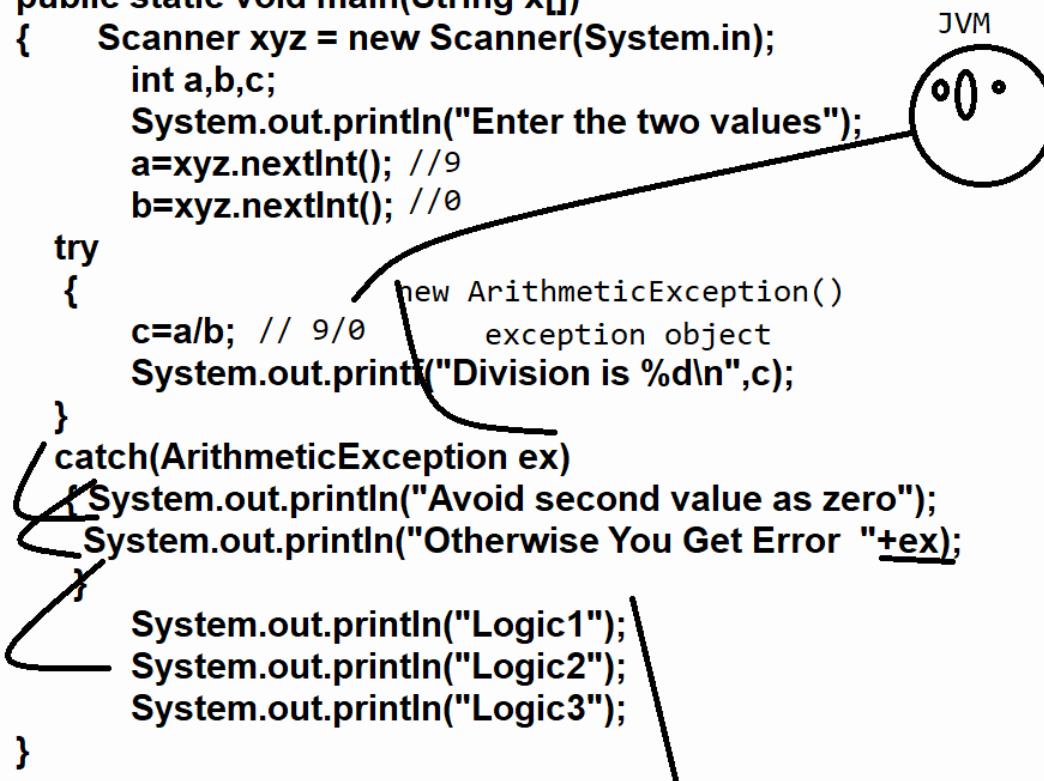
---

```
try
{
    write here code
}
catch(Exceptiontype1 ref)
{
}
catch(Exceptiontype2 ref)
{
}
```

## Example

---

```
import java.util.*;
public class DivApplication
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in);
        int a,b,c;
        System.out.println("Enter the two values");
        a=xyz.nextInt(); //9
        b=xyz.nextInt(); //0
        try
        {
            c=a/b; // 9/0
            System.out.printf("Division is %d\n",c);
        }
        catch(ArithmaticException ex)
        {
            System.out.println("Avoid second value as zero");
            System.out.println("Otherwise You Get Error "+ex);
        }
        System.out.println("Logic1");
        System.out.println("Logic2");
        System.out.println("Logic3");
    }
}
```



## Output

```
C:\Program Files\Java\jdk1.8.0_291\bin>java DivApplication
Enter the two values
9
0
Avoid second value as zero
Otherwise You Get Error  java.lang.ArithmaticException: / by zero
Logic1
Logic2
Logic3
```

**Single Try can have multiple catch blocks.**

```
import java.util.*;
```

```
public class DivApplication
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in);
        int a,b,c;
        try
        {
            System.out.println("Enter the two values");
            a=xyz.nextInt();
            b=xyz.nextInt();
            c=a/b;
            System.out.printf("Division is %d\n",c);
        }
        catch(ArithmeticException ex)
        {
            System.out.println("Avoid second value as zero");
            System.out.println("Otherwise You Get Error "+ex);
        }
        catch(InputMismatchException ex)
        {
            System.out.println("Error is "+ex);
        }
        System.out.println("Logic1");
        System.out.println("Logic2");
        System.out.println("Logic3");
    }
}
```

## **Output:**

---

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac DivApplication.java  
C:\Program Files\Java\jdk1.8.0_291\bin>java DivApplication  
Enter the two values  
6.5  
Error is java.util.InputMismatchException  
Logic1  
Logic2  
Logic3
```

If we write only Exception class in catch block then we can manage the any kind of Exception using single catch block.

```
import java.util.*;
public class DivApplication
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in);
        int a,b,c;
        try
        {
            System.out.println("Enter the two values");
            a=xyz.nextInt();
            b=xyz.nextInt();

            c=a/b;
            System.out.printf("Division is %d\n",c);
        }
        catch(Exception ex)
        {
            System.out.println("Avoid second value as zero");
            System.out.println("Otherwise You Get Error "+ex);
        }
        System.out.println("Logic1");
        System.out.println("Logic2");
        System.out.println("Logic3");
    }
}
```

## Output

---

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac DivApplication.java

C:\Program Files\Java\jdk1.8.0_291\bin>java DivApplication
Enter the two values
5
0
Avoid second value as zero
Otherwise You Get Error  java.lang.ArithmetricException: / by zero
Logic1
Logic2
Logic3

C:\Program Files\Java\jdk1.8.0_291\bin>java DivApplication
Enter the two values
5.4
Avoid second value as zero
Otherwise You Get Error  java.util.InputMismatchException
Logic1
Logic2
Logic3
```

## Example for ArrayIndexOutOfBoundsException

ArrayIndexOutOfBoundsException occur when we try to store value more than index of array.

```
import java.util.*;
public class ArrayIndexTestApp
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in);
        int a[]={10,20,30};
        System.out.println("Value is "+a[3]);
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac ArrayIndexTestApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java ArrayIndexTestApp
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
        at ArrayIndexTestApp.main(ArrayIndexTestApp.java:9)

C:\Program Files\Java\jdk1.8.0_291\bin>
```

If we think about above code we have array with 3 elements means in array contain 0 1 and 2 index but we try to fetch element from 3<sup>rd</sup> index of array but array having last index value is 2 so JVM not found the 3<sup>rd</sup> index array so we get error `ArrayIndexOutOfBoundsException`

If we want to handle this problem we need to handle the `ArrayIndexOutOfBoundsException` in Program.

Following Example show How Handle the `ArrayIndexOutOfBoundsException`

---

```
import java.util.*;
public class ArrayIndexTestApp
{
    public static void main(String x[])
    {
        Scanner xyz = new Scanner(System.in);
        try
        {
            int a[]={10,20,30};
            System.out.println("Value is "+a[3]);
        }
        catch(ArrayIndexOutOfBoundsException ex)
        { System.out.println("Array Limit Exceed");
        }
    }
}
```

**Output :**

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac ArrayIndexTestApp.java
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>java ArrayIndexTestApp  
Array Limit Exceed
```

finally

throw

throws

## **Finally keyword**

---

**finally** : finally is keyword in exception handling which always execute if exception generate program or not. Normally finally block is used for resource cleaning purpose like as for database connection close,file close etc

---

```
try
{
    write here exception logic
}
finally
{
    write here logic those want to execute always
}
```

Or we can write try catch and finally at time.

---

```
try
{
    write here exception logic
}
catch(Exceptiontype ref)
{
    write here your logics
}
finally
{
    write here logic those want to execute always
}
```

**Q. can you write try without catch?**

---

Yes we can write the try without catch by using finally.

```
import java.util.*;
public class TestFinallyApp
{
    public static void main(String x[])
    {
        int a,b,c;
        Scanner xyz = new Scanner(System.in);
    try
    {
        System.out.println("Enter the two values");
        a=xyz.nextInt();
        b=xyz.nextInt();
        c=a/b;
        System.out.printf("Division is %d\n",c);
    }
    finally
    {
        System.out.println("I can execute always");
    }
    System.out.println("Logic1");
    System.out.println("Logic2");
    System.out.println("Logic3");
}
}
```

## Output

---

```
Enter the two values
8
2
Division is 4
I can execute always
Logic1
Logic2
Logic3

C:\Program Files\Java\jdk1.8.0_291\bin>java TestFinallyApp
Enter the two values
9
0
I can execute always
Exception in thread "main" java.lang.ArithmetricException: / by zero
at TestFinallyApp.main(TestFinallyApp.java:13)

C:\Program Files\Java\jdk1.8.0_291\bin>
```

**Note:** finally cannot handle the exception finally only execute its own block in any situation. If we want to handle the exception in java we have to write catch block.

### **Q. what is the diff between catch and finally ?**

Catch is responsible to handle the exception and manage the other program but finally cannot handle the exception just execute its own block and if exception generate in program then finally get executed before exception .

### **Source Code**

---

```
import java.util.*;
public class TestFinallyApp
{
    public static void main(String x[])
    {
        int a,b,c;
        Scanner xyz = new Scanner(System.in);
        try
```

```
{  
    System.out.println("Enter the two values");  
    a=xyz.nextInt();  
    b=xyz.nextInt();  
    c=a/b;  
    System.out.printf("Division is %d\n",c);  
}  
catch(Exception ex)  
{ System.out.println("Error is "+ex);  
}  
finally  
{ System.out.println("I can execute always");  
}  
System.out.println("Logic1");  
System.out.println("Logic2");  
System.out.println("Logic3");  
}  
}
```

## **Output**

---

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac TestFinallyApp.java  
C:\Program Files\Java\jdk1.8.0_291\bin>java TestFinallyApp  
Enter the two values  
9  
0  
Error is java.lang.ArithmetricException: / by zero  
I can execute always  
Logic1  
Logic2  
Logic3  
C:\Program Files\Java\jdk1.8.0_291\bin>
```

## **Q. What is the diff between final, finally and finalize ?**

---

Final is keyword we can use with variable, function and class

Final keyword work with variable for constant declaration purpose, for method avoids the method overriding and for class for avoids the inheritance.

Finally is block use with try block in exception handling situation for execute the code any time and finalize is method of object class which is used perform garbage collection.

## **Throws keyword**

---

Throws is keyword in exception handling we can use with function.

### **Important points Related with throws keyword**

---

- 1) Throws keyword is used for handle the checked exception
- 2) When we use throws keyword then we not need to write try and catch block in function definition we need to write try and catch block at the time of function.

3) When exception generate in function definition it will throw on function calling.

### Syntax of throws keyword

---

```
returntype functionname(arguments) throws exceptionclassname  
{  
    write here logic in function  
}
```

**Note:** Normally throws use with function to generate the compile warning related with exception.

Suppose consider we write function and it contain some logic and logic may be generate the exception run time and our function use by some another person then we need to provide the warning to related exception to program at compile time to avoid the run exception in this case we can use the throws with function and handle the checked exception

Means when someone calls our function then compiler generate the compile time warning related with exception.

```
class Div
{
    void setValue(int x,int y) throws Exception
    {
        int z = x/y; // 8/0
        System.out.println("Division is "+z);
    }
}
public class DivApplication
{
    public static void main(String x[])
    {
        try
        {
            Div d = new Div();
            d.setValue(8,0);
        }
        catch(Exception ex)
        {
            System.out.println("Error is "+ex);
        }
    }
}
```

## Output

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac DivApplication.java

C:\Program Files\Java\jdk1.8.0_291\bin>java DivApplication
Error is java.lang.ArithmeticException: / by zero

C:\Program Files\Java\jdk1.8.0_291\bin>
```

## **throw keyword**

throw is keyword or clause which is specially design for handle the user defined exception.

## **Q. what is the user defined exception?**

User define exception means those exception defined by user for its own use called as user defined exception.

## **Q. Why we need to use user defined exception?**

---

- 1)** If programmer want to create the customize error message and exception according to his application then he can write the user defined exception.
- 2)** If programmer having logical error at run time and java API not provide the appropriate classes to handle that logical error then user can create own exception class and handle the error then he need to create user defined exception

## **Q. How To Create User Defined Exceptions?**

---

If we want to create the user defined exceptions in java we have the some important steps.

### **Step1**

---

**Create the class and inherit the any exception class in it.**

```
class VoterException extends ArithmeticException  
{  
}
```

### **Step2:**

---

**Define the function under the class and write your own logic**

```
class VoterException extends ArithmeticException  
{    int age;  
        String getVoterAge(int age)  
        { this.age=age;  
        }
```

```
String errorMessage()
{    if(age<18)
    { return "you are not eligible for voting";
    }
    if(age<0)
    { return " Invalid Age ";
    }
}
```

### **Step3**

---

If we want to handle the user defined exception then we need to create the object of user defined exception class and throw it and pass the reference of user exception class in catch block where we call it.

## Wrapper classes

Wrapper classes is used for perform the conversion between or type casting between primitive data type to object or reference data type and reference data type to primitive data type.

### Before Learn the Wrapper classes we need to what is the Type casting or Type Conversion

Type casting means if we change the one type of value in to the type for single line of code called as type casting.

Basically there are two types of casting in java

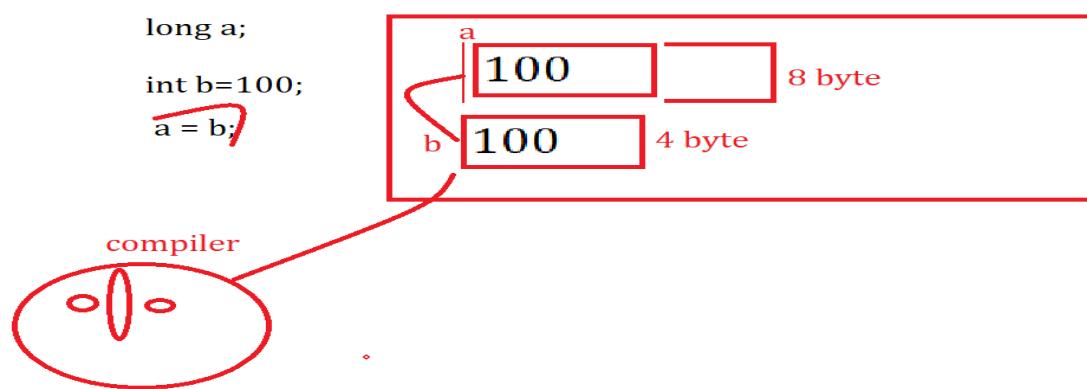
**1) Primitive conversion:** primitive conversion means if we perform casting between primitive data type called as primitive conversion.

**Example:** convert int to float or float to int

### There Are Two Types of Primitive Conversion

**a) Implicit conversion:** implicit conversion means when we put the larger value at left hand side and smaller type value at right hand side then compiler is able to perform conversion automatically called as implicit conversion.

#### Example



### Program for Implicit Conversion

---

## Example

---

```
package org.techhub;
public class TestConversionApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        long a;
        int b = 100;
        a = b; // implicit conversion
        System.out.println("A is " + a);
    }
}
```

**b) Explicit conversion** : if we put the smaller type value at left hand side and larger type value at right hand side then compiler is unable to perform conversion automatically then programmer need to perform conversion manually called as explicit conversion.

---

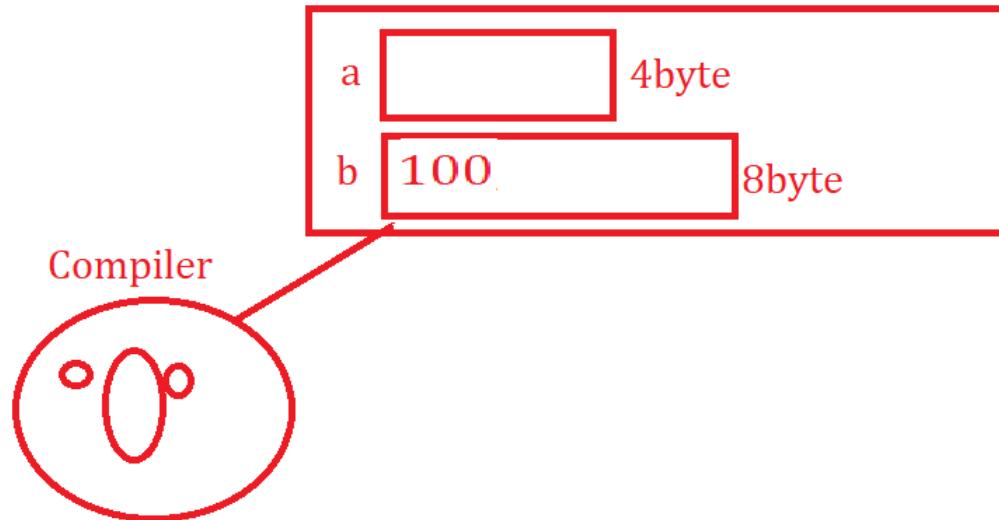
## Example

---

int a;

long b=100;

a=b; //here compiler will generate the error



So if we want to solve the error mention in above diagram we have to perform conversion manually called as explicit conversion shown in following diagram.

int a;

long b=100;

a=(int)b; // explicit conversion

System.out.printf("A is %d\n",a);

## Source Code example

```
package org.techhub;
public class TestConversionApp {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        long a;
        int b = 100;
        a = b; // implicit conversion
        System.out.println("A is " + a);

        int c;
        long d=1000;
        c=(int)d; // explicit conversion
        System.out.printf("C is %d",c);
    }
}
```

Implicit conversion and explicit conversion get failed if we try to convert the primitive type value to object type and object type value to primitive type

### Example

---

```

public class TestConversionApp {

    public static void main(String[] args) {
        String str="12345";   1 it will generate the error to us
        int a=(int)str;-----2
        System.out.printf("A is %d\n",a);
    }
}

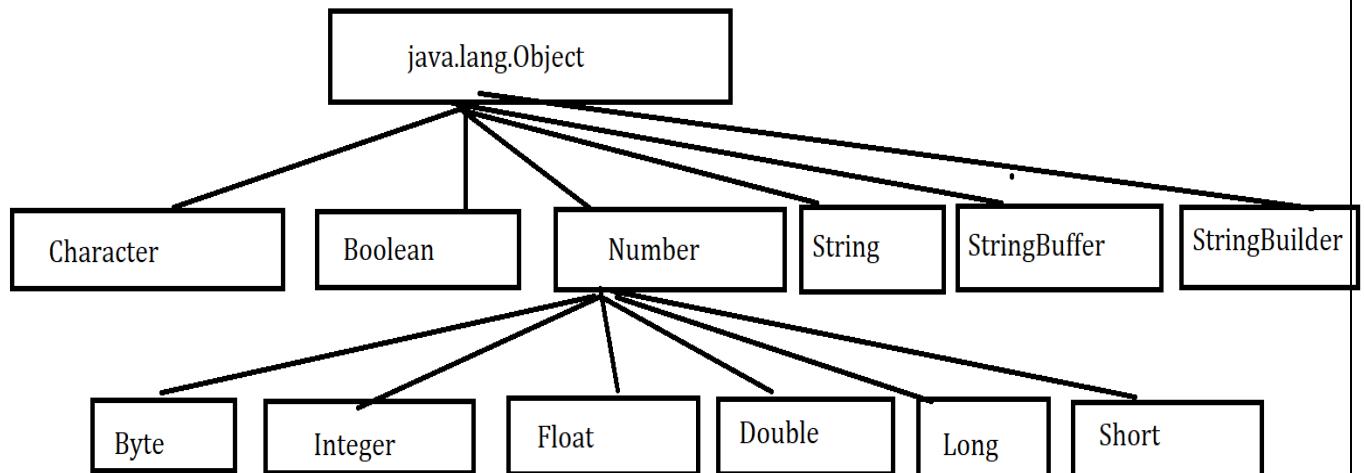
```

here String is class means str is reference of of String class  
and int a is primitive data type  
so we cannot convert the reference value to  
primitive value so we cannot perform conversion  
by implicit way or explicit way

If we want to solve this type of problem in java. Java Provide the special type of classes to us called as wrapper classes.

If we want to work with wrapper classes in java we have to know the following class hierarchy in java

---



**2) Object conversion or referential conversion:** here we perform conversion between two different objects

## **There Are Two Types of conversion in Wrapper classes**

---

**Autoboxing :** auto boxing means if we convert any primitive value to object value called as auto boxing.

**Autounboxing :** auto unboxing means if we convert any object to primitive value directly called as auto unboxing.

---

Following Examples the autoboxing and autounboxing

---

```
public class BoxingApplication {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Integer a; //Integer is class and a is reference here  
        int b=100; //int is data type so it is primitive type data  
  
        a=b; //autoboxing.  
        System.out.printf("A is %d\n",a);  
  
        int c;  
        Integer d = 600;  
        c=d; //auto unboxing  
        System.out.printf("C is %d\n",c);  
    }  
}
```

The auto boxing and auto unboxing get failed if we try to convert the different type of primitive with a different type of reference.

---

```
public class BoxingApplication {  
    public static void main(String[] args) {  
        String str="123";  
        int b=str;  
    }  
}
```

If we think about code we try to convert the string reference to integer reference using auto unboxing technique.

In this case auto boxing and auto unboxing get failed

So if we perform this type of conversion so java provides the special type of classes to us called as wrapper classes.

## **Number class**

---

Number class is used for convert the any numeric value to primitive type value and for that Number class provide the xxxValue() method to us for convert the Numeric object to primitive object.

int intValue() : this is used for convert any numeric object to the primitive type.

### **Example of intValue() method**

---

```
public class BoxingApplication {
```

```
public static void main(String[] args) {  
    Float a=5.4f;  
  
    int b= a.intValue(); //convert float object to integer primitive .  
  
    System.out.printf("B is %d\n", b);  
}  
  
}
```

float floatValue(): this is used for convert the any numeric object to floating primitive type.

Example

---

```
public class BoxingApplication {  
  
    public static void main(String[] args) {  
  
        Long l=100L;  
        float b= l.floatValue(); //convert Long object to float primitive .  
  
        System.out.printf("B is %d\n", b);  
    }  
  
}
```

double doubleValue(): this is used for convert the any numeric object to double type.

```
public class BoxingApplication {
```

---

```
public static void main(String[] args) {  
  
    Long l=100L;  
  
    double b= l.doubleValue(); //convert Long object to  
    double primitive .  
  
    System.out.printf("B is %d\n", b);  
}  
  
}
```

long longValue(): this is used for convert the any numeric object to long type.

```
public class BoxingApplication {  
  
    public static void main(String[] args) {  
  
        Float f=56.5f;  
        long b=f.longValue(); //convert Float object to long  
        primitive .  
        System.out.printf("B is %d\n", b);  
    }  
  
}
```

byte byteValue(): this is used for convert the any numeric object to byte type.

etc

```
public class BoxingApplication {  
  
    public static void main(String[] args) {
```

```

        Integer a=5;
byte b=a.byteValue();//convert Float object to long primitive .
System.out.println("B is "+ b);
}

}

```

**parseXXX()**: this is used for convert the string object to primitive type and it is static method present in every wrapper class.

### Example

---

**int variable=Integer.parseInt(string)**: convert string to integer  
**float variable=Float.parseFloat(String)**: convert string to float  
**long variable=Long.parseLong(String)**: convert string to long  
**double variable=Double.parseDouble(String)**: convert string to double.

etc

### Example

---

```

public class BoxingApplication {
    public static void main(String[] args) {
        String str="12345";
        int val =Integer.parseInt (str);
        System.out.printf ("Value is "+val);
        String str1="5.4";
        float value1=Float.parseFloat (str1);
        System.out.printf ("\nValue 1 is %f\n", value1);
    }
}

```

**valueOf()**: valueOf() method is used for convert the primitive type value to object value and it is also static method present in every wrapper class.

## Example

---

```
package org.techhub;
public class BoxingApplication
{
    public static void main (String[] args) {
        int a=12345;
        String str=String.valueOf(a); //convert integer primitive to string
        object
        System.out.printf ("String is %s\n",str);
        float b=5.4f;
        Float d=Float.valueOf(b); //convert float primitive to float object
        System.out.printf ("D is %f\n", d);
    }
}
```

**String toString():** this method is used for convert the any object to string.

```
public class BoxingApplication
{
    public static void main(String[] args) {

        Integer a=100;
        String str=a.toString();
        System.out.println("Str is "+str);
    }
}
```

**Note:** `toString()` we will discuss in depth when we learn the Object class.

## **String, StringBuffer And StringBuilder**

---

### **Q. What is the String in java?**

---

String is immutable class of java. In java every “ ” consider as string object

### **What is the immutable class?**

---

Immutable means once we assign value we cannot change later called as immutable.

If we want to work with string in java we have the two ways

---

#### **a) By using initialization technique:**

syntax: String variablename="values";  
e.g String s="Good";

#### **b) By using new keyword :**

**syntax:** String variablename = new String("value");  
e.g String str = new String("Good");

### **Q. what is the diff between above mention approaches?**

---

If we use the initialization technique for string then string object get created in string pool constant and if we use the new keyword for string object creation then string object get created in heap section.

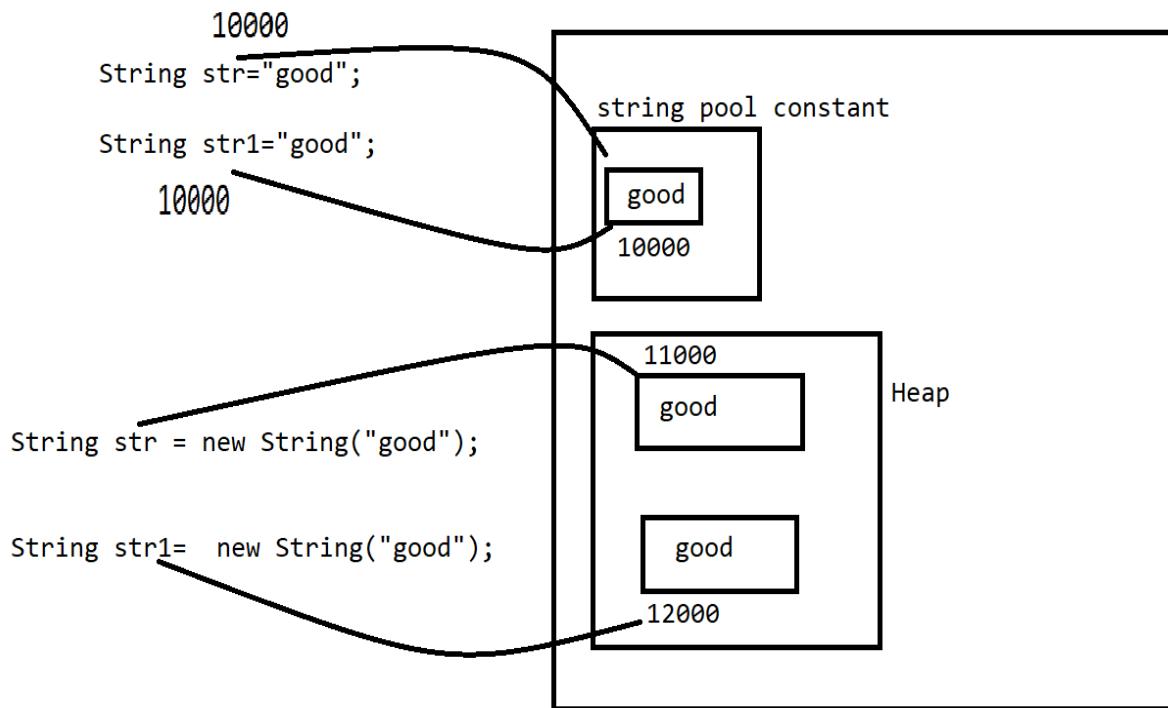
### **What is the string pool constant?**

---

String pool constant is part of memory in heap which specially design for store the object initialized by string the benefit is if we have two strings of same value and if we initialize it then JVM not create the separate memory for different object allocate single memory for all objects and use the same reference or address in all different variables.

But if we use the new keyword and if we create multiple objects with same value then JVM creates a new object every time.

**Following Diagram shows the meaning of above statements**



As per above diagram if we think about string constant diagram we have the two string names as str and str1 with value "good" so JVM creates a single object of both str and str1 and share the address of "good" object to str and str1.

If we think about the heap space diagram we have the string names as str and str1 with "good" value and JVM creates two different objects of same string.

## String with Initialization approach

```
public class BoxingApplication
{
    public static void main(String[] args) {

        String str="Good";
        String str1="Good";
        System.out.println("Address of str is "+System.identityHashCode(str));
        System.out.println("Address of str1 is "+System.identityHashCode(str1));

    }
}
```

}      Output:

```
Address of str is 123961122
Address of str1 is 123961122
Note: address of str and str1 is same
means we have same object and with two
references
```

## String with new keyword approach

```
package org.techhub;
public class BoxingApplication
{
    public static void main(String[] args) {
        String str=new String("Good");
        String str1=new String("Good");
        System.out.println("Address of str is "+System.identityHashCode(str));
        System.out.println("Address of str1 is "+System.identityHashCode(str1));
    }
}
}      Output
Address of str is 123961122
Address of str1 is 942731712
```

we have two string with different address  
space means we have two object in memory

## String class methods

String class provide the some inbuilt method to us for work with  
string

**int length():** this is used for calculate the length of string

**char charAt(int index):** this is used for return character from its index.

### Example

---

```
public class BoxingApplication
{
    public static void main(String[] args) {

        String str="Good Morning";
        int l=str.length();
        for(int i=0; i<l; i++)
        {
            char ch= str.charAt(i);
            System.out.printf("str[%d]--->%c\n",i,ch);
        }
    }
}
```

### Output

---

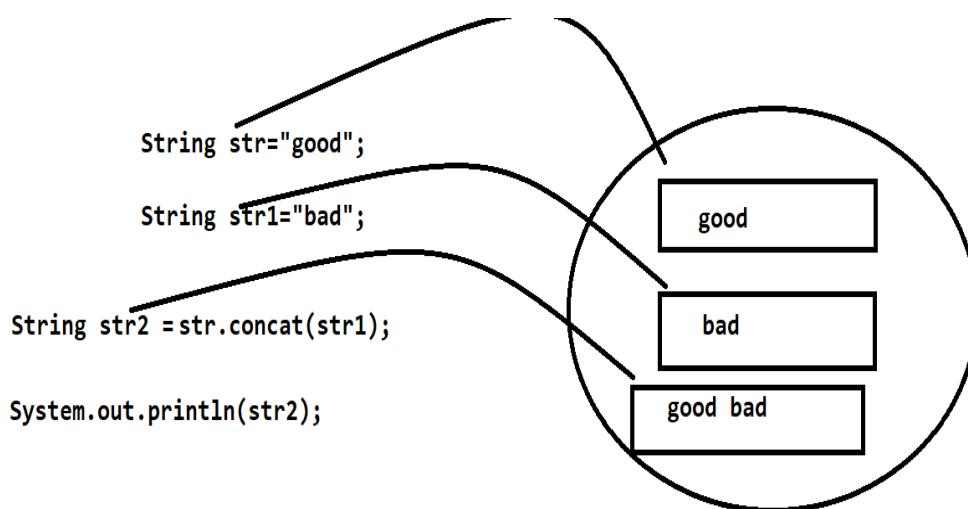
~~String str="good"; String str1="bad"; String str2 = str.concat(str1); System.out.println(str2);~~

```
str[0]--->G  
str[1]--->o  
str[2]--->o  
str[3]--->d  
str[4]--->  
str[5]--->M  
str[6]--->o  
str[7]--->r  
str[8]--->n  
str[9]--->i  
str[10]--->n  
str[11]--->g
```

**strcat()**: this is used for concat the two string with each other and generate the third new string from it.

Following Diagram shows the working strcat() method

---



## Example

---

```
public class BoxingApplication
{
    public static void main(String[] args) {

        String str="good";
        String str1="bad";

        String str2=str.concat(str1);
        System.out.println("String is "+str2);
    }

}
```

## Output

```
String is goodbad
```

**String toUpperCase()**: this is used for convert the lower case string to upper case string.

```

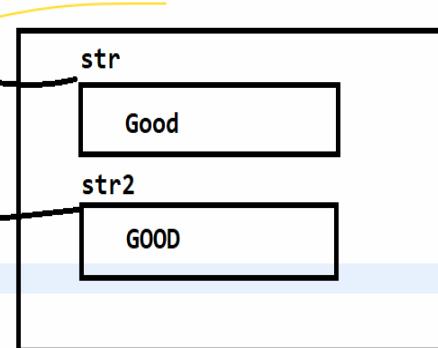
public class BoxingApplication
{
    public static void main(String[] args) {
        String str="Good";
        System.out.println("Before Conversion is "+str);
        GOOD
        String str2 = str.toUpperCase();
        System.out.println("After Conversion is "+str2);
    }
}

```

**Output**

---

Before Conversion is Good  
After Conversion is GOOD



**String trim ()**: this method is used for remove the white spaces from string at beginning and ending.

```

public class BoxingApplication
{
    public static void main(String[] args) {
        String str= " Good";
        System.out.println("Before Remove Spaces "+str);
        String str1=str.trim();
        System.out.println("After Remove Spaces "+str1);
    }
}

```

**Output**

---

Before Remove Spaces	Good
After Remove Spaces	Good

**String substring(int start,int end):** this method is used for extract the some specified portion from string.

```
String str="Good Morning India";
Morning
String str1 = str.substring(5,12);
starting index
                           ending index
```

## Sample code

---

```
public class BoxingApplication
{
    public static void main(String[] args) {

        String str="Good Morning India";
        String str1=str.substring(5,12);
        System.out.println("Extract String is "+str1);
    }
}
```

### Output

```
Extract String is Morning
```

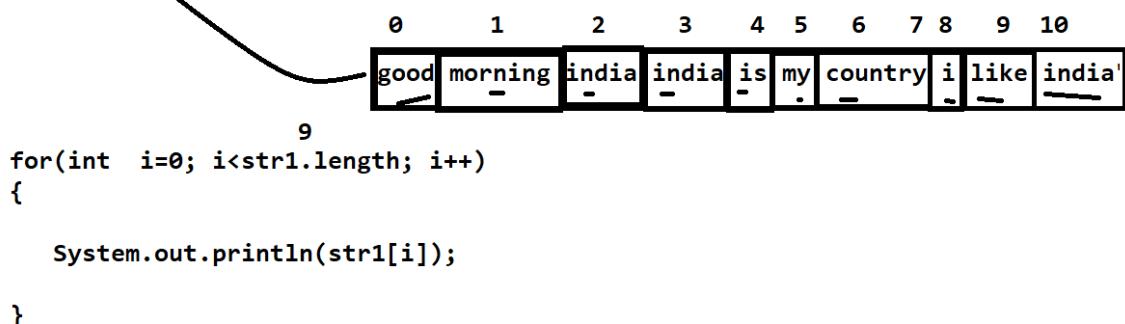
**String [] split(String character):** this method is used split the string using some specified character.

## Following Diagram shows the working split method

---

```
String str="good morning india india is my country i like india";
```

```
String str1 []= str.split(" ");
```



## Example

---

```
package org.techhub;
```

```
public class BoxingApplication
{
    public static void main(String[] args) {

        String str="Good Morning India";
        String str1[]=str.split(" ");

        for(int i=0; i<str1.length;i++)
        {
            System.out.println(str1[i]);
        }
    }
}
```

Output

---

```
Good
Morning
India
```

## **StringBuffer and StringBuilder**

---

StringBuffer and StringBuilder are the mutable classes in java  
Mutable means once we initialize value in it we can modify it called as mutable.

**Note:** we can use the StringBuffer and StringBuilder by using new keyword only.

StringBuffer and StringBuilder contain the some additional methods as per compare with string.

**void insert(int index,int data):** this method can append data on specified index in String

**void insert(int index,float data):** this is used for the floating data on specified

**Note:** this is the overloaded method with all data types for inserting data.

**void append(int data):** this method can add the new data at the end of string of type integer

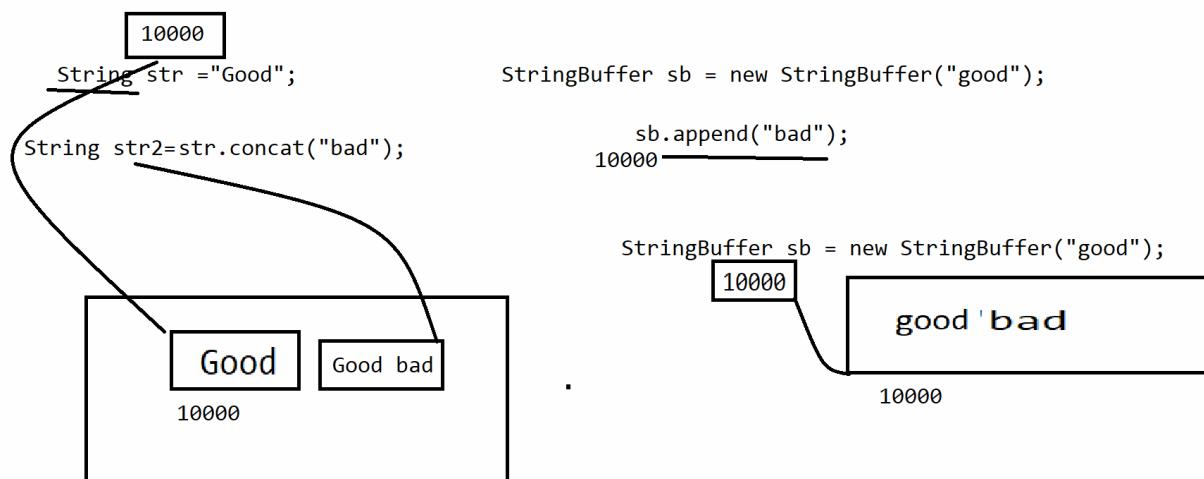
**void append(float data):** this method can add the new data at the end of string of type float

**Note:** this is also overloaded method with all data types

**void delete(int startindex,int endindex):** this is used for delete the data between two specified index

**Following Diagram shows the working of mutable and immutable**

---



## Source Code Example

```
public class MutableVsImutable
{
    public static void main (String x[])
    {   String str="Good";
        String str2=str.concat ("bad");
        System.out.println (str2);
        StringBuffer sb = new StringBuffer ("Good");
        sb.append (" bad");
        System.out.println (sb);
    }
}
```

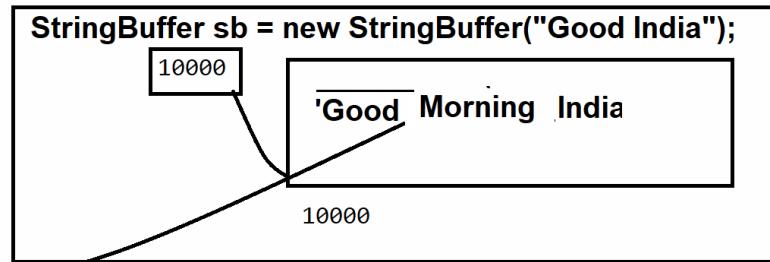
```
C:\Program Files\Java\jdk1.8.0_291\bin>java MutableVsImutable
Goodbad
Good bad
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>mspaint
```

## How to insert the value on specified index using StringBuffer

```
public class MutableVsImmutable
{
    public static void main(String x[])
    {
        StringBuffer sb = new StringBuffer("Good India");
        System.out.println("Before inserting value "+sb);
        sb.insert(5," Morning ");
        System.out.println("After inserting value "+sb);
    }
}
```

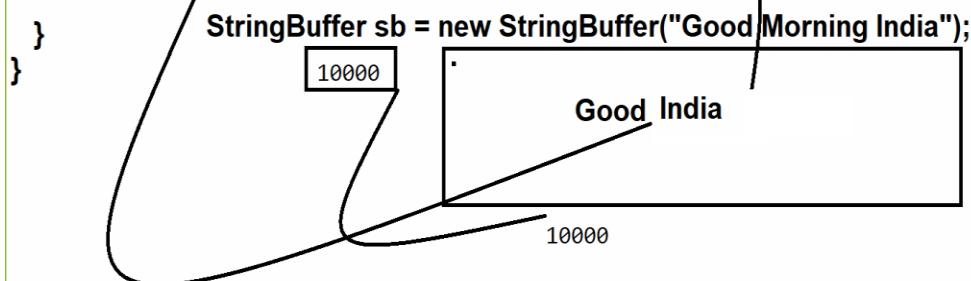
Before Inserting value is Good India  
After Inserting value Good Morning India



## How to delete data between two specified indexes from StringBuffer

```
public class MutableVsImmutable
{
    public static void main(String x[])
    {
        StringBuffer sb = new StringBuffer("Good Morning India");
        System.out.println("Before Deleting value "+sb);
        sb.delete(5,12);
        System.out.println("After Deleting value "+sb);
    }
}
```

C:\Program Files\Java\jdk1.8.0\_291\bin>java MutableVsImmutable  
Before Deleting value Good Morning India  
After Deleting value Good India



## What is the diff between StringBuffer and StringBuilder

The major diff between StringBuffer and StringBuilder is  
StringBuffer is synchronized and StringBuilder is not means  
StringBuffer is thread safe class and StringBuilder is not thread safe.

## **Multithreading in java**

---

Multithreading in java is process of executing multiple threads simultaneously.

### **What is the Thread?**

---

A thread is lightweight sub process, the smallest unit of processing it is a separate path of execution or thread is sub part of process

### **What is the process?**

---

Process current running program in ram or application execution in ram called as process.

### **Why use the multi threading in java or Advantages of Multithreading in java**

---

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

### **How to implement the thread practically in java**

---

If we want to use the thread in java we have the two ways

- 1) Using Thread class
- 2) Using Runnable interface

**Note:** Threading is concept of operating system and java is language where we can implement the thread concept

### **How to Implement the Thread by using Thread class**

---

If we want to implement or use the thread in java using Thread class we have the some important steps.

#### **1) Add the java.lang package in application**

**import java.lang.\* :** java.lang is default package in java means we not need to import it.

#### **2) Create the class and inherit the Thread class in it**

---

## **class A extends Thread**

```
{  
|  
}  
}
```

### **3) Override the run () method and write the thread logics**

---

run () method is originally member of Runnable interface and Thread is implementer class of Runnable interface so we have to override the run method for writing the thread logic

Internal code of Run implementation

---

```
interface Runnable  
{  
    public void run();  
}  
class Thread implements Runnable  
{  
    public void run()  
    {  
    }  
}  
class A extends Thread  
{  
    public void run()  
    { //write here logics of thread  
    }  
}
```

run () is abstract method from Runnable interface and which is used for writing the logics of thread.

### **Why java provide the run() as abstract method**

---

It is best example of abstraction suppose consider we have the three java developer name as A ,B and C and every developer want to write the logic of thread but every developer having different logic of thread so java provide the only one method name as run() as abstract so every developer can override the run() method and can modify its logic as per his need.

```
class A extends Thread
{
    public void run()
    {
        try
        {
            for(int i=1; i<=5; i++)
            { System.out.printf("I =%d\n",i);
            }
        }
        catch(Exception ex)
        { System.out.println("Error is "+ex);
        }
    }
}
```

#### 4) Create the object of Thread child class and use the Thread methods to work with Thread

**Thread class provide the number method to us to manage the thread**

**void start()** : this method is used for start the thread means when we call the start() method then run() method automatically get executed internally.

**static void sleep(int milliseconds)** : this method is used for sleep or stop thread execution for specified time period not permanently.

**void stop()**: this method is used for stop the execution of thread.

**void join()**: this method can hold the another execution thread execution when ever running thread is not completed.

**boolean isAlive()**: this method can check wheather thread is running or not or live or not if thread is running return true otherwise return false.

**void notify(),void notify(),wait(),wait(int),yield() etc** : we will discuss later in this chapter.

#### Example of Thread using start(),run() and sleep() methods

```
package org.techhub;
import java.lang.*;
class A extends Thread
{ public void run()
    { try
        { for(int i=1;i<=5; i++)
            { System.out.printf ("First Thread is %d\n",i);
              Thread.sleep (10000);
            }
        }
        catch (Exception ex)
        { System.out.println ("Error is "+ex);
        }
    }
}
class B extends Thread
{ public void run()
    { try
        { for(int i=1;i<=50; i++)
            { System.out.printf("Second Thread is %d\n",i);
              Thread.sleep(1000);
            }
        }
        catch(Exception ex){
            System.out.println("Error is "+ex);
        }
    }
}
public class ThreadingApplication {
    public static void main(String[] args) {
        A a1 = new A();
        a1.start();
        B b1 = new B();
        b1.start();
    }
}
```

## **join () methods of Thread class**

---

The join () method waits for a thread to die. In Other words it causes the currently running threads to stop executing until the thread it joins with completes its task or join() method hold the other thread execution when ever running thread is not completed

### **Example**

---

```

package org.techhub;
import java.lang.*;
class A extends Thread
{ public void run()
    { try
        { for(int i=1;i<=5; i++)
            { System.out.printf("First Thread is %d\n",i);
              Thread.sleep(10000);
            }
        }
        catch(Exception ex)
        {
            System.out.println("Error is "+ex);
        }
    }
}
class B extends Thread
{ public void run()
    { try
        { for(int i=1;i<=50; i++)
            { System.out.printf("Second Thread is %d\n",i);
              Thread.sleep(1000);
            }
        }
        catch(Exception ex)
        { System.out.println("Error is "+ex);
        }
    }
}
public class ThreadingApplication {

```

```

public static void main(String[] args)throws Exception {
    // TODO Auto-generated method stub

    A a1 = new A();
    a1.start();
    a1.join();
    B b1 = new B();
    b1.start();

}

```

**boolean isAlive()**: this method is used for check whether thread is live or not means check thread is running or not if thread is running return true otherwise return false.

### Example

---

```

package org.techhub;
import java.lang.*;
class A extends Thread
{ public void run()
  { try
    { for(int i=1;i<=5; i++)
      { System.out.printf("First Thread is %d\t%b\n",i,isAlive());
        Thread.sleep(10000);
      }
    }
    catch(Exception ex)
    {
      System.out.println("Error is "+ex);
    }
  }
}

class B extends Thread
{ public void run()
  { try
    { for(int i=1;i<=50; i++)

```

```

        { System.out.printf("Second Thread is %d\n",i);
          Thread.sleep(1000);
        }
      }
    catch(Exception ex)
    { System.out.println("Error is "+ex);
    }
  }
}

public class ThreadingApplication {
  public static void main(String[] args) throws Exception {
    // TODO Auto-generated method stub
    A a1 = new A();
    a1.start();
    a1.join();
    //die first thread
    System.out.println("Now Status of First Thread is "+a1.isAlive());
    B b1 = new B();
    b1.start();
  }
}

```

**void stop()**: this method is used for stop the execution of thread or terminate the thread execution.

```

package org.techhub;
import java.lang.*;
class A extends Thread
{ public void run()
  { try
    { for(int i=1;i<=5; i++)
      { System.out.printf("First Thread is %d\t%b\n",i,isAlive());
        if(i==3)
        {
          stop(); //terminate the execution of thread
        }
        Thread.sleep(10000);
      }
    }
  }
}

```

```
        }
    catch(Exception ex)
    {
        System.out.println("Error is "+ex);
    }
}
class B extends Thread
{
    public void run()
    {
        try
        {
            for(int i=1;i<=50; i++)
            {
                System.out.printf("Second Thread is %d\n",i);
                Thread.sleep(1000);
            }
        }
        catch(Exception ex)
        {
            System.out.println("Error is "+ex);
        }
    }
}
public class ThreadingApplication {

    public static void main(String[] args)throws Exception {
        // TODO Auto-generated method stub
        A a1 = new A();
        a1.start();
        a1.join();
        //die first thread
        System.out.println("Now Status of First Thread is "+a1.isAlive());
        B b1 = new B();
        b1.start();
    }
}
```

## Synchronization and Asynchronization in Threading

Synchronization means if two or more than two threads use the single resource/object one by one sequentially called as synchronization in Thread.

Asynchronization means if two or more than two threads use the single resource/object simultaneously called as Asynchronization.

Q. What is the meaning resource?

---

Here resource indicate the object of the particular class

### **Example**

---

```
package org.techhub;
class Table
{
    synchronized void showTable(int x)
    {
        try
        {
            for(int i=1;i<=10;i++)
            {
                System.out.printf("%d X %d = %d\n",i,x,i*x);
                Thread.sleep(1000);
            }
        }
        catch(Exception ex)
        {
            System.out.println("Error is "+ex);
        }
    }
}

class Two extends Thread
{
    Table t;
    void setTable(Table t)
    {
        this.t=t;
    }
}
```

---

```
public void run()
{
    t.showTable(2);
}
}
class Three extends Thread
{ Table t;
    void setTable(Table t)
    {
        this.t=t;
    }
    public void run()
    {
        t.showTable(3);
    }
}
public class SyncAsyncApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Table t=new Table();
        Two tw=new Two();
        tw.setTable(t);
        tw.start();
        Three th=new Three();
        th.setTable(t);
        th.start();
    }
}
```

Note: refer Thread part-5 video for above concept.

### **Wait(), notify() and notifyAll() methods of Thread class**

Wait() method is used for hold the thread execution for specified time period.

#### **There are two types of wait**

**1) Conditional wait:** conditional wait means if thread re-execute automatically after some specified time period called conditional wait.

**Syntax: void wait (int milliseconds):** this is used for hold the thread execution for specified time period when time period finish then thread re-executes itself.

**2) Unconditional wait:** unconditional wait means thread not re-execute self we have to send the request to thread for re-execution purpose for that we have the two methods name as notify() and notifyAll()

**Syntax: void wait ():**

**notify():** notify() method is used for recall the waited thread for re-execution in first in first out format and call one thread at time.

**notifyAll():** this method is used for call the all waited thread in single stroke and in last in first out format.

### **Example**

---

```
package org.techhub;
import java.util.*;
class Table {
    synchronized void showTable(int x) {
        try {
            for (int i = 1; i <= 10; i++) {
                System.out.printf("%d X %d = %d\n", i, x, i * x);
                if (i == 5) {
                    wait(); // unconditional wait
                }
                Thread.sleep(1000);
            }
        } catch (Exception ex) {
            System.out.println("Error is " + ex);
        }
    }
}
```

```
synchronized void restartTable() {
    try {
        notifyAll();
    } catch (Exception ex) {
        System.out.println("error is " + ex);
    }
}

class Two extends Thread {
    Table t;

    void setTable(Table t) {
        this.t = t;
    }

    public void run() {
        t.showTable(2);
    }
}
class Three extends Thread {
    Table t;

    void setTable(Table t) {
        this.t = t;
    }

    public void run() {
        t.showTable(3);
    }
}

public class SyncAsyncApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Table t = new Table();
    }
}
```

```
Two tw = new Two();
tw.setTable(t);
tw.start();
Three th = new Three();
th.setTable(t);
th.start();
Scanner xyz = new Scanner(System.in);
do {
    String msg=xyz.nextLine();
    if(msg.equals("restart"))
    {
        t.restartTable();
    }
    if(msg.equals("stop"))
    {
        System.exit(0);
    }
}while(true); //infinite string
}
```

### **What is the diff between wait() and sleep()**

- 1) wait() is method of java.lang.Object and sleep() is static method of Thread class
- 2) wait() method can work with conditional wait as well as uncondition wait but sleep() method can only work with conditional wait
- 3) wait() method only works in synchronized block and sleep() can work in synchronized as well as asynchronized block.

### **What is the diff between notify() and notifyAll() method**

notify() method call one thread time from waiting stage of threads and work in first in first out manner means it work in queue format and notifyAll() method call all the waited thread in last in first out format means notifyAll() method work as stack format.

The similarity of notify() and notifyAll() is both method from java.lang.Object class as well as they need synchronized block if we want to use it.

## **How to create the thread by using Runnable interface**

---

### **Q. why we need to implement the thread by using Runnable interface?**

---

If we use the Thread class for thread implementation purpose then there is possibility of multiple inheritance and in java we cannot perform the multiple inheritance by using class for that we have to use the interface for resolve this problem we use the Runnable interface.

### **How to implement the thread by using Runnable interface**

---

If we want to use the thread by using Runnable interface we have the some important steps.

#### **1) add the java.lang package in application**

---

java.lang is default package of java we not need to import it.

#### **2) Create the class and implement the Runnable interface in it**

---

##### **class A implements Runnable**

```
{  
}
```

#### **3) Override the run() method of Thread class and write its logic**

---

```
class A implements Runnable
{
    public void run()
    {
        try
        {
            for(int i=1; i<=5; i++)
            { System.out.printf("I =%d\n",i);
              Thread.sleep(1000);
            }
        }
        catch(Exception ex)
        { System.out.println("Error is "+ex);
        }
    }
}
```

### **3) Create the object of class in which Runnable implemented**

---

A a1 =new A();

### **4) create the object of Thread class in given fashion**

---

**Thread ref = new Thread (Runnable):** when we use the Runnable interface for thread implementation purpose then we have the pass Runnable interface reference in Thread class constructor. So as per the rule dynamic polymorphism if we have the reference of parent class then we can pass the reference of child as parent.

A a1 = new A();

Thread t = new Thread(a1);

Sample code

---

```
package org.techhub;
class A implements Runnable
{
    public void run() {
        try
```

```

    {
        for (int i=1; i<=5; i++)
        {
            System.out.printf ("I=%d\n", i);
            Thread.sleep (10000);
        }
    }
    catch (Exception ex)
    {
        System.out.println ("Error is "+ex);
    }
}
public class ThreadUsingRunnableApp {
    public static void main (String [] args) {
        A a1=new A ();
        Thread t=new Thread (a1);
        t.start ();
    }
}

```

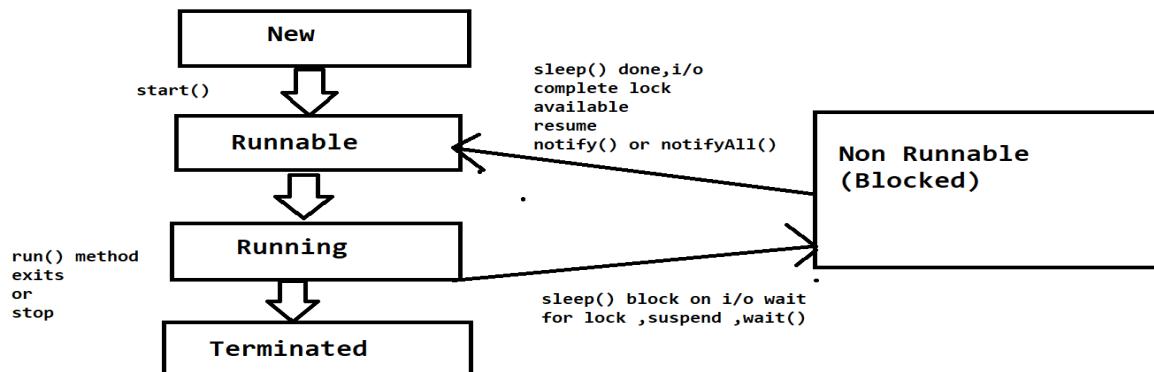
## **Thread Life Cycle**

---

A thread can be in one of the five states according to sun, there are 4 states in thread life cycle in java, new, runnable and non runnable and terminated.

There is no running state but for better understanding the thread we have to know the five 5 states of thread

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



**1) New:** The thread is in new state if you create an instance of Thread class but before the invocation of state method

**class A extends Thread**

```
{
}
```

A a1 = new A(); //New state

**2) Runnable:** the thread is in runnable state after invocation of `start()` method but the thread scheduler has no selected it to be then running thread.  
`a1.start();` //runnable

**3) Running:** the thread is in running state if the thread scheduler has selected it.

e.g `public void run() { logic }`

**4) Non-Runnable (Blocked):** this is the state when the thread is still alive but is currently not eligible to run

e.g `sleep()`, `wait()` etc

**5) Terminated:** a thread is in terminated or dead state when its `run()` method is exists.

## Executor Services and Thread pool

Executor a framework for creating and managing threads and thread pool.  
If use the Executor Framework or Executor framework help you

**1) Thread Creation:** it provides various methods for creating thread more specially manage the thread pool. That your application can use to run task concurrently.

**2) Thread Management :** it manages the life cycle of the thread in the thread pool. You don't need to worry about whether thread in thread pool is active or busy or dead before submitted task for execution.

**3) Task Submission and Execution:** Executor Framework provides methods for submitting task for execution in the thread pool and also gives power to decide when task will be executed.

**Example:** schedule task to be executed later or make then execute periodically.

**Java Currency API:** Api defines the following three executor interfaces that cover everything that is needed for creating and managing thread using executor services.

**1) Executor:** a sample interface that contains a method called execute (Runnable) to launch task specified by Runnable object.

**2) Executor Service:** A sub interface of executor that add functionality to manage life cycle of the task. It also provide a submit method whose overloaded version can accept Runnable as well as Callable Object.

Callable is similar to Runnable to except that the task specified by callable object can also return value.

**3) ScheduledExecutorService:** A sub interface of ExecutorService . It add functionality to schedule the execution of the task.

Apart from the above three interfaces The API also provides an executors class. That contains factory method for creating different kind of Executor Service.

## **Thread Pool Concept**

A Thread pool is used for reuse the previously created threads to execute the current task and offer a solution to the problem of thread life cycle overhead and resource thrashing.

## **Why we need to Thread Pooling concept**

---

In Server programming such as database and web server repeatedly execute the requests from multiple clients and these are oriented around processing a large number of short task. An Approach for building server application would be to create new thread each time a request arrives and service this new request in the new created thread. While this approach seems simple to implement it has significant disadvantages. A server that creates a new thread for every request would spend more time and consume more system resources in creating and destroying threads than processing actual request. Since active threads consume system resources, JVM creating too many threads at the same time can cause the system to run out of memory. So better way creating new thread every time and destroying if we reuse the same created thread for new request then we have to use the thread pool constant.