

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 sprice;
```

```
public void setDetail(String name,int id,float price,String dealerName,String  
compName,int pprice,int sprice)
```

```
{
```

```
this.name=name;
```

```
this.id=id;
```

```
this.price=price;
```

```
this.dealerName=dealerName;
```

```
this.compName=compName;
```

```
this.pprice=pprice;
```

```
this.sprice=sprice;
```

```
}
```

```
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.

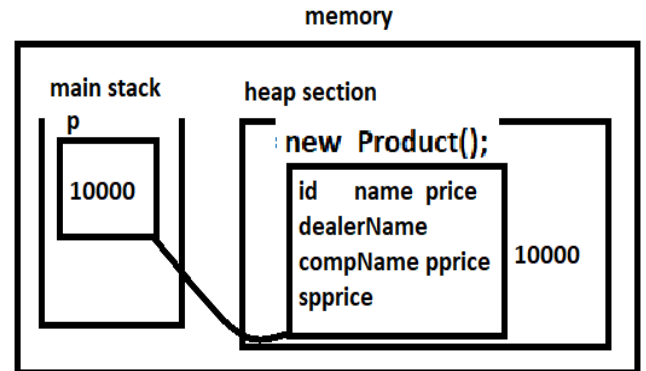
Following diagram shows the memory structure of object and reference

```

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 sprice;

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

```
}  
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("Biscuits");
s.setProductInfo(p);
s.validateUser("admin");
}

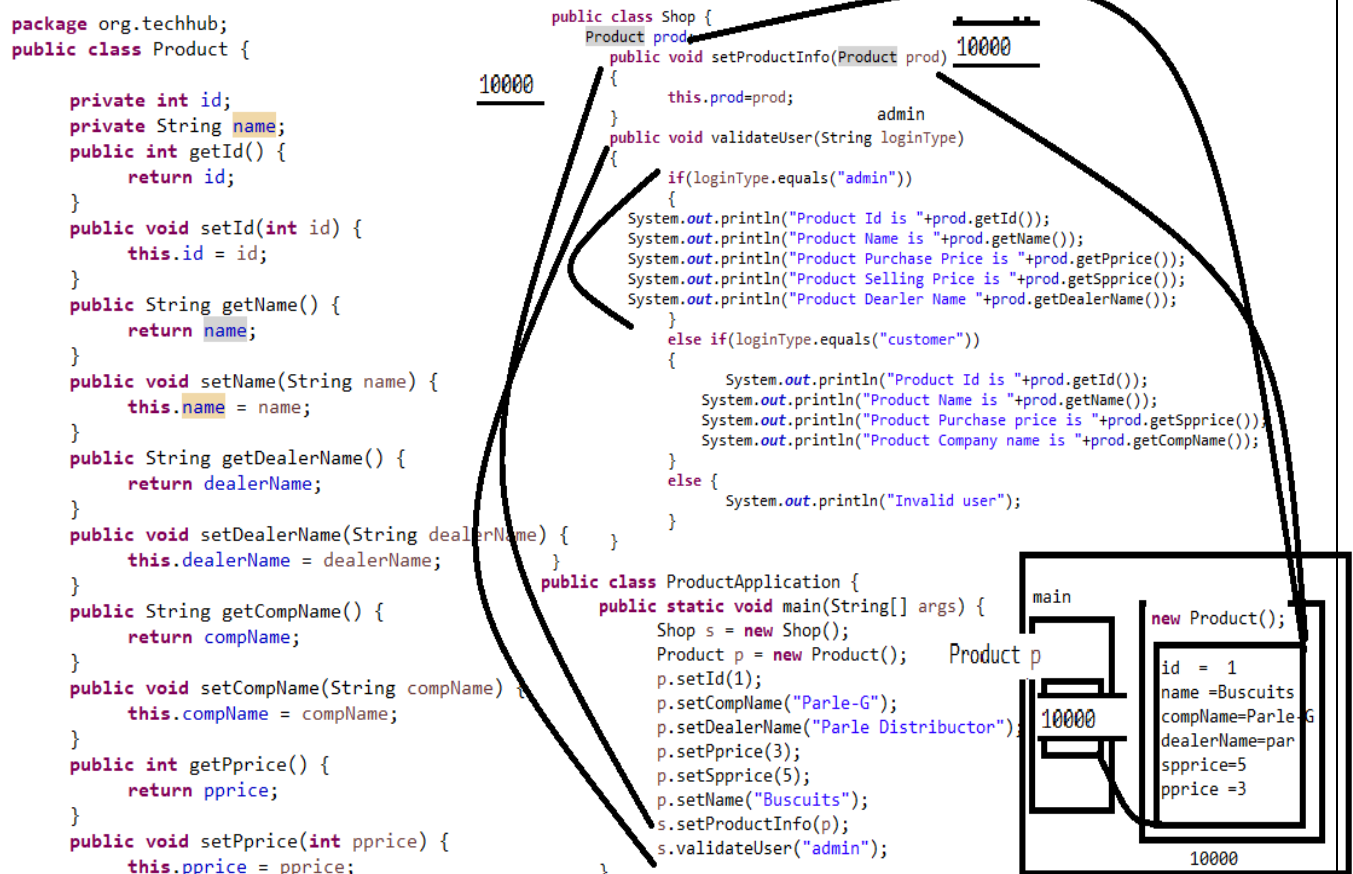
```

```

}

```

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[]=new int[] {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[]=new int[] {60, 60, 60, 60, 60, 60};
        cp.acceptMarks(a);
        cp.showPer();
    }
}

```

	0	1	2	3	4	5
marks	60	60	60	60	60	60
agg	1000	1002	1004	1006	1008	1012

Method with Variable arguments

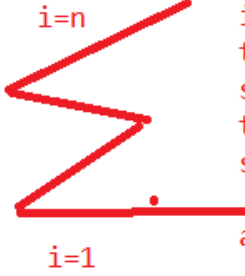
Method with variable argument is facility in java where we can pass the infinite parameter to the function

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



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

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
}
```

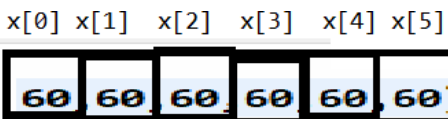
Here ... indicate the variable arguments to the function

... 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



```

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         s.setValue(60,60,60,60,60,60);
18     }
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 are 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 specifiers?

- 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 PERMGEN 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 PERMGEN or METASPACE

PERMGEN stands for **Permanent Generation**. It is a special type of heap space. It is separate from the main memory (heap). JVM uses **PERMGEN** 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. **PERMGEN** 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

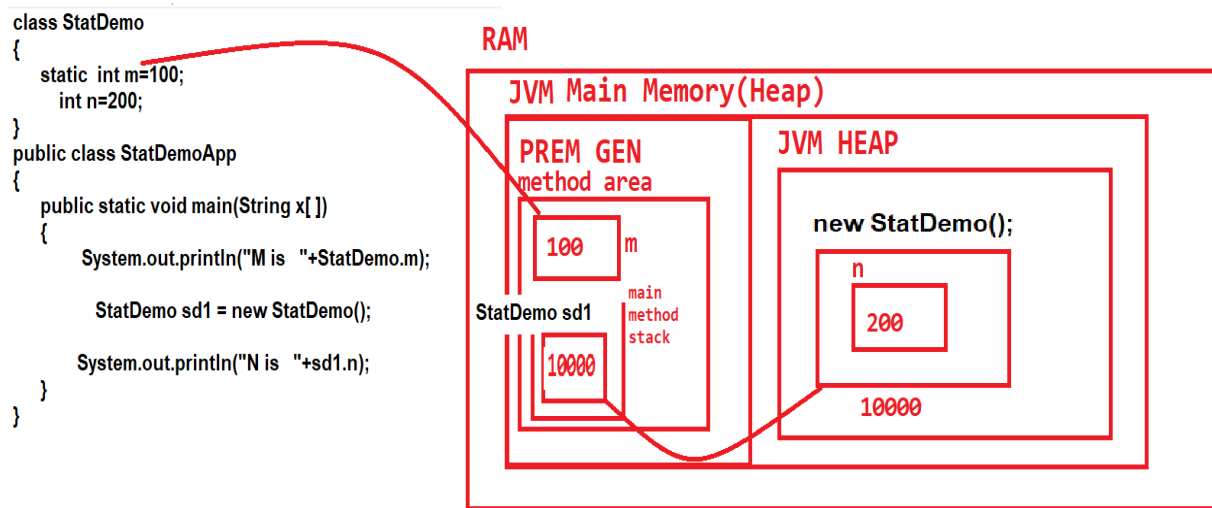
```
System.out.println (ABC.m);
```

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



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

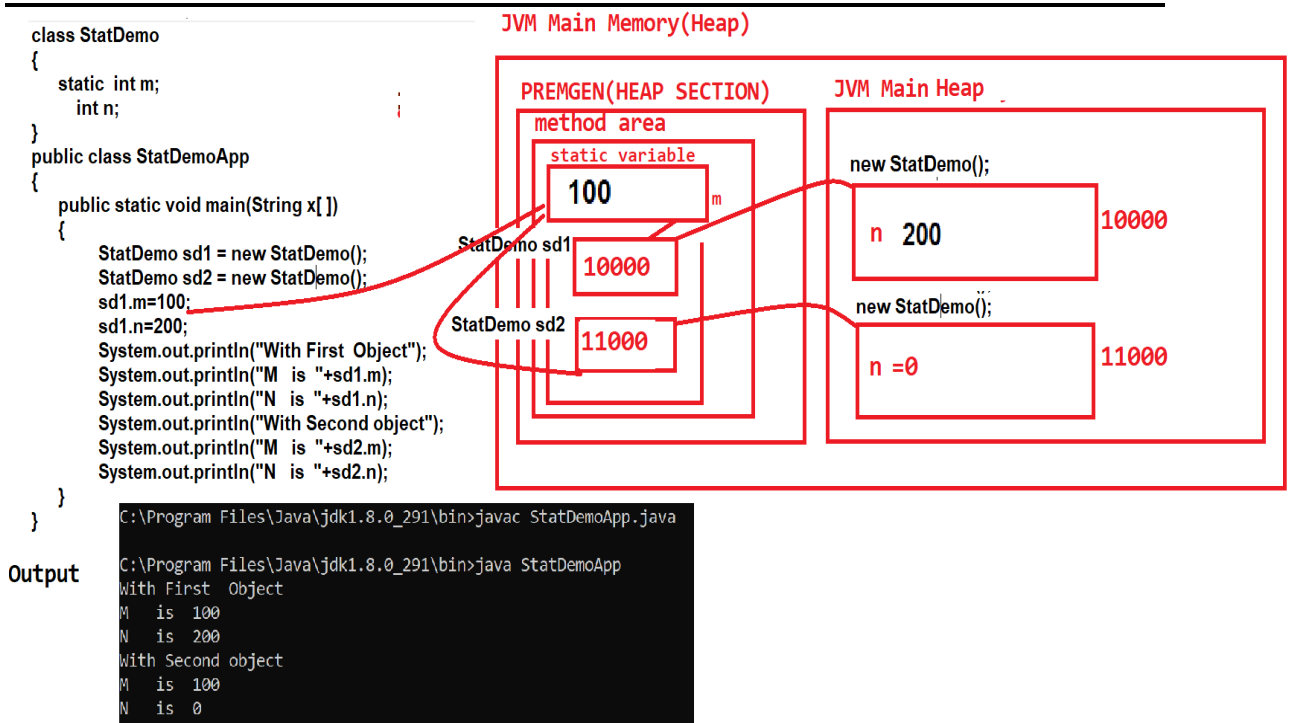
C:\Program Files\Java\jdk1.8.0_291\bin>java StatDemoApp
M is 100
N is 200

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



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 PERMGEN 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 our 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 PERMANENT 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?

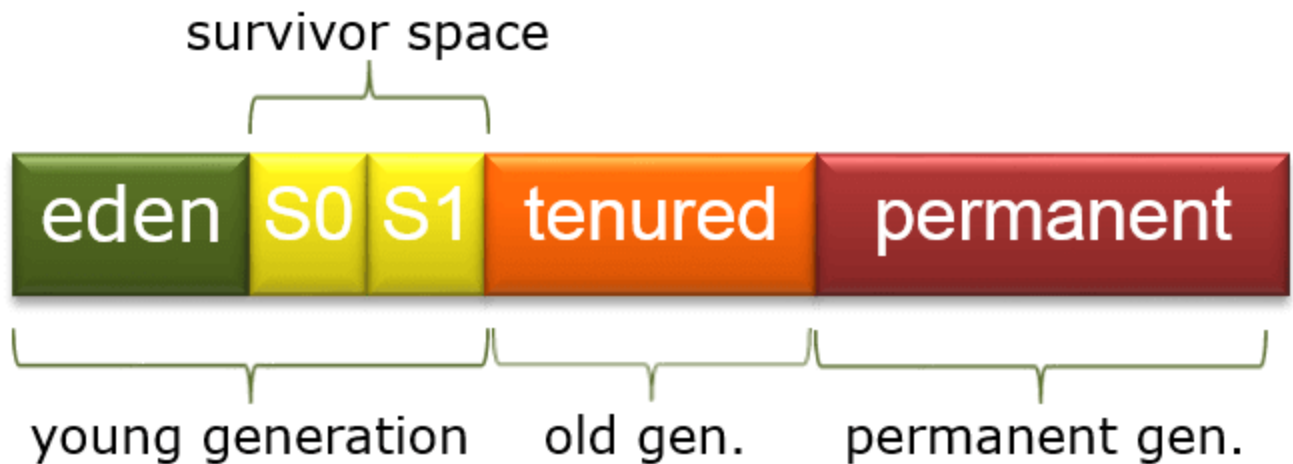
Java garbage collection is the process by which Java programs perform automatic memory management. Java programs compile to byte code that can be run on a Java Virtual Machine or JVM for short. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated 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 2-1 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

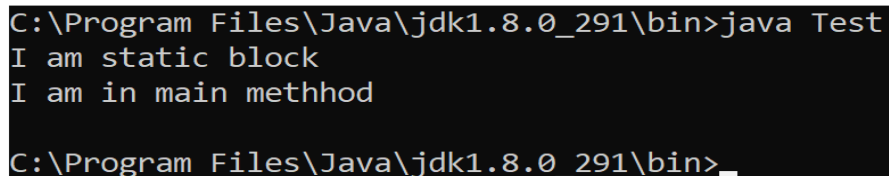
Static block in java

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) {
```

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

Here System will generate the compile time error

```
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;
        y=y;
    }
    void showAdd()
    {
        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();
    }
}

```

Note: Here instance variable name and local variable name is same so instance variable never use in block in this block so local variable not copy its value in instance variable

Here instance variable get use but the default value on instance variable is 0 so we have the output

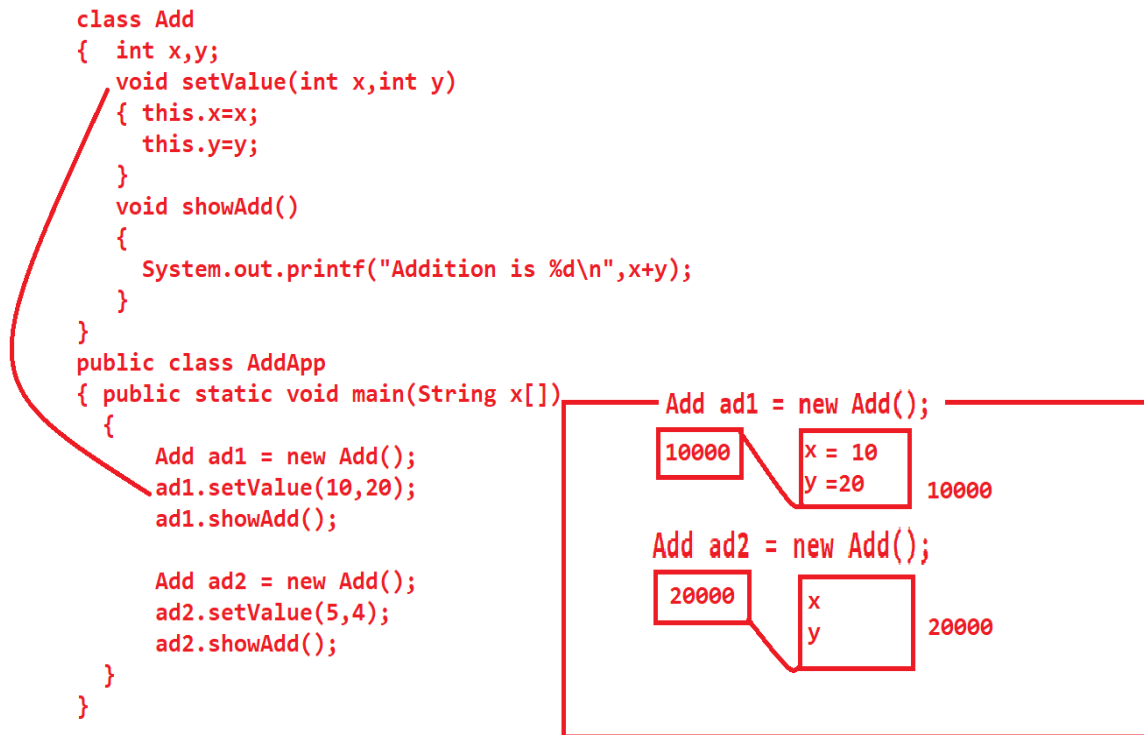
Addition is 0

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



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

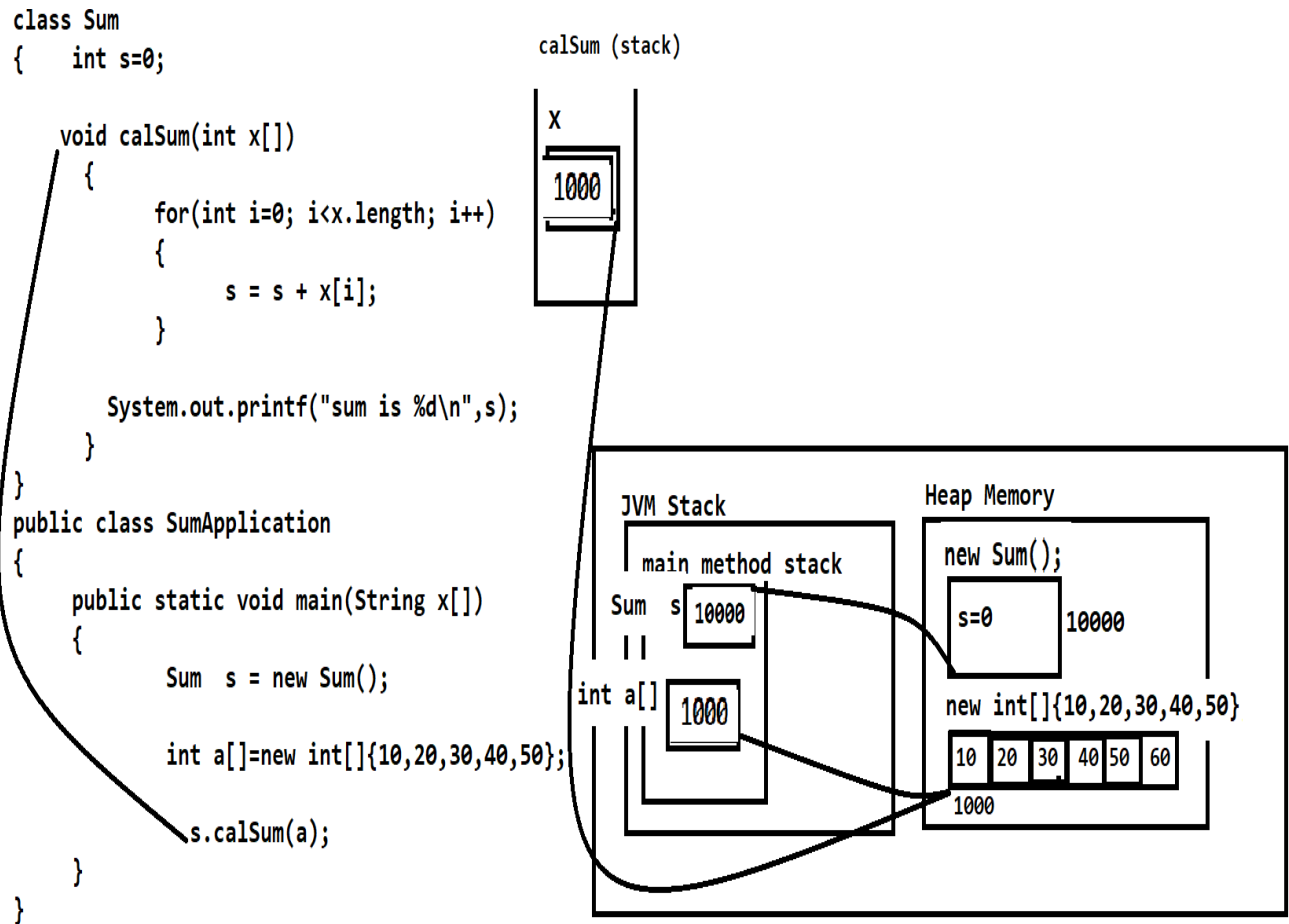
When we want to pass multiple parameter of same type to the function so passing single single parameter to the function is better way to pass array as parameter to the function.

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.

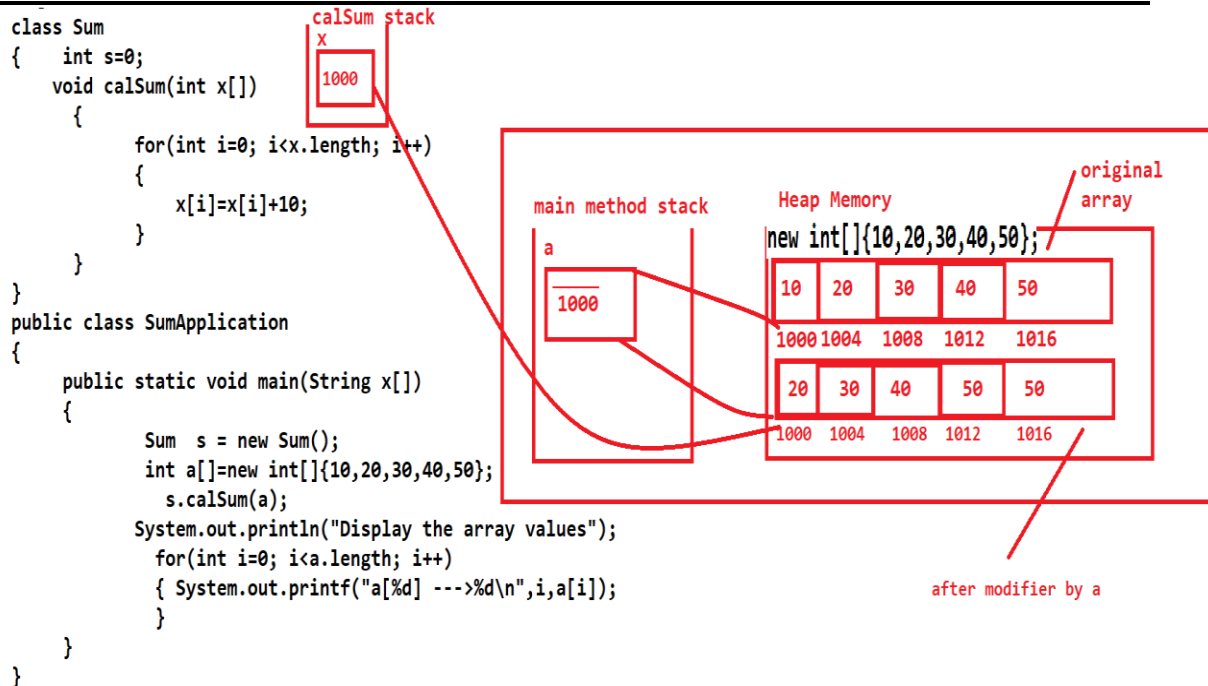
Example: we want to pass six parameter to the function using array and we want to calculate its addition.



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



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.

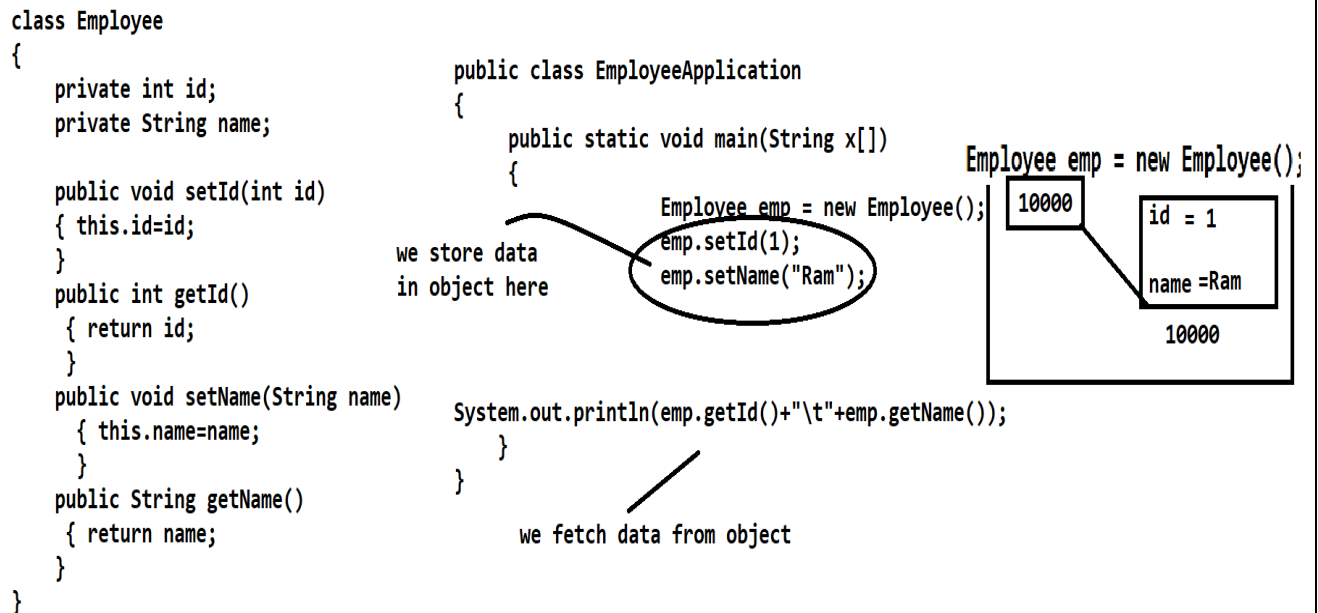
We can design the POJO class like as

```
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



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.

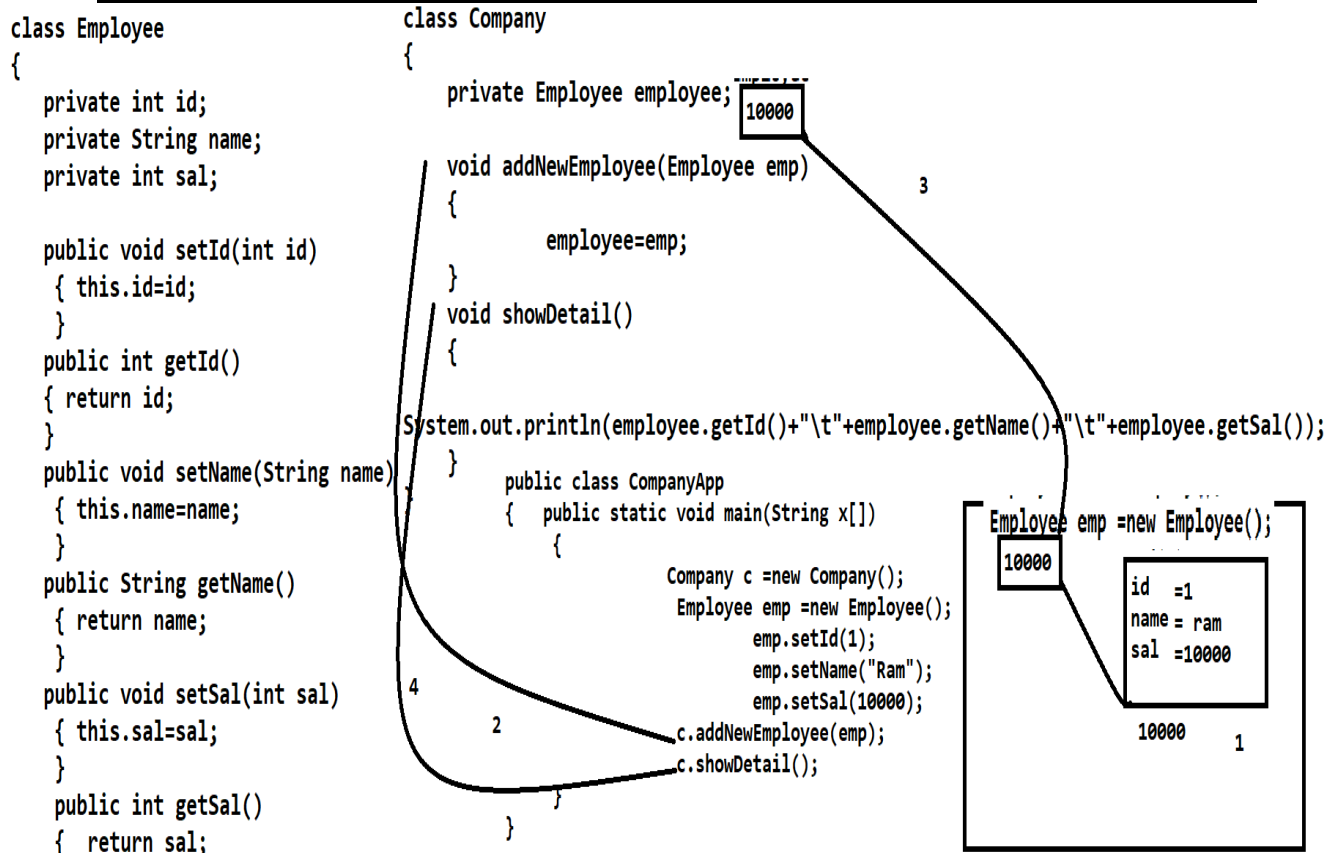
Means as per our example we create the class name as Employee and store the three values in Employee class id,name and sal.

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 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.

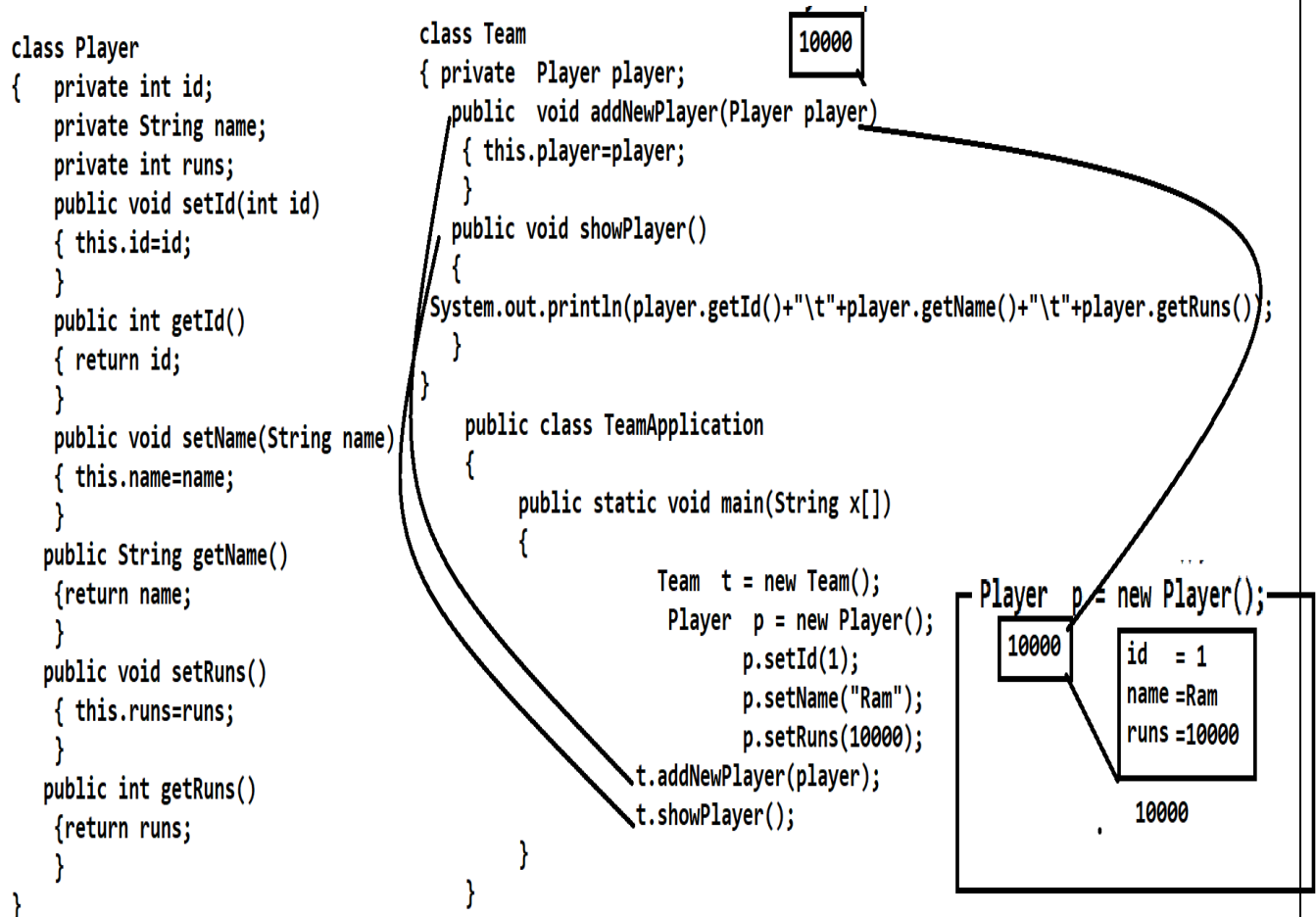
Following Program show the meaning of above paragraph



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

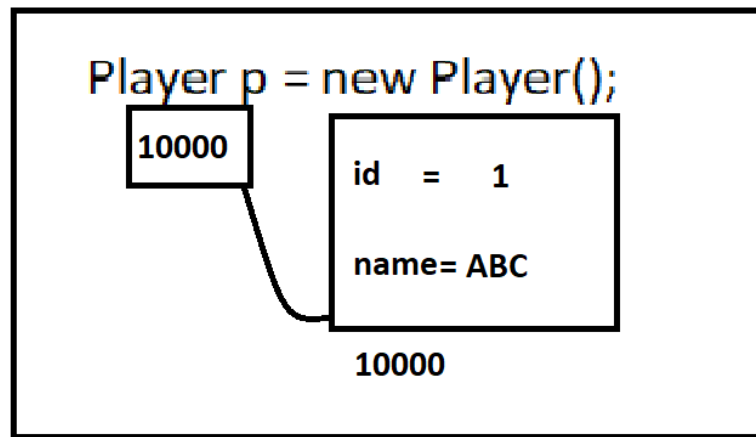
If we think about the above code then we have three class's one is Player which is POJO class and we use it for storing the information about the Player and in this class we have the two field name and id. We have the one more class name as Team and in Team class we have the method name as addNewPlayer(Team 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.

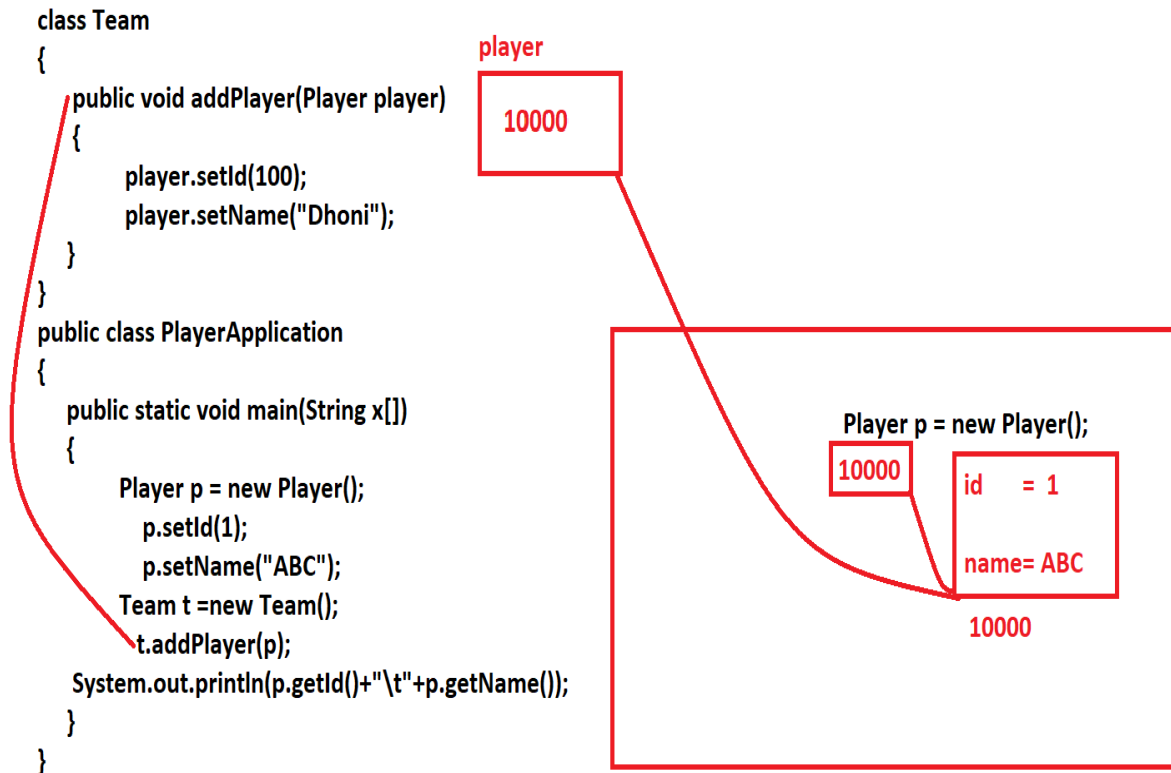
```
Player p = new Player();  
  
p.setId(1);  
  
p.setName("ABC");
```



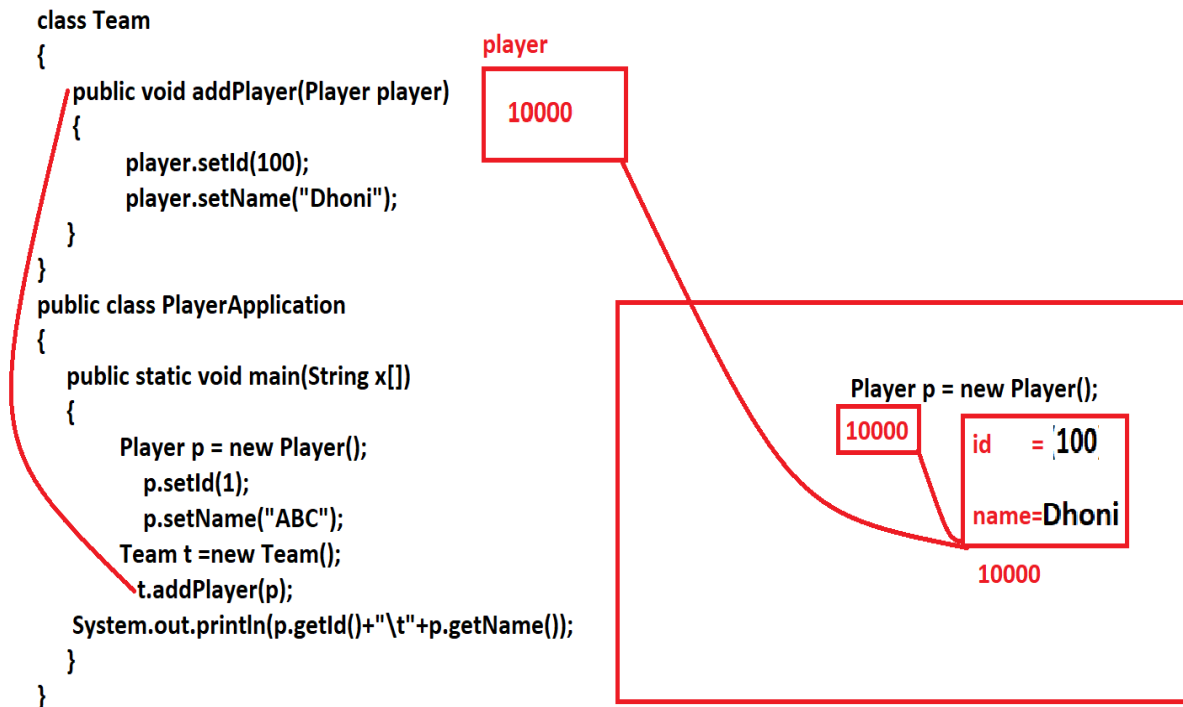
Here we create the object of Player class whose address is 10000 as per our example and we store this address in reference name as p after that we have the statement

```
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.



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 address as 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.



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

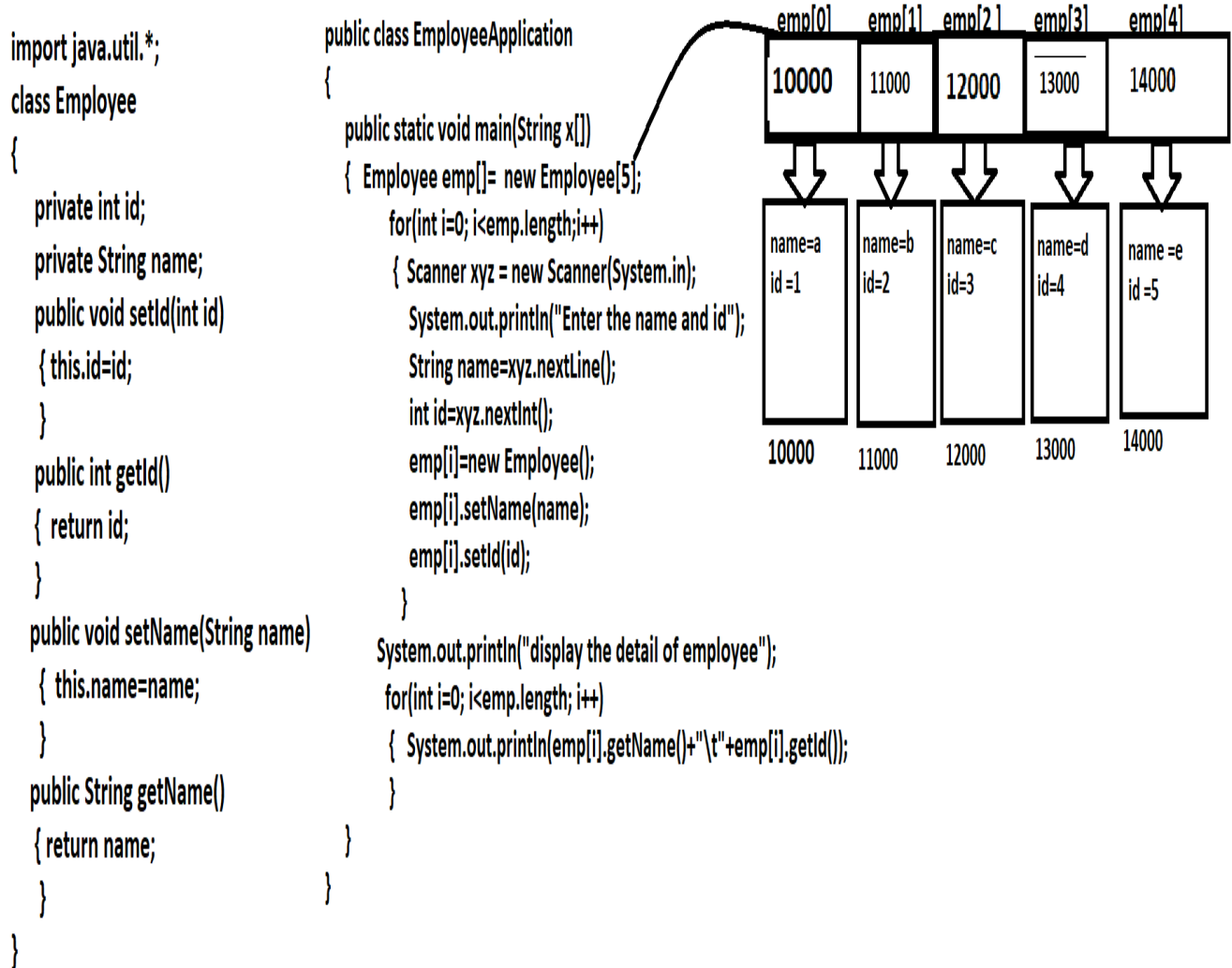
```

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.



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());
}
```

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

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

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

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

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

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