Classes and Object

Q. What is class?

---

class is a combination of state and behavior here state means variable declared within class and behavior means function define within class

or

class is combination of instance variable , static or class variable , methods , constructor, instance initializer, static initializer and nested classes

Instance variable means variable declared within class without static keyword called as instance variable
class variable means variable declared within class with static keyword called as class variable
If we define any function within class called as method.
Function same as class name or method same name as class name called as constructor
If we define block within class without static keyword called as instance initializer.
If we define block within class with static keyword called as static initializer.
If we define class within another class called as nested class

How to declare the class?

If we want to declare the class we have following syntax

```
access specifier class classname
{
    access specifier data type variablename;

   access specifier return type functionname(datatype arguments)
   {  write here your logics
   }
}
```

Example with source code

```
Example:
class ABC
{   int a=5;  //instance variable
    static int b=10; // class variable or static variable
     //nested class
     class MNO{

     }
     void show() //method
     {
     }
     ABC() //constructor
     {
     }
    //instance initializer
     {
     }
  //static initializer
    static
    {
     }
}
```

Q. Why use class or what is benefit of class?

There are three benefits of class

1. Ability to store different type of data

Note: some time people say class is complex data structure or it is a user defined data type.

```
class Employee
{
   private int id;
   private String name;
   private long sal;
   private Date dob;
}
```

Note: if we think about left hand side we have class name as Employee with field id with type int, name with type string, sal with long and dob with DOB it contain different types of data so we can say ability to store different types of data.

we provide class name as Employee means user can decide the name of class as well as user can decide what kind of data will store or use in class means class structure and data is 100% dependent on user choice so we can say class is a user defined data type or it is referential data type.

2. Reusability of code: if we think about class we can declare class only once and we can reuse it more than one time.

How we can reuse class more than one time?

_____

If we want to reuse class more than one time we can object or by using object we can reuse class more than one time.

Q. What is object?

_____

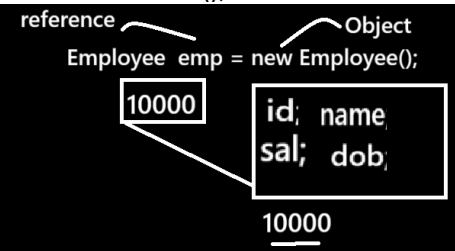
Object is block of memory where class data store
or
It is instance of class
or
It is runtime entity
or
It is photocopy of class.

How to create object in java

_____

Syntax: classname ref = new classname();

```
class Employee
{
   private int id;
   private String name;
   private long sal;
   private Date dob;
}
```



Note: if we think about Employee emp = new Employee() here emp is reference variable and new Employee() is our actual object.

Q. What is difference between reference and object?

_____

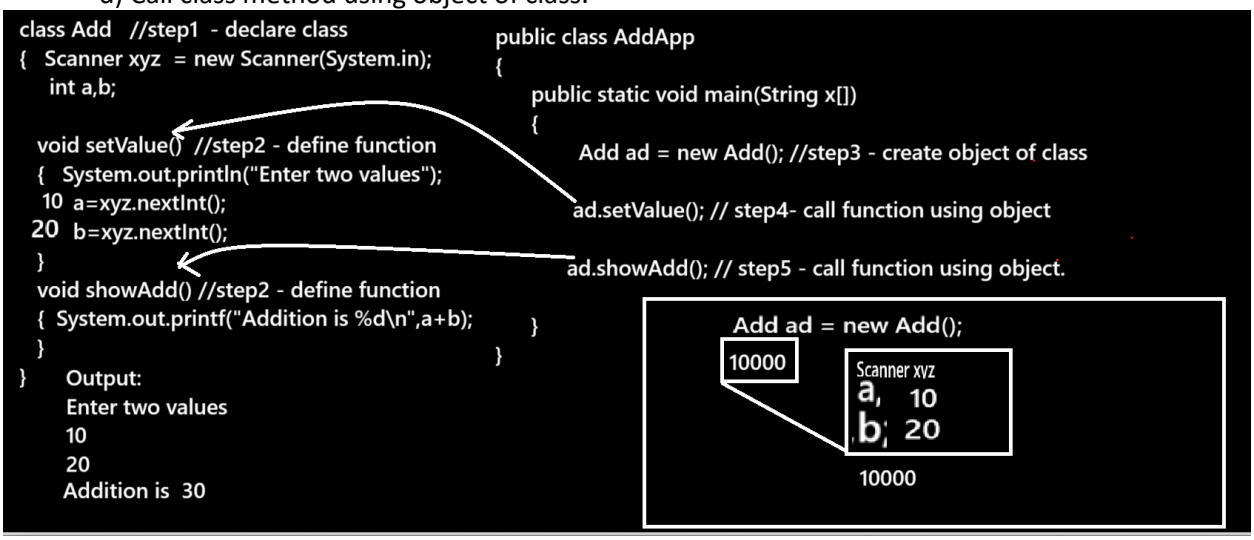
reference is variable which hold address of object and object is block of memory where class data store.

Standard steps to use any class

_____

a) Declare class
b) Declare variable & Define function within class
c) Create object of class

d) Call class method using object of class.



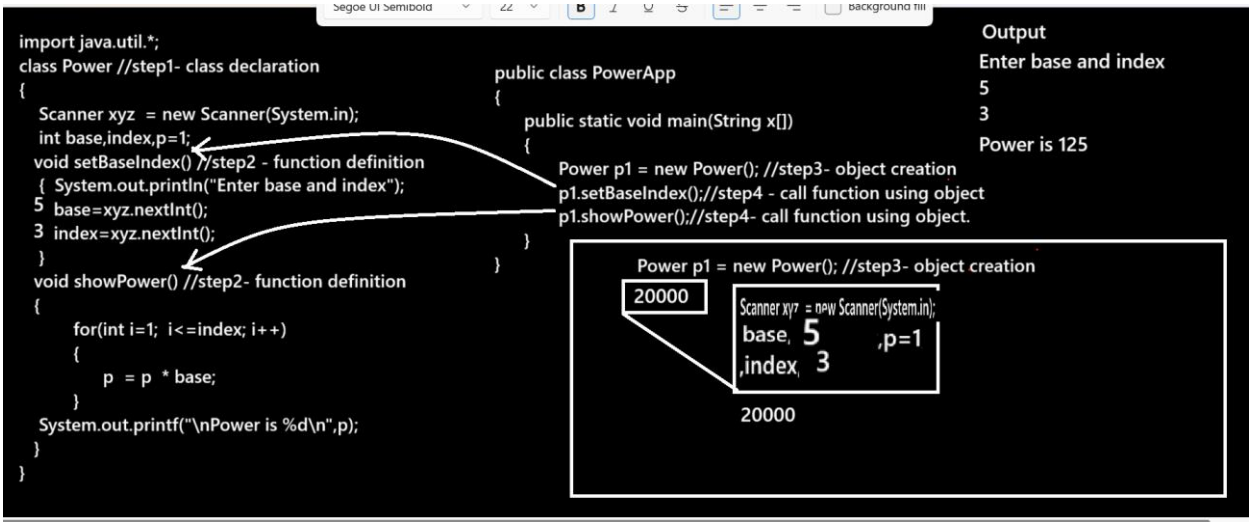
Example with source code

```
import java.util.*;
class Add  //step - class declaration
{ Scanner xyz  = new Scanner(System.in);
  int a,b;
  void setValue() //step2 - function definition
  { System.out.println("Enter two values");
    a=xyz.nextInt();
            b=xyz.nextInt();
  }
  void showAdd() //step2 - function definition
  { System.out.printf("Addition is %d\n",a+b);
  }
}
public class AddAppSep2024
{ public static void main(String x[])
        {       Add ad = new Add(); //step3 - object creation
                  ad.setValue();  //step4 - member calling by using object
                  ad.showAdd();   //step4 - member calling by using object
        }
}
```

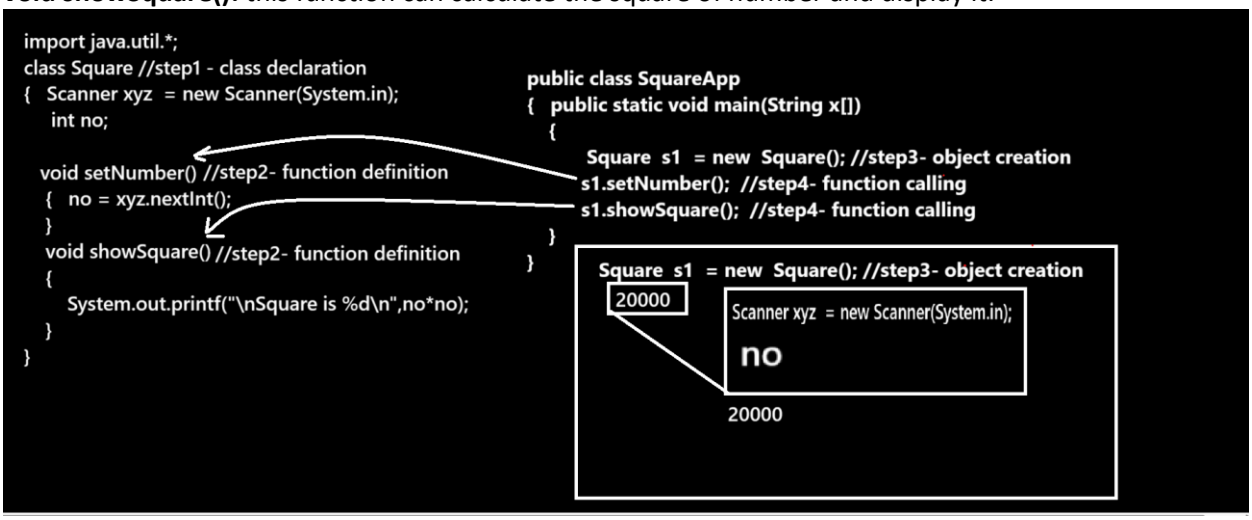
Output

```
C:\Program Files\Java\jdk-23\bin>javac AddAppSep2024.java

C:\Program Files\Java\jdk-23\bin>java AddAppSep2024
Enter two values
10
20
Addition is 30
```

Example: WAP to create class name as Power with two functions
void setBaseIndex(): this function can accept base and index as input value
void showPower(): this function can calculate power of two number and display it.



Example: WAP to create class name as Square with two functions
void setNumber(): this function can accept number from keyboard
void showSquare(): this function can calculate the square of number and display it.



If we think about above statement Square s1 = new Square() here s1 is reference of Square class and new Square() is actual object

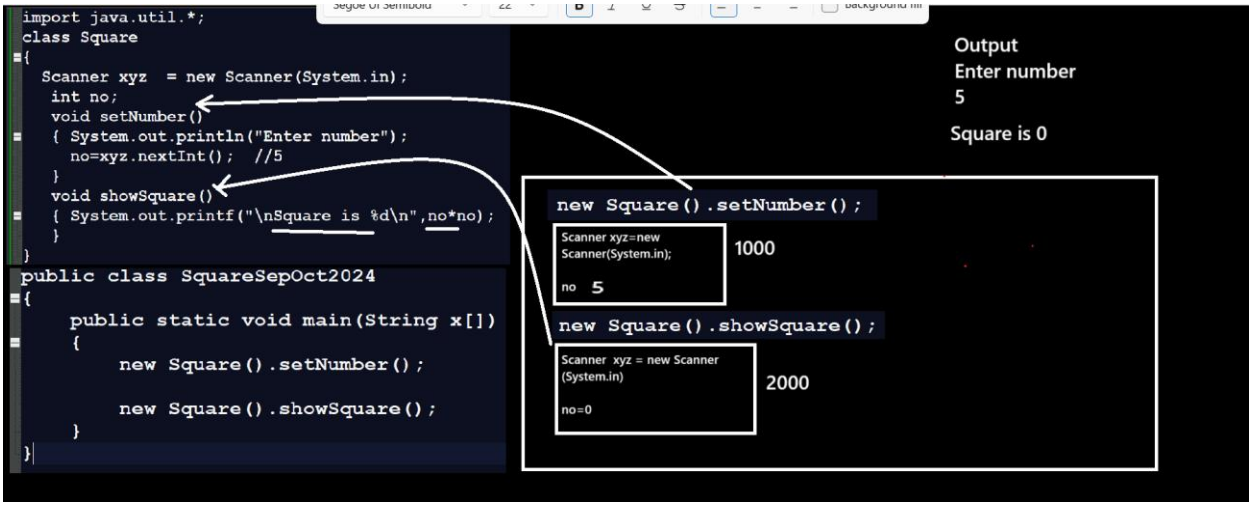
Q. What is difference between reference and object?

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

Q. What happen if we not use reference with object?

if we not use reference with object then JVM create new object every time in memory and when we have new object every time in memory means we have new block every time in memory and every object has different content means we cannot use same object content/data more than one time if we required.

Note: if we not use reference with object then technically object known as anonymous object.



if we think about above code we have statement new Square().setNumber() here we create object of Square class whose address is 1000 and setNumber() function call by object whose address is 1000 and setNumber() function we have logic

Enter number

5

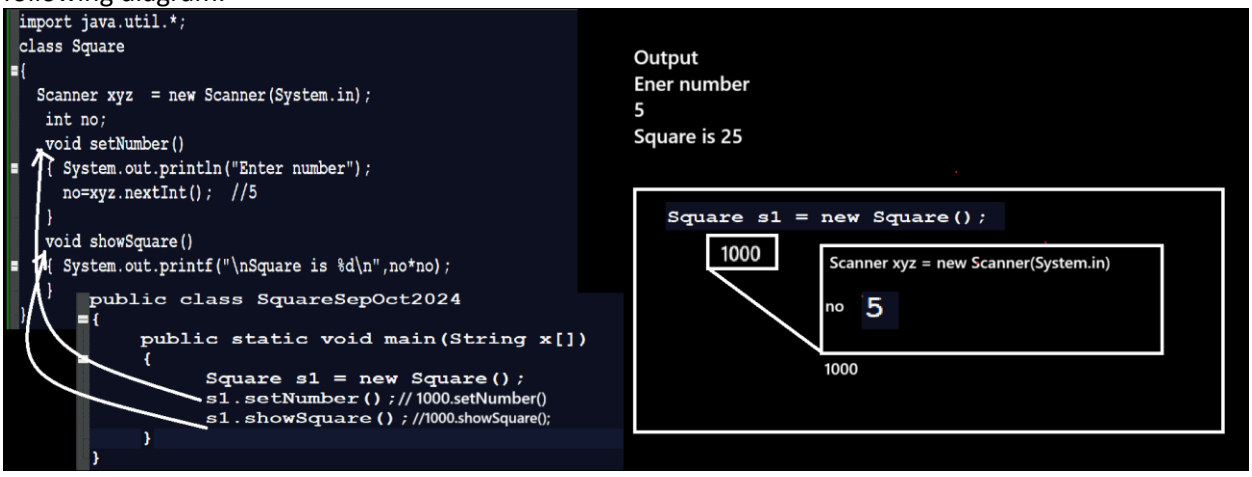
so this 5 value get stored in no=5 in object whose address is 1000 again we have one more statement in class new Square().showSquare() means we create second object in memory whose address is 2000 we consider diagram and newSquare().showSquare() indicate we call function using object whose address is 2000 or reference is 2000 so in showSquare() we have logic

Square is no*no means this no is refer from second object whose address is 2000 but we initialize value to object whose address is 2000 so the default of value no is 0 so we get answer square is 0

so the conclusion is when we use new keyword then every time new object in memory means we can say anonymous object can use only once in application cannot use more than one time but if we want to proper answer of above so we required to object whose address is 1000 two times or more than one time means first time with setNumber() function and second time with showSquare() function so for resolve this problem we need to store its address in class variable i.e reference

Q. Why use reference with object?

If we want to use same object more than one time then we can use reference with object shown in following diagram.

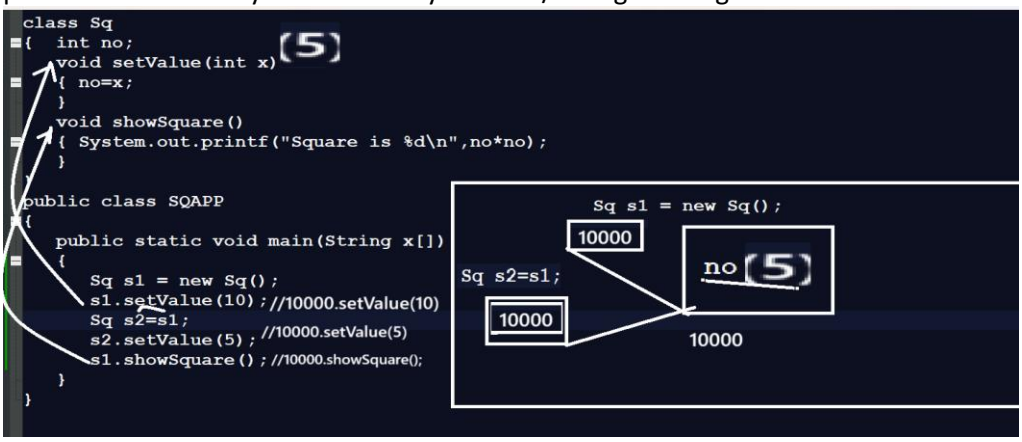


If we think about above diagram we have statement Square s1 = new Square() means we have object in memory whose address is 1000 and we store this address in reference variable s1 and we have statement s1.setNumber() means 1000.setNumber() means we call this function using object whose address is 1000 and when we call this function we have logic
Enter number
5
Again we have statement s1.showSquare () the meaning of this statement is we not create new object in memory just we use the object whose address is 1000 so means we use same object with two functions means two time using reference so showSquare() contain logic Square is no*no means this no variable refer from address 1000 i.e. previous object so we get answer is 25 means the conclusion is we can use same object more than one time with the help of reference

Q. Can we use more than one reference on single object and what happen if we use it?

---

yes we can apply more than one reference on single object and if we apply more than one reference on single object and if we perform change on object content by using any one reference then object previous content may be lost or may be state/data get change

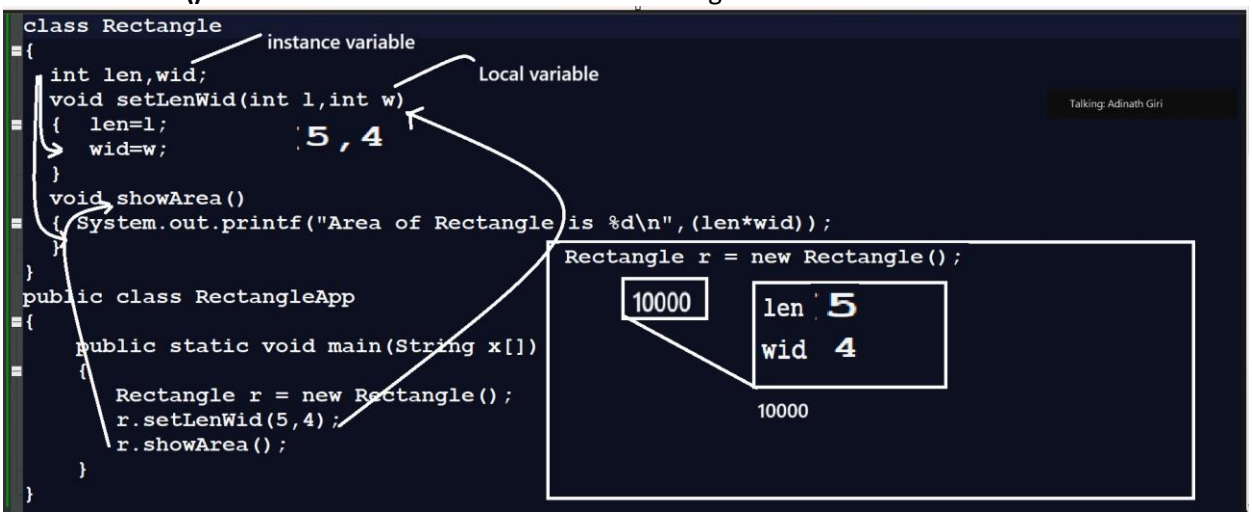


Note: if we think about above diagram we have statement Square s1=new Square() here we create object in memory whose address is 10000 and we store this address in reference variable s1 and we have statement s1.setValue(10) means 10000.setValue(10) means we store 10 value in no variable whose object address is 10000 again we have statement Square s2=s1 means we initialize or assign address of s1 in s2 reference means s2 points to 10000 location so we have only one object in memory but we have two reference variable which points same object and again we have statement s2.setValue(5) means 10000.setValue(5) so we pass 5 value in no variable whose object address is 10000 means this newly pass 5 value get override on 10 which is previous value of no variable and when we call s1.showSquare() means 10000.showSquare() so 10000 address contain updated value i.e no=5 so we have square is 25 so final conclusion is value change by s2 reference but s1 get impacted because reference s2 change the state of object or variable of object or data of object also
means when we have multiple reference on single object and if we change object using any reference then all references get affected.

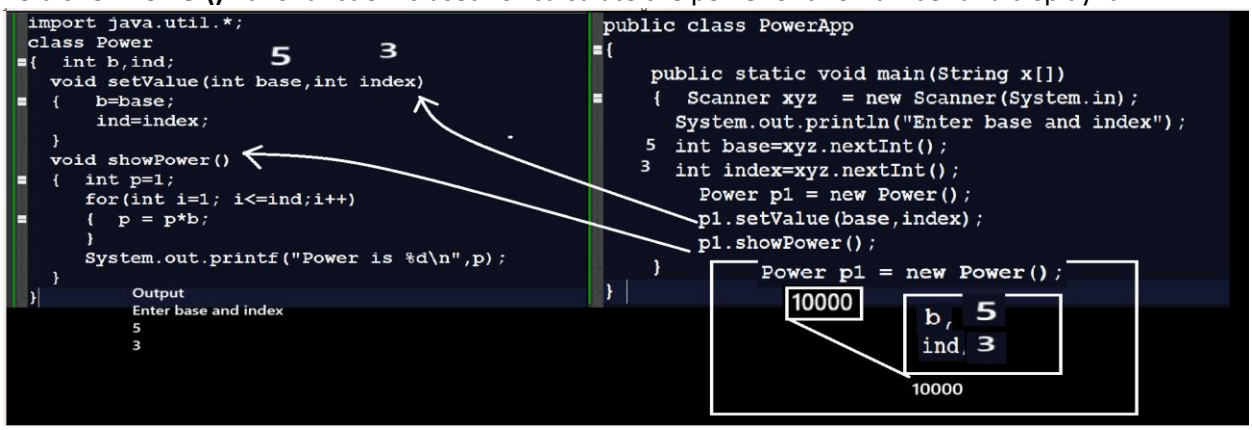
How to pass parameter to class function

If we want to pass parameter to class function we need to copy the value of local variable in to instance variable because local variable cannot access outside of his function block
Example:

WAP to create class name as Rectangle with two functions
void setLenWid(int wid,int len): this function can accept two values as parameter
void showArea(): this function can calculate area of Rectange



Example: WAP to create class name as Power with two functions
void setValue(int base,int index): this function is used for accept the two values as parameter first is base and second index.
void showPower(): this function is used for calculate the power of two number and display it.



Method with variable arguments

Method with variable argument is facility in java where we can pass n number of parameter to function called as method with variable arguments.

Syntax of method with variable arguments

```
return type functionname(datatype  ...variablename)
{
}
Note:  ... indicate the variable arguments

Example:
void calSum(int ...x)
{
}
if we think about above code we can say calSum() can accept infinite number of parameter of type
integer.
```

Note … internally work as dynamic array.

```
class Sum
{   void calSum(int ...x)
    {   int s=0;
        for(int i=0; i<x.length; i++)
        { s = s  + x[i];
        }
   System.out.printf("Sum of all value is %d\n",s);
    }
}
public class SumApp
{ public static void main(String x[ ])
    {   Sum  s1 = new Sum();

        s1.calSum(10,20,30,40,50);
    }
}
```

10 20 30 40 50

Example: WAP to create class name as MaxFinder with two functions

void acceptValue(int …x): this function can accept infinite parameters
int getMaxValue(): this function can find the max value from array and return it or variable argument and return it.

```
class MaxFinder
{  int []arr;
   void setValue(int ...v)
   { arr=v;
   }
   int getMax()
   {  int max=arr[0];
     for(int i=1; i<arr.length; i++)
     {
         if(arr[i]>max)
         {
            max=arr[i];
         }
     }
     return max; //9
   }
}
```

6 7 8 9 3

```
public class MaxFinderApp
{
    public static void main(String ...x)
    {
       MaxFinder mf= new MaxFinder();

       mf.setValue(6,7,8,9,3);
    9  int result=mf.getMax();
       System.out.printf("Max value is %d\n",result);
    }
}
```

Example: WAP to create class name as SearchLinear with two functions
void setValue(int …x): this function can accept infinite integer as parameter
int getResult(int value): this function can pass value as parameter and search in variable argument and if value found return its index if not found return -1

```
class Search
{  int arr[];
  void setValue(int ...v)
  { arr=v;
  }
  int getResult(int value)
  {
     int index=-1;
            for(int i=0; i<arr.length; i++)
            {
               if(arr[i]==value)
                     {  index=i;
                       break;
                     }
            }
            return index;
  }
}
public class SearchApp
{
   public static void main(String x[])
          {
                  Search s1 = new Search();
                  s1.setValue(6,4,5,2,9);
                  int result = s1.getResult(5);
                  if(result!=-1)
                  { System.out.println("Data found");
                  }
                  else{
                           System.out.println("Data not found");
                  }
          }
}
```

Example: WAP to Create class name as Sort with two functions
void acceptValue(int …x): this function can accept integer as parameter
void performSort(): this function can perform sorting on array and display it.

Example with source code

```
class Sort
{   int arr[];
   void acceptValue(int ...x)
          {
            arr=x;
          }
          void display()
          {
            for(int i=0; i<arr.length;i++)
```

```
        { System.out.printf("%d\t",arr[i]);
        }
    }
    void performSort()
    {

        for(int i=0;i<arr.length;i++)
        {
            for(int j=(i+1); j<arr.length; j++)
                {
                    if(arr[i]>arr[j])
                        {
                            int temp=arr[i];
                                arr[i]=arr[j];
                                arr[j]=temp;
                        }
                }
        }

    }
}
public class SortVarArgApp
{
   public static void main(String x[])
    {
        Sort s1 = new Sort();
        s1.acceptValue(5,2,4,3,1);
        System.out.println("\nDisplay unsorted array");
        s1.display();
        s1.performSort();
        System.out.println("\nDisplay sorted array");
        s1.display();
    }
}
```

If we want to work with variable argument we have some important points or we have some important rules.

---

a) … must be left hand side of variable or right hand side of data type

b) We cannot pass more than variable argument in single function (means cannot pass … with more than one variable in single function)

```
class RuleVarArg
{
        void setValue(int ...x,String ...y)
        {
        }

}
```

c) If we have some simple parameter and variable argument in same function then variable argument must be last parameter in function.

```
class RuleVarArg
{
      void setValue(String studName,int ...subMarks)
      {
         System.out.printf("Name is %s\n",studName);
          for(int i=0; i<subMarks.length; i++)
         {  System.out.printf("%d\t",subMarks[i]);
         }
      }
}
public class SortVarArgApp
{
    public static void main(String x[])
    {
        RuleVarArg rva = new RuleVarArg();
        rva.setValue("ABC",60,60,60,60,60,60);
    }
}
```

Output:

```
C:\Program Files\Java\jdk-23\bin>java SortVarArgApp
Name is ABC
60      60      60      60      60      60
C:\Program Files\Java\jdk-23\bin>
```

If we want to pass infinite data as parameter in class function then we can use POJO class concept.

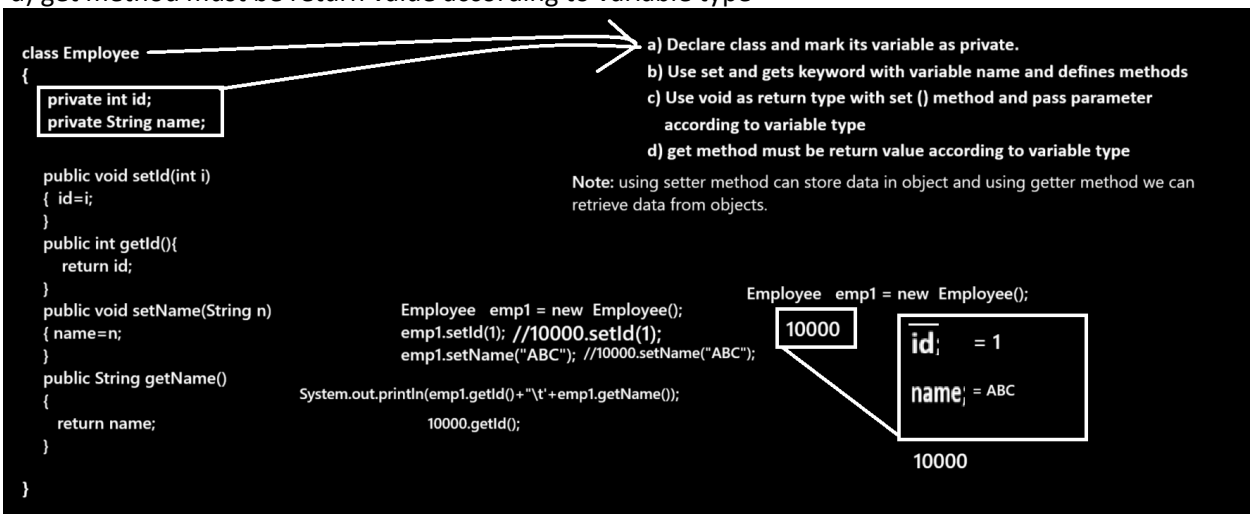
Q. What is POJO class?

POJO stands for plain old java objects called as POJO class basically it is a class which contain setter and getter method and which is used for store data and pass in application from one location to another location.

If we want to work with POJO class we have some standard.

---

a) Declare class and mark its variable as private.
b) Use set and gets keyword with variable name and defines methods
c) Use void as return type with set () method and pass parameter according to variable type
d) get method must be return value according to variable type

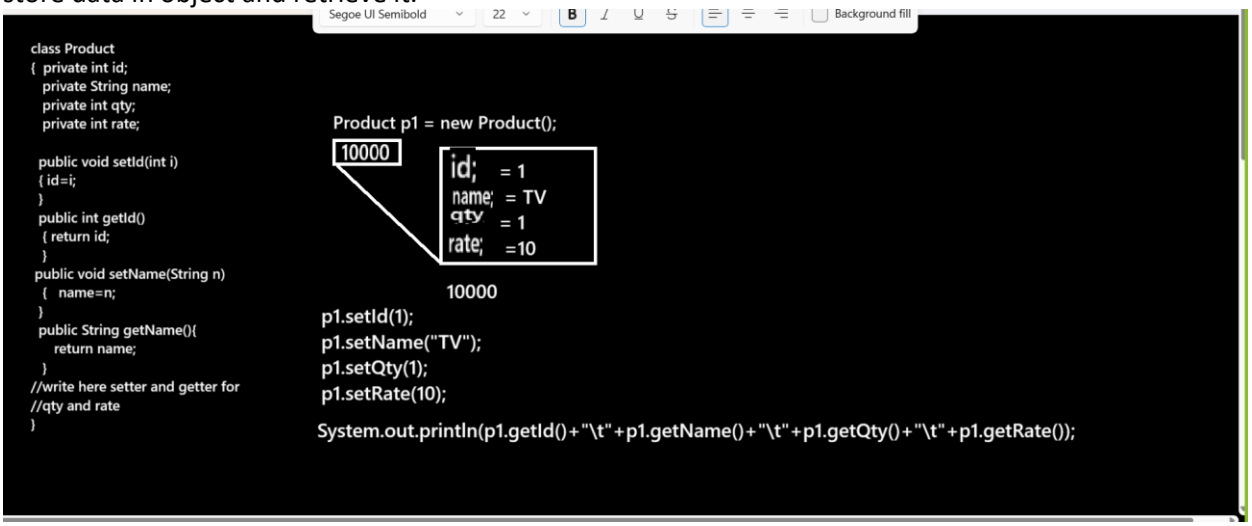


Example with source code

class Employee
{

```
  private int id;
  private String name;

  public void setId(int i)
  { id=i;
  }
  public int getId(){
    return id;
  }
  public void setName(String n)
  { name=n;
  }
  public String getName()
  { return name;
  }
}
public class EmpPOJOAPP
{
   public static void main(String x[])
          {
             Employee emp = new Employee();
                   emp.setId(1);
                   emp.setName("ABC"); //set data in object or store data in object.
                   int empId=emp.getId();  //fetch data from object
                   String name=emp.getName();
                   System.out.println("Employee Id is  "+empId);
                   System.out.println("Employee Name is "+name);
          }
}
```

Example: WAP to create class name as Product with field id,name,qty and rate and create pojo class store data in object and retrieve it.



Example with source code

```
class Product
{  private int id;
   private String name;
   private int qty;
   private int rate;

   public void setId(int i)
   { id=i;
   }
   public int getId()
   { return id;
   }
   public void setName(String n)
   { name=n;
   }
   public String getName()
   { return name;
   }
   public void setQty(int q)
   { qty=q;
   }
   public int getQty()
   { return qty;
   }
   public void setRate(int r)
   { rate=r;
   }
   public int getRate(){
    return rate;
   }
}
public class  ProdPOJOAPPOCT
{  public static void main(String x[])
   {
                        Product p1 = new Product();
                        p1.setId(1);
                        p1.setName("TV");
                        p1.setQty(1);
                        p1.setRate(10);
                        System.out.println("Product Name  : "+p1.getName());
                        System.out.println("Product Id    : "+p1.getId());
                        System.out.println("Product Qty   : "+p1.getQty());
                        System.out.println("Product Rate  : "+p1.getRate());
   }
}
```

Q. Why use POJO class?

According to coding standard if we have function which contains more than three different types of parameter then passing single single parameter is not good approach because of following reason.

a) When we have different types of parameter in function and when we call the function then we need to remember the every parameter type, sequence and it is very complicated when we have large number of parameter in function

b) Must be pass all parameter compulsory if required or not

Example: Suppose consider we have class name as Company with two functions
void setEmployee(String name,int id,String address,String qualification,int prevSal,int expSal,String preCompName,int age,String skillSet): this function can accept the details of employee.
void showEmployee(): this function can display the details of employee

Example with source code

```
class Company
{
  private String empName;
  private String empAddr;
  private String empQual;
  private String empPrevComp;
  private String empSkillSet;
  private int empId;
  private int empAge;
  private int empPrevSal;
  private int empExpSal;

  void setEmployee(String name,int id,String address,String qualification,int prevSal,int expSal,String
prevCompName,int age,String skillSet)
  {  empName=name;
            empAddr=address;
            empQual=qualification;
            empPrevComp=prevCompName;
            empSkillSet=skillSet;
            empId=id;
            empAge=age;
            empPrevSal=prevSal;
            empExpSal=expSal;
  }
  void showEmployeeDetail()
  {
    System.out.println("Employee Name is "+empName);
            System.out.println("Employee Address is  "+empAddr);
            System.out.println("Employee Qualification is "+empQual);
            System.out.println("Employee Previous Company "+empPrevComp);
            System.out.println("Employee SkillSet "+empSkillSet);
            System.out.println("Employee Id is  "+empId);
            System.out.println("Employee Age is "+empAge);
            System.out.println("Employee Previous Salary  "+empPrevSal);
```

```
        System.out.println("Employee Expected Salary is "+empExpSal);
 }
}
public class EmpCompAppOCTSEP
{
  public static void main(String x[])
  {
     Company c=new Company();
          c.setEmployee("RAM",1,"PUNE","BTECH",100000,120000,"TCS",25,"FULL STACK DEVELOPER");
          c.showEmployeeDetail();
  }
}
```

```
C:\Program Files\Java\jdk-23\bin>javac EmpCompAppOCTSEP.java

C:\Program Files\Java\jdk-23\bin>java EmpCompAppOCTSEP
Employee Name is RAM
Employee Address is  PUNE
Employee Qualification is BTECH
Employee Previous Company TCS
Employee SkillSet FULL STACK DEVELOPER
Employee Id is  1
Employee Age is 25
Employee Previous Salary  100000
Employee Expected Salary is 120000

C:\Program Files\Java\jdk-23\bin>
```

if we think about above code then we have function name as setEmployee() which contain different types of data so when we call this function then we need to remember the every parameter type and its proper sequence so when we have 20 or more than 20 parameter then it is very tedious task to developer remember the every parameter type and sequence

We have one limitation is if we want to skip the some parameter from function calling point it is not possible and if we try to skip some parameter then we get compile time error shown in following code.

Example with source code

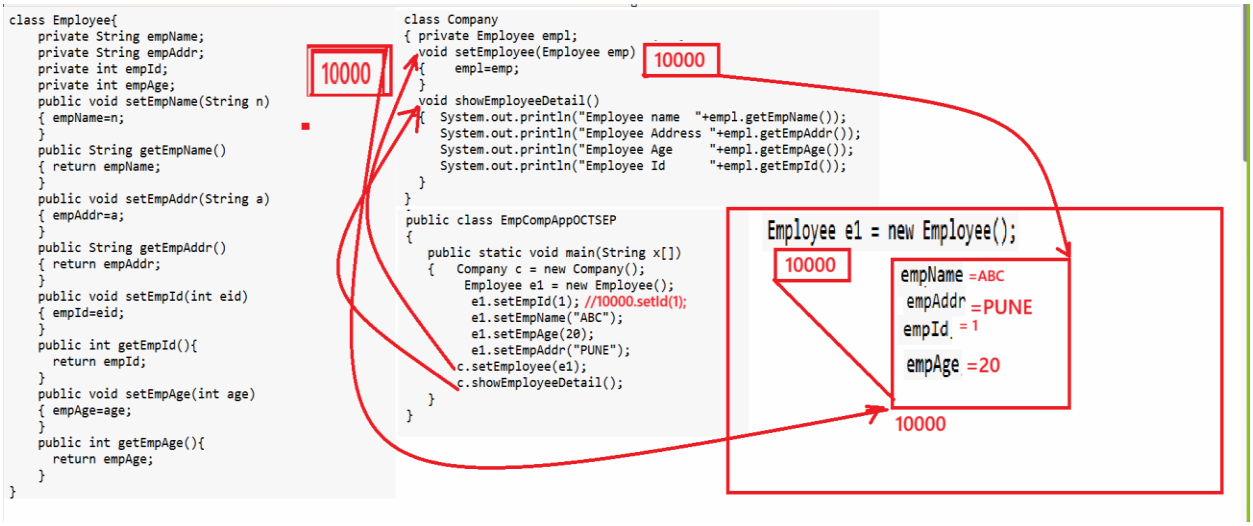
```
class Company
{
 private String empName;
 private String empAddr;
 private String empQual;
 private String empPrevComp;
 private String empSkillSet;
 private int empId;
 private int empAge;
 private int empPrevSal;
 private int empExpSal;

 void setEmployee(String name,int id,String address,String qualification,int prevSal,int expSal,String
prevCompName,int age,String skillSet)
 { empName=name;
         empAddr=address;
         empQual=qualification;
         empPrevComp=prevCompName;
```

```
        empSkillSet=skillSet;
        empId=id;
        empAge=age;
        empPrevSal=prevSal;
        empExpSal=expSal;
 }
 void showEmployeeDetail()
 {
  System.out.println("Employee Name is "+empName);
        System.out.println("Employee Address is  "+empAddr);
        System.out.println("Employee Qualification is "+empQual);
        System.out.println("Employee Previous Company "+empPrevComp);
        System.out.println("Employee SkillSet "+empSkillSet);
        System.out.println("Employee Id is  "+empId);
        System.out.println("Employee Age is "+empAge);
        System.out.println("Employee Previous Salary  "+empPrevSal);
        System.out.println("Employee Expected Salary is "+empExpSal);
 }
}
public class EmpCompAppOCTSEP
{
  public static void main(String x[])
  {
    Company c=new Company();
        c.setEmployee("RAM",1,"PUNE","BTECH",100000,"TCS",25,"FULL STACK DEVELOPER");
        c.showEmployeeDetail();
  }
}
```

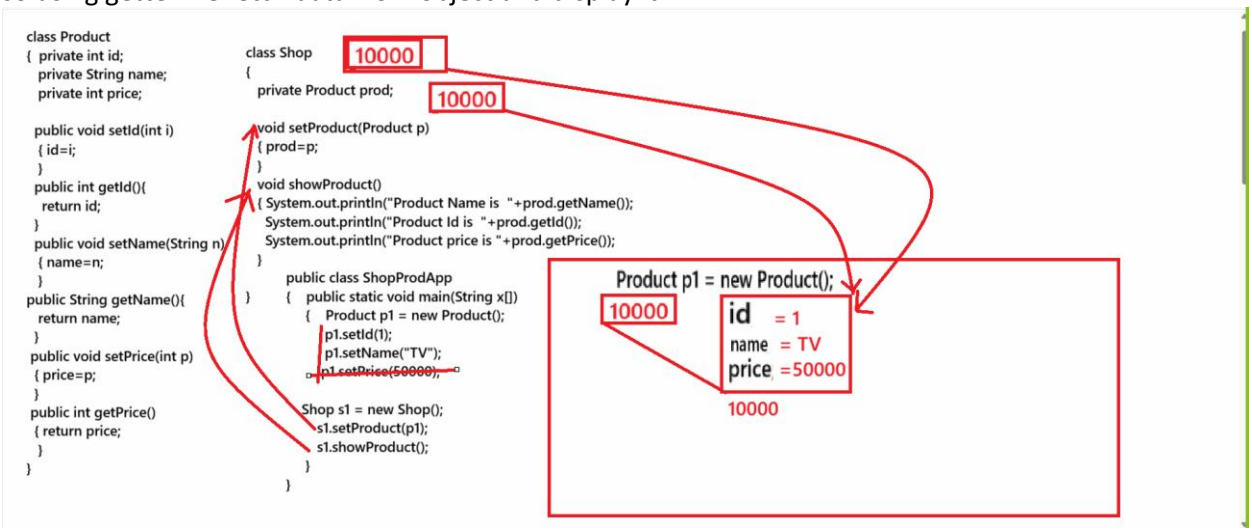
```
C:\Program Files\Java\jdk-23\bin>javac EmpCompAppOCTSEP.java
EmpCompAppOCTSEP.java:42: error: method setEmployee in class Company
cannot be applied to given types;
          c.setEmployee("RAM",1,"PUNE","BTECH",100000,"TCS",25,"FULL
 STACK DEVELOPER");
           ^
  required: String,int,String,String,int,int,String,int,String
  found:    String,int,String,String,int,String,int,String
  reason: actual and formal argument lists differ in length
1 error
```

we get error because we have 9 parameter in setEmployee() function but we pass 8 parameter from calling so we get compile time means we must be pass 9 parameter any how

so if we want to solve this problem java suggest us store all parameter in POJO class and pass POJO class as parameter in setEmployee class



if we think about above code we have Company class which contain method void setEmployee(Employee emp) here we have function name as setEmployee() in this function we pass reference of Employee class and we copy this reference i.e address in empl and if we think about main method we have statement Employee e1 = new Employee() means we create object in memory whose address is 10000 and we store 10000 reference in Employee e1 and using setter method calling we store data in Employee object whose address is 10000 and we have statement c.setEmpoyee(e1) means we pass e1 as reference in Company class function means e1 send 10000 which is address of Employee object in reference variable emp means here emp points to 10000 i.e e1 and again we copy the address of emp in empl means empl also points to 10000 means here we have only one object in memory and three different references points to same object and we have statement c.showEmployeeDetail()
so using getter we fetch data from object and display it.



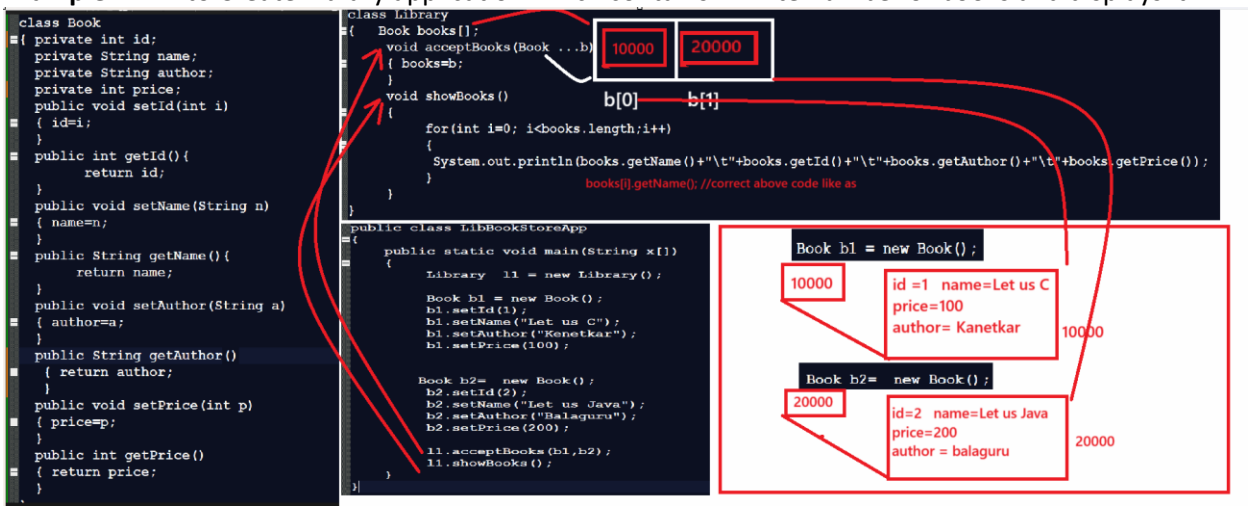
Example POJO class with variable

```
class Product
{
  private int id;
  private String name;
  private int price;
```

```
  public void setId(int i)
   { id=i;
   }
  public int getId(){
    return id;
   }
  public void setName(String n)
   { name=n;
   }
  public String getName()
   {return name;
   }
  public void setPrice(int p)
   { price=p;
   }
  public int getPrice()
   { return price;
   }
}
class Shop
{
   private Product []prod;
   public void setProduct(Product ...pr)
   {  prod=pr;
   }
   public void showProduct()
   {
     for(int i=0;i <prod.length;i++)
     { System.out.println(prod[i].getId()+"\t"+prod[i].getName()+"\t"+prod[i].getPrice());
     }
    }
}
public class ShopProdApp
{
    public static void main(String x[])
    {
       Product p1 = new Product();
                  p1.setId(1);
                  p1.setName("TV");
                  p1.setPrice(50000);

            Product p2 = new Product();
               p2.setId(2);
               p2.setName("MOBILE");
               p2.setPrice(60000);

             Product p3 = new Product();
```

```
        p3.setId(3);
        p3.setName("FREEZE");
        p3.setPrice(40000);

       Shop s1 = new Shop();
       s1.setProduct (p1, p2,p3);
       s1.showProduct ();
  }
}
```

Example: WAP to Create Library application which contains infinite number of books and displays it.



Example with source code

```
class Book
{ private int id;
  private String name;
  private String author;
  private int price;
  public void setId(int i)
  { id=i;
  }
  public int getId(){
              return id;
  }
  public void setName(String n)
  { name=n;
  }
  public String getName(){
             return name;
  }
  public void setAuthor(String a)
  { author=a;
  }
  public String getAuthor()
```

```
  { return author;
  }
 public void setPrice(int p)
 { price=p;
 }
 public int getPrice()
 { return price;
 }
}
class Library
{   Book books[];
        void acceptBooks(Book ...b)
        { books=b;
        }
        void showBooks()
        {
                for(int i=0; i<books.length;i++)
                {

System.out.println(books[i].getName()+"\t"+books[i].getId()+"\t"+books[i].getAuthor()+"\t"+books[i].ge
tPrice());
                }
        }
}
public class LibBookStoreApp
{
   public static void main(String x[])
        {
                Library  l1 = new Library();

                Book b1 = new Book();
                b1.setId(1);
                b1.setName("Let us C");
                b1.setAuthor("Kenetkar");
                b1.setPrice(100);


               Book b2=  new Book();
                b2.setId(2);
                b2.setName("Let us Java");
                b2.setAuthor("Balaguru");
                b2.setPrice(200);

                l1.acceptBooks(b1,b2);
                l1.showBooks();
        }
}
```

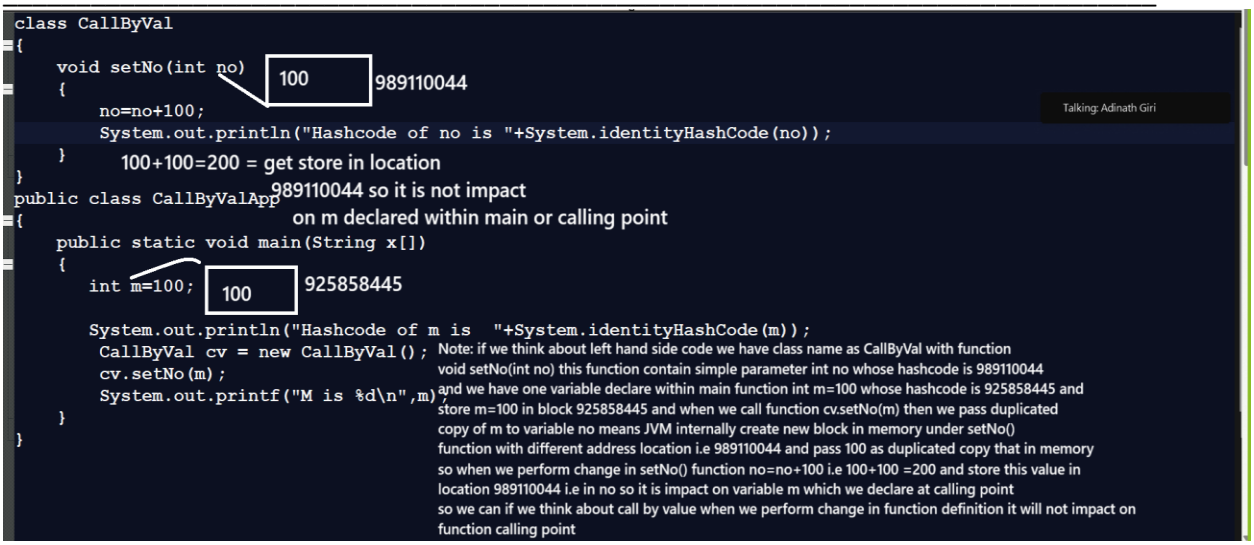
Output

```
C:\Program Files\Java\jdk-23\bin>javac LibBookStoreApp.java

C:\Program Files\Java\jdk-23\bin>java LibBookStoreApp
Let us C        1       Kenetkar        100
Let us Java     2       Balaguru        200

C:\Program Files\Java\jdk-23\bin>
```

Call by Value and Call by reference

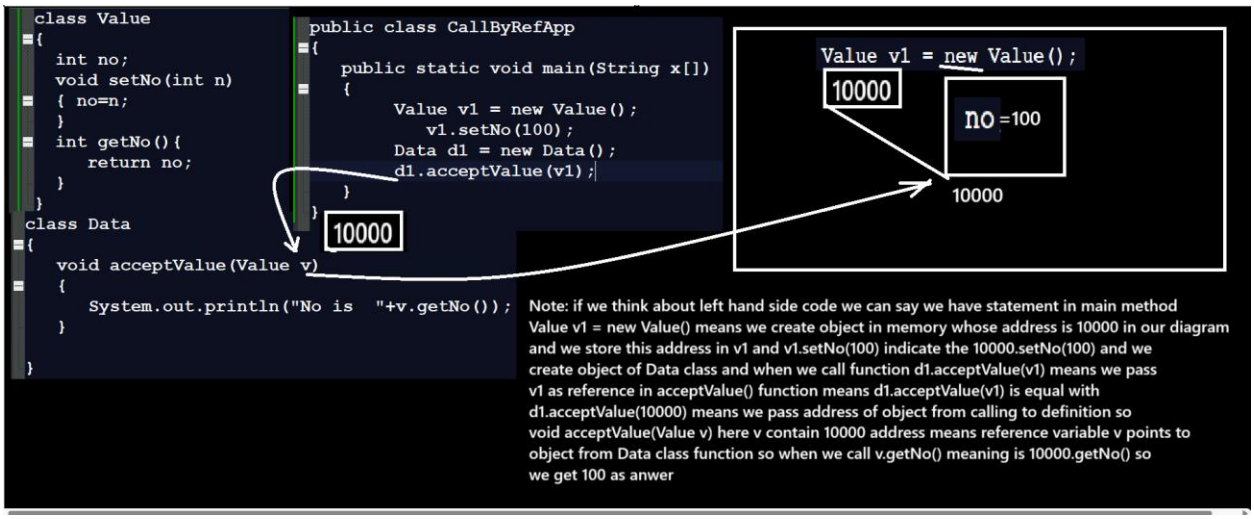
Call by value means when we pass direct value of variable from function calling point to function definition called as call by value and when we pass address of variable from function calling point to function definition called as call by reference.
if we want to identify the call by value and call by reference in java we have simple logic
When we pass any variable without using new keyword called as call by value and when we pass any variable using new keyword called as call by reference.
If we think about call by value when we perform change in function definition then it never applicable on function calling point or not impact on function calling point and if we think about call by reference when we perform change in function definition it will impact on function calling point.

Example of call by value

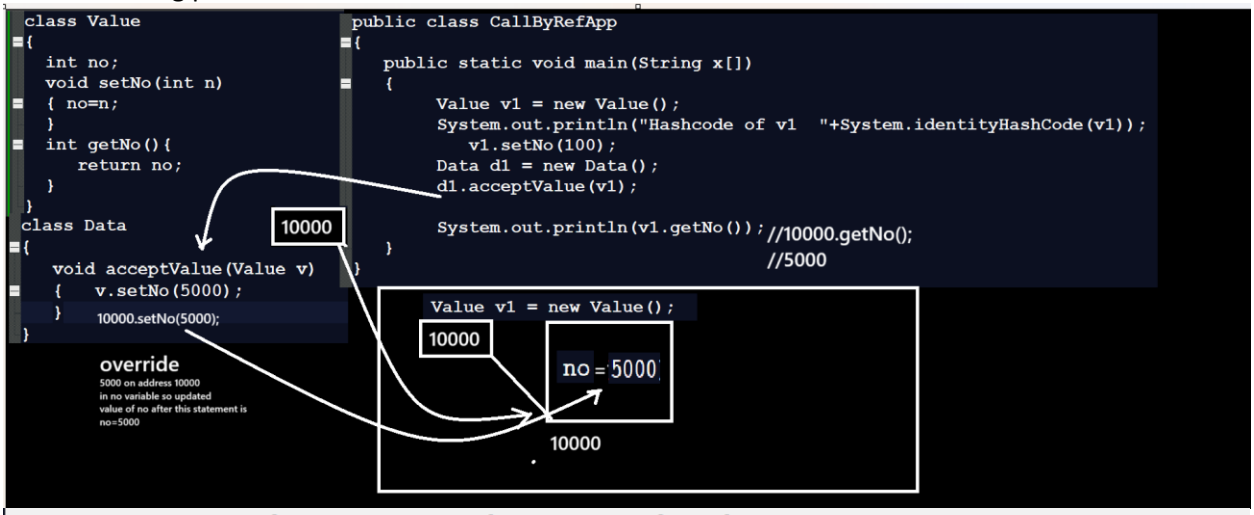


Example of call by reference

When we pass address from function calling point to function definition called as call by reference.
When we pass any reference variable using new keyword from function calling to definition then we can say it is implementation of call by reference.



Note: if we think about call by reference if we perform any change in function definition it will reflect on function calling point.




```
C:\Program Files\Java\jdk-23\bin>java CallByRefApp
Hashcode of v1  798154996
5000
```

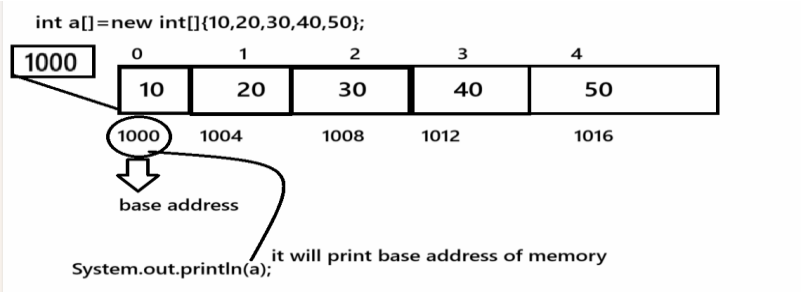
if we think about above code we have statement Value v1=new Value() in memory means here we have object whose address is 10000 and we store this address in reference variable v and we have statement v1.setNo(100) means 10000.setNo(100) means we store 100 value in no variable which is present in 10000 memory block and we have statement d1.acceptValue(v) means d1.acceptValue(10000) means we pass 10000 memory from function calling point to function definition point means reference variable Value v points to object from Data class function whose address is 10000 so acceptValue() function we have statement v.setNo(5000) means 1000.setNo(5000) means we pass 5000 value to no variable which is present in block or object whose address is 10000 so 5000 get override on 100 so the latest of no is 5000 from object 10000 location and if we think about main method we have statement v1.getNo()so we updated value i.e. 5000

So we can say in call by reference when we perform any change in function definition it will reflect on function calling point.

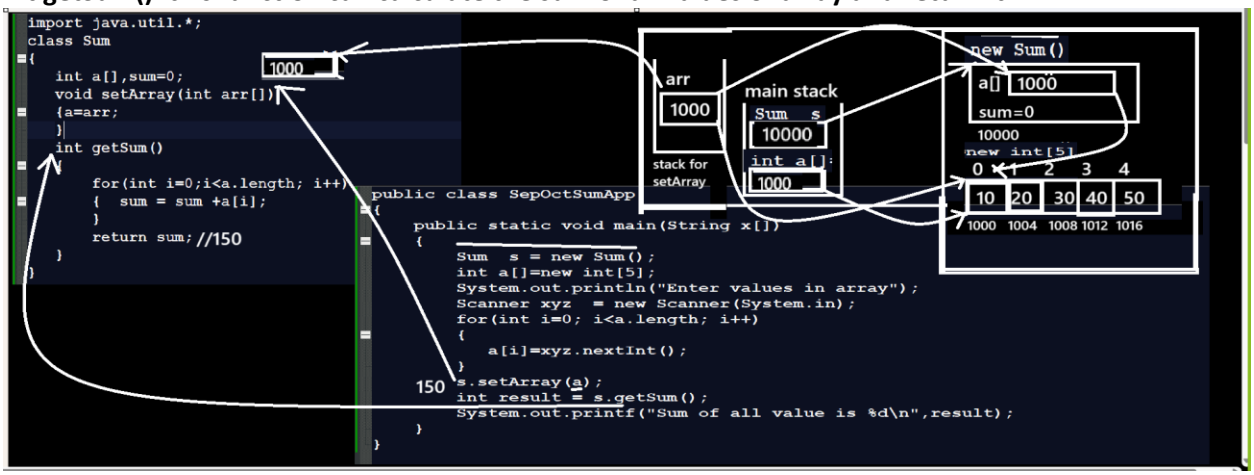
How to pass array as parameter in class function

If we want to pass array as parameter in function we need to pass base address of array in class function
0^{th} index address called as base address

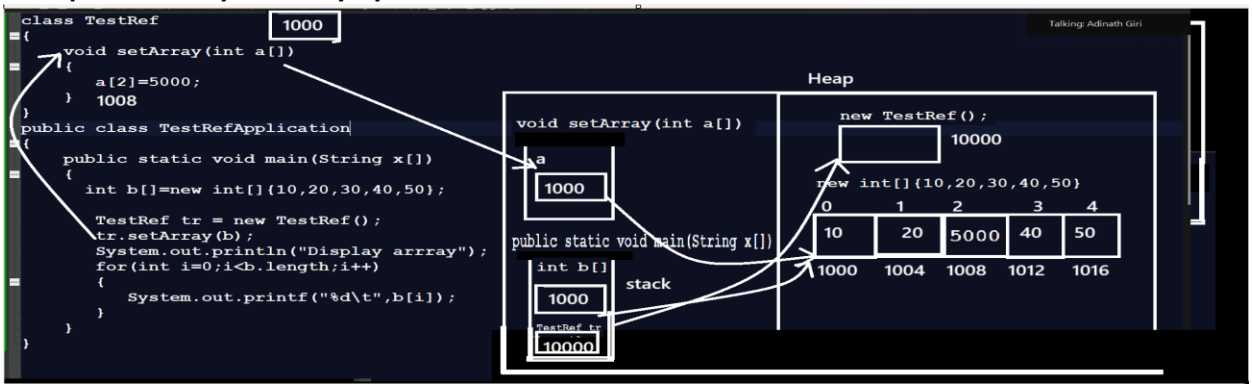
Note: if we use array without subscript or bracket then we get base address of array.

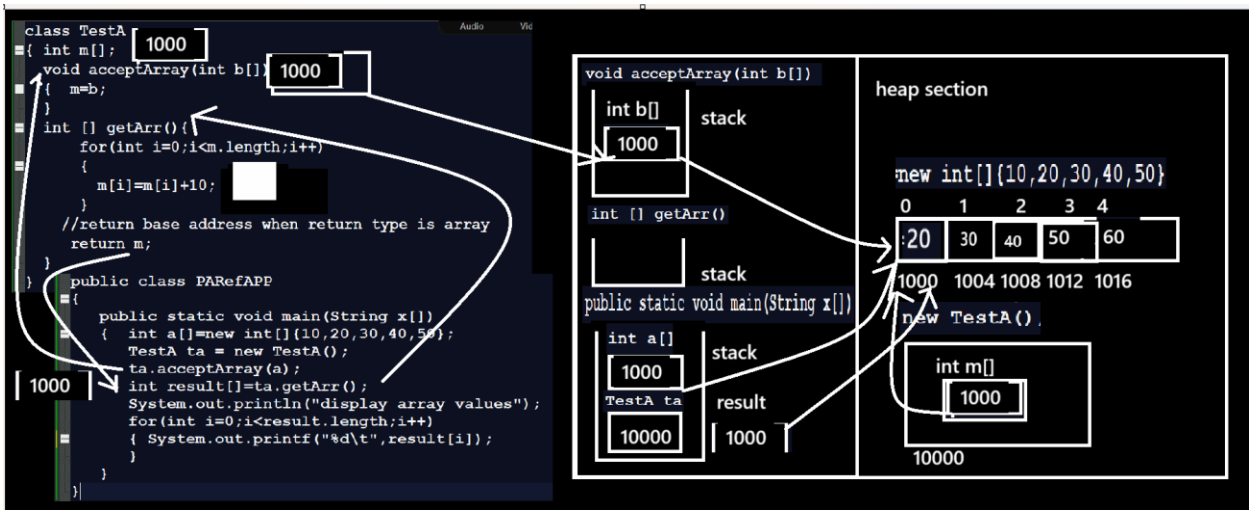


Example: WAP to create class name as Sum with two methods
void setArray(int a[]): this function can accept array as parameter
int getSum(): this function can calculate the sum of all values of array and return it.



Example to modify the array by reference





Array of object concept

_____

Q. What is array of objects?

_____

If you think about array of objects we can store more than one object data using single name reference called as array of objects.

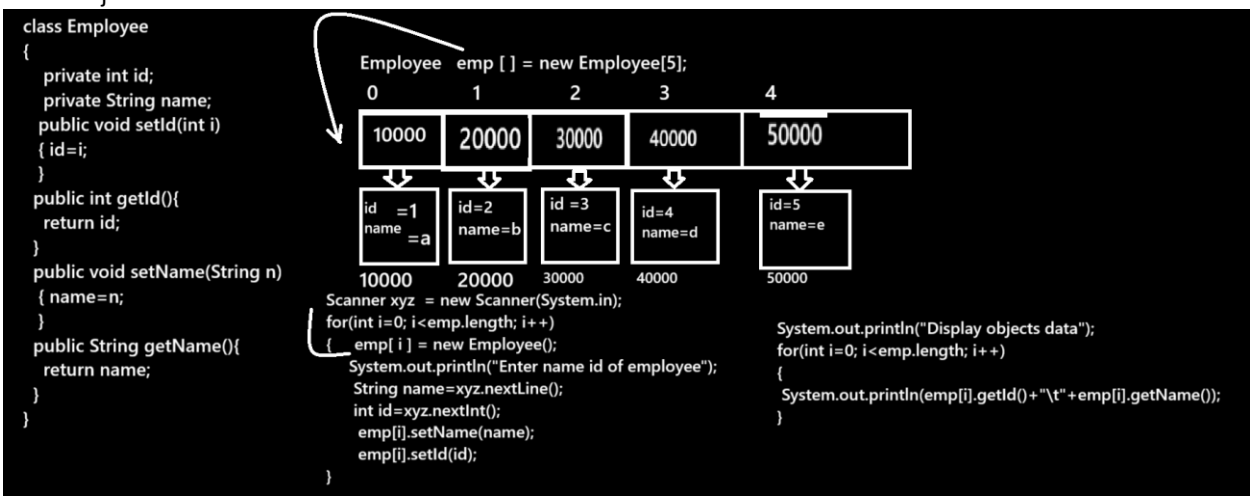
Example: Consider we have Employee class and we want to create 5 employee objects and store data in it and display it.

How to create array of objects in JAVA?

_____

Syntax: classname ref [] = new classname[size]; //array of references

```
for(int i=0; i<ref.length; i++)
{
    ref[i] = new classname(); //array of objects
}
```



Example:

```
import java.util.*;
class Employee
```

```
{
   private int id;
          private String name;

          public void setId(int i)
          { id=i;
          }
          public int getId()
          { return id;
          }
          public void setName(String n)
          { name=n;
          }
          public String getName()
          { return name;
          }
}
public class EmployeeApp
{
  public static void main(String x[])
  {
     Employee emp[]=new Employee[5]; //array of reference
            Scanner xyz = new Scanner(System.in);
            for(int i=0;i<emp.length;i++)
            {
                    emp[i]=new Employee();//array of objects
                          System.out.println("Enter name and id of employee");
                          String name=xyz.nextLine();
                          int id=xyz.nextInt();
                          emp[i].setName(name);
                          emp[i].setId(id);
                          xyz.nextLine();
            }
            System.out.println("Display employee data");
            for(int i=0; i<emp.length;i++)
            {
                    System.out.println(emp[i].getId()+"\t"+emp[i].getName());
            }

  }
}
```

Example:

```
import java.util.*;
class Product
{ private int id;
  private String name;
  private int price;
```

```
 public void setId(int i)
 {  id=i;
 }
 public int getId(){
   return id;
 }
 public void setName(String n)
 { name=n;
 }
 public String getName(){
   return name;
 }
 public void setPrice(int p){
  price=p;
 }
 public int getPrice(){
  return price;
 }
}
class Shop
{
   void searchProduct(Product p[],int id)
          {   int index=-1;
             for(int i=0; i<p.length;i++)
                    {
                      int prodId=p[i].getId();
                      if(prodId==id)
                      {  index=i;
                                break;
                      }
                    }
                    if(index!=-1)
                    { System.out.println(p[index].getId()+"\t"+p[index].getName()+"\t"+p[index].getPrice());
                    }
                    else{
                     System.out.println("Product Not Found");
                    }
          }
}
public class ProdArrApp
{   public static void main(String ...x)
          { Scanner xyz  = new Scanner(System.in);
             Shop s1 = new Shop();
                    Product prod[]=new Product[5];// array of reference.
                    for(int i=0; i<prod.length; i++)
                    {
                       prod[i]=new Product();
                              System.out.println("Enter name id and price of product");
```

```
                String name=xyz.nextLine();
                int id=xyz.nextInt();
                int price=xyz.nextInt();
                prod[i].setName(name);
                prod[i].setId(id);
                prod[i].setPrice(price);
                xyz.nextLine();
            }
            System.out.println("Enter id for search product");
            int productId=xyz.nextInt();
            s1.searchProduct(prod,productId);
        }
}
```

How to set path of JDK\bin folder

Before that we should have to know why we need to set path of JDK\bin

Q. Why we need to set path of JDK\bin folder?

By default we save our java files in JDK\bin folder and compile from bin and run from bin folder so the limitation is when we uninstall JDK from your system then your code may be lost as well as we cannot manage the proper java file when we have larger number of classes so better you can save your java file other location in system according to choice and compile your code from different location except bin as per your choice and run it.

How to set the JDK\bin path?

---

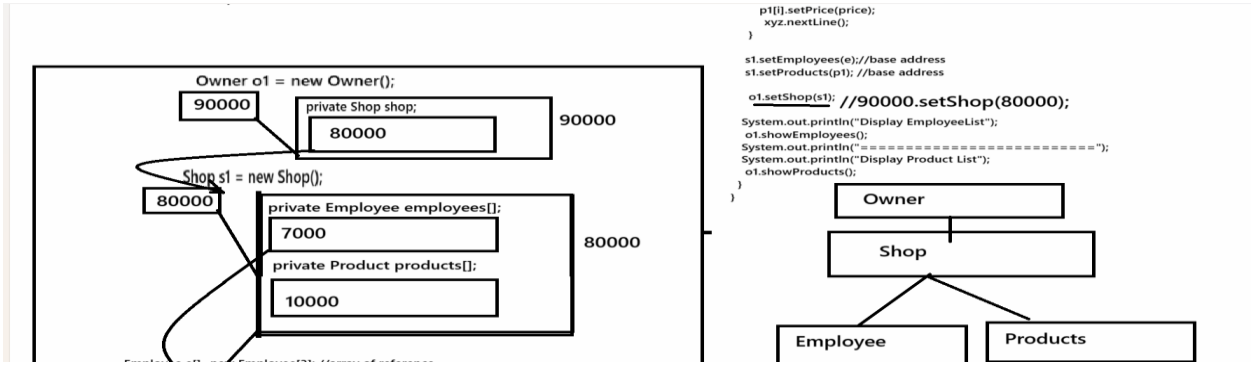
search ----view advanced system setting ---- environmental variable ---- select system variable ----- select path variable ---click on edit button ----click on new button ----copy path of JDK\bin folder where we install it ---- paste at last in window of edit environmental variable ----- click on ok and ok

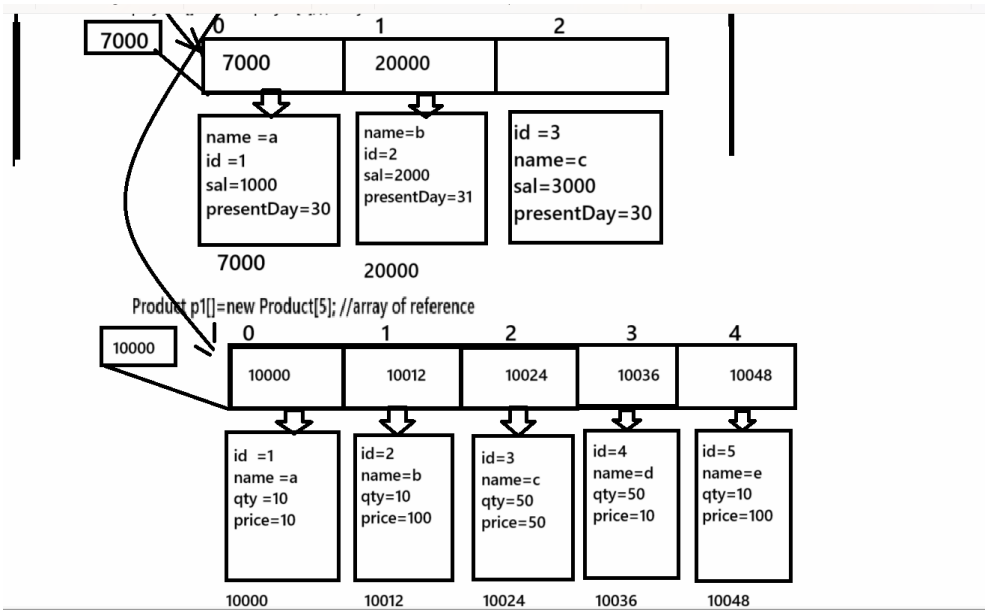
Note: if we want to check your path is set or not then we use following steps.

a) Open command prompt and type javac command if we get following options we can consider your path is set successfully.

```
C:\Users\Admin>javac
Usage: javac <options> <source files>
where possible options include:
  @<filename>                  Read options and filenames from file
  -Akey[=value]                Options to pass to annotation processors
  --add-modules <module>(,<module>)*
        Root modules to resolve in addition to the initial modules,
        or all modules on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path>, -bootclasspath <path>
        Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
        Specify where to find user class files and annotation processors
  -d <directory>               Specify where to place generated class files
  -deprecation
```

Example: InventoryManagmentApplication





Example with source code

Product.java

```
public class Product
{
   private int id;
```

```
    private String name;
    private int price;
    private int qty;
    public void setId(int id)
    { this.id=id;
    }
    public int getId()
     { return id;
     }
    public void setName(String name)
    { this.name=name;
    }
    public String getName()
    { return name;
    }
    public void setPrice(int price)
    { this.price=price;
    }
    public int getPrice()
    { return price;
    }
    public void setQty(int qty)
    { this.qty=qty;
    }
    public int getQty()
    { return qty;
    }

}
```

Employee.java

```
public class Employee
{
    private int id;
    private String name;
    private int sal;
    private int presentDay;

    public void setId(int id)
    { this.id=id;
    }
    public int getId()
    { return id;
    }
    public void setName(String name)
     { this.name=name;
     }
    public String getName()
    { return name;
```

```
   }
   public void setSal(int sal)
    { this.sal=sal;
    }
  public int getSal()
   {return sal;
    }
  public void setPresentDay(int presentDay)
   {this.presentDay=presentDay;
   }
  public int getPresentDay()
   { return presentDay;
   }
}
```

Shop.java

```
public class Shop
{
  private Product []products;
  private Employee []employees;

  public void setProducts(Product products[])
   {  this.products=products;
   }
  public Product[] getProducts()
  { return products;
  }
  public void setEmployees(Employee employees[])
  {   this.employees=employees;
  }
  public Employee[] getEmployees()
   { return employees;
   }
  public Employee[] getEmployeeRecordWithBonusSalary()
   {
          for(int i=0; i<employees.length; i++)
          {  int presenty=employees[i].getPresentDay();
             if(presenty>=30)
             {
                    int empCurrSal=employees[i].getSal();
                    int bonusSal=empCurrSal*10/100;
                    empCurrSal=empCurrSal+bonusSal;
                    employees[i].setSal(empCurrSal);
        }
          }
          return employees;
   }
  public Product[] getBillWithDiscount()
```

```
  {
          for(Product p1:products)
     {
       int qty=p1.getQty();
            int rate=p1.getPrice();
            int total=qty*rate;
            if(total>500)
            {
              int discount=total*20/100; //120/1
              total=total-discount; //600-120

            }
           p1.setTotal(total);
     }
          return products;
  }
}
```

Owner.java

```
public class Owner
{
   private Shop shop;
   public void setShop(Shop shop)
   { this.shop=shop;
   }
   public Shop getShop()
   { return shop;
   }

public void showEmployees()
  {  Employee employees[]=shop.getEmployees();
    for(int i=0;i<employees.length;i++)
    {
      System.out.println(employees[i].getId()+"\t"+employees[i].getName()+"\t"+
    employees[i].getSal()+"\t"+employees[i].getPresentDay());
    }
  }
  public void showProducts()
  { Product products[]=shop.getProducts();
    for(Product p:products) //for(int i=0;i<products.length;i++)
    {
      System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getPrice()+"\t"+p.getQty());
    }
  }

  public void creditSal()
   {
          Employee emp[]=shop.getEmployeeRecordWithBonusSalary();
           for(Employee e:emp)
```

```
            {
            System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
        }
    }
   public void showProdDetailsWithDiscount()
   {  Product p[]=shop.getBillWithDiscount();
            for(Product p1:p)
            {
            System.out.println(p1.getId()+"\t"+p1.getName()+"\t"+p1.getQty()+"\t"+p1.getPrice()+"\t"+
            (p1.getQty()*p1.getPrice())+"\t"+p1.getTotal());
            }

   }
}
```

InventoryManagmentApp.java

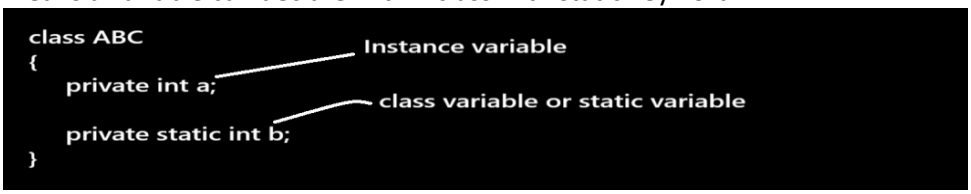
```
import java.util.*;
public class InventoryManagmentApp
{
  public static void main(String x[])
  {  Scanner xyz =new Scanner(System.in);

    Owner o1 = new Owner();

    Shop s1 = new Shop();

    Employee e[]=new Employee[3]; //array of reference.
    for(int i=0; i<e.length;i++)
    {
        e[i]=new Employee(); //array of objects
            System.out.println("Enter name id and salary as well as number of present days");
            String name=xyz.nextLine();
            int id=xyz.nextInt();
            int sal=xyz.nextInt();
            int presentDay=xyz.nextInt();
            e[i].setName(name);
            e[i].setId(id);
            e[i].setSal(sal);
            e[i].setPresentDay(presentDay);
      xyz.nextLine();
    }
    Product p1[]=new Product[5]; //array of reference
    for(int i=0; i<p1.length;i++)
    {
      p1[i]=new Product(); //array of object
            System.out.println("Enter name id price and quantity");
            String name=xyz.nextLine();
            int id=xyz.nextInt();
```

```
            int price=xyz.nextInt();
            int qty=xyz.nextInt();
            p1[i].setName(name);
        p1[i].setId(id);
        p1[i].setQty(qty);
        p1[i].setPrice(price);
            xyz.nextLine();
     }

    s1.setEmployees(e);//base address
    s1.setProducts(p1); //base address

     o1.setShop(s1);
   System.out.println("Display EmployeeList");
    o1.showEmployees();
   System.out.println("==========================");
   System.out.println("Display Product List");
     o1.showProducts();
   System.out.println("Show Employee Record with Bonus Salary");
     o1.creditSal();
   System.out.println("Show products bill with discount amount");
    o1.showProdDetailsWithDiscount();
  }
}
```

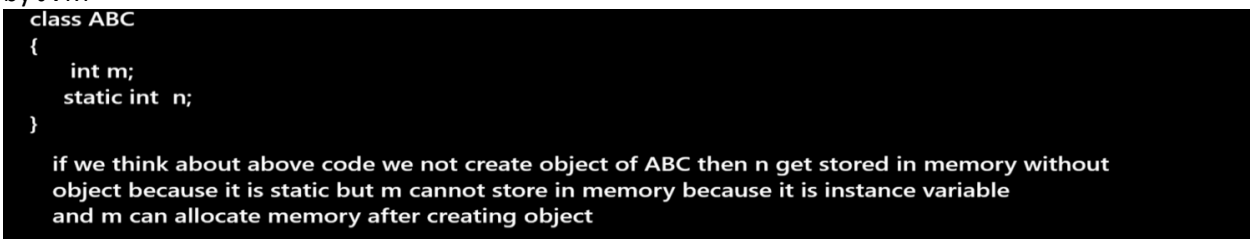
Instance Variable and Static Variable or class level variable

---

a) Instance variable means variable we can declare within class without static keyword and class variable means a variable can declare within class with static keyword



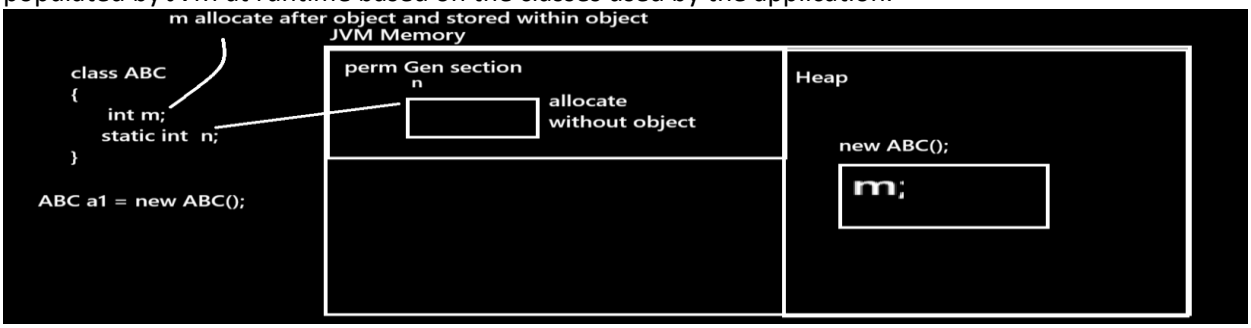
b) Instance variable allocated in memory after creating object of class and static variable allocate in memory before creating object of class means allocate memory at the time of class loaded in memory by JVM



C) Instance variable stored within object and static variable stored within Permanent Generation section of memory or Meta section After JDK 1.8

Q. What is permanent Generation section?

Permanent Generation section contains application Meta data required by JVM to describe the classes and methods used in the application means Perm gen section contain class meta data like as class name, its method names,method return type, variable names and variable type as well as it provide space static members without creating object etc . Perm Gen section is not part of heap memory it is populated by JVM at runtime based on the classes used by the application.



Q. How we can prove static variable allocate before object?

If we want to check static variable allocate before object we can access it using class name
Means we can say we can access static variable without object with the help of classname but we cannot access the instance variable using class name and if we try to access instance variable using class name we get compile time error means we can say instance variable not present in memory

```
class StatVar
{
   static int n=100;
   int m=200;
}
public class StatVarApp
{
    public static void main(String x[])
    {
     System.out.printf("N  = %d\n",StatVar.n);

    }
}
```

Note: we not create object of StatVar class but we use variable n using class name because it is static

Talking: Adinath Giri

```
Command Prompt

D:\sep2024\corejava\31-12-2024>javac StatVarApp.java

D:\sep2024\corejava\31-12-2024>java StatVarApp
N  = 100

D:\sep2024\corejava\31-12-2024>
```

```
class StatVar
{
   static int n=100;
   int m=200;
}
public class StatVarApp
{
    public static void main(String x[])
    {
     System.out.printf("N  = %d\n",StatVar.n);
      System.out.printf("M  = %d\n",StatVar.m);
    }
}
```

Note: we get compile time error because we try to access instance variable m using class name and it is possible so if we want to resolve this problem we can create object of the class and access the instance variable means we can say instance variable allocate memory after creating object of class.

Talking: Adinath Giri

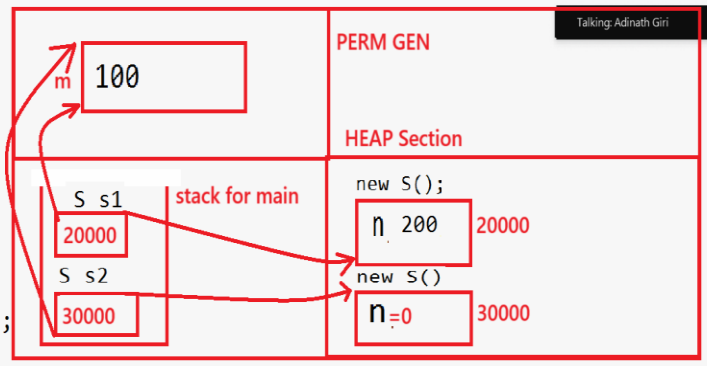
```
Command Prompt

D:\sep2024\corejava\31-12-2024>javac StatVarApp.java
StatVarApp.java:11: error: non-static variable m cannot be referenced from a static context
      System.out.printf("M  = %d\n",StatVar.m);
                                    ^
1 error

D:\sep2024\corejava\31-12-2024>
```

```
class StatVar
{
   static int n=100;
   int m=200;
}
public class StatVarApp
{
    public static void main(String x[])
    {
     System.out.printf("N  = %d\n",StatVar.n);
      System.out.printf("M  = %d\n",new StatVar().m);
    }
}
```

```
D:\sep2024\corejava\31-12-2024>javac StatVarApp.java

D:\sep2024\corejava\31-12-2024>java StatVarApp
N  = 100
M  = 200

D:\sep2024\corejava\31-12-2024>
```

D) static variable share its common copy between more than one object of same class and instance variable share its separate copy for every object.

```
class S
{
  static int m;
  int n;
}
public class SAPP
{ public static void main(String x[])
    {
        S s1= new S();
        S s2= new S();
 s1.m=100;
      s1.n=200;
System.out.println("With First Object");

  System.out.printf("M =%d\tN=%d\n",s1.m,s1.n);
 System.out.println("With Second Object");
 System.out.printf("\nM=%d\tN=%d\n",s2.m,s2.n);
    }
}
```

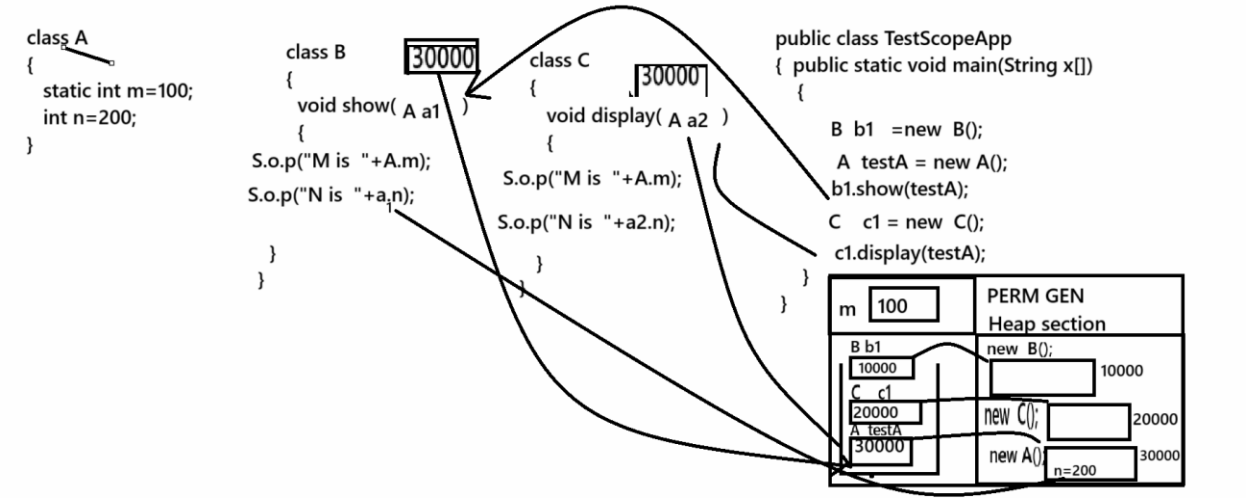


Output:
With First Object
M= 100 N=200
With Second object
M=100 N=0

if we think about above code we have class s with two variable static int m and int n here static variable m or class variable allocated in meta section of memory and if we think about main method we have two objects S s1 = new S() means we create object of class in heap section of memory and we store its reference in s1 i.e 20000 according to our example and we have one more object name as S s2 = new S() means here we create new object in memory whose address is 30000 and we store this address in s2 and we have statement s1.m=100 i.e 20000.m=100 here object first access the static variable and store 100 value in it and again we have statement s1.n=200 i.e 20000.n=200 means here first object access variable n but from own block and again we have statement for display s1.m and s2.n so we get output M=100 and N=200 and we have one more statement s2.m means 30000.m so we access static variable m by using second objet and static variable updated value is 100 and we have statement s2.n means 30000.n means we access variable n using second object but from own block and we not initialize the value of variable n present in second object so its default value is 0 so second time we get answer M=100 and N=0 so we can say if we think static variable it is commonly share by multiple object of same class shown in following output.

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

C:\Program Files\Java\jdk1.8.0_291\bin>java SAPP
With First Object
M =100  N=200
With Second Object

M=100   N=0

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

E) Static variable has global space in application means we can initialize static variable only once in application and we can use it anywhere in program without its object and if we want to use instance variable then we need to pass reference of its object or create new object.



F) Static variable life is present whenever application running in memory and instance variable life is present whenever objects running memory.

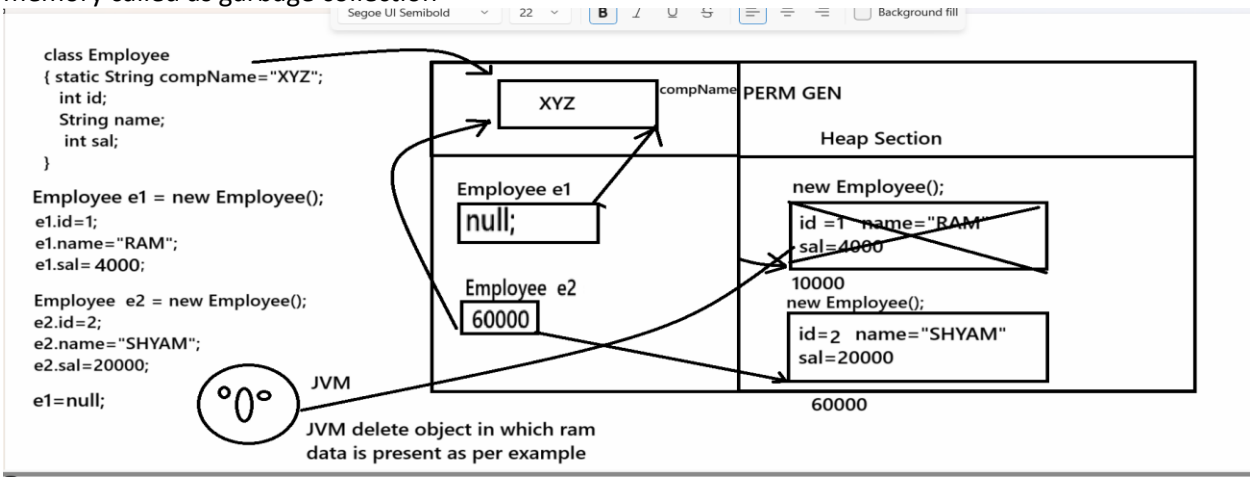
Means if we think about above statement we can say static variable is dependent on program execution and instance variable life is dependent on object.

Note: if we want to calculate the life of instance variable we need to know the life of objects.

Q. Whenever object present in memory?

_____

if object use by particular reference then object present in memory if object not use any reference in code then JVM automatically delete the memory of object or delete object from heap section of memory called as garbage collection



G) static and instance variable has some default values provided by JAVA as per the their data type.

Data Type	Variable name
int	0
Float	0.0
Double	0.0
Long	0
Short	0
Byte	0
Boolean	False
String	Null
Char	blank

```
class D
{  boolean b;
   char ch;
   String s;
   int d;
   void show()
   { System.out.println("Boolean default value "+b);
     System.out.println("String default value "+s);
     System.out.println("Integer Default value "+d);
     System.out.println("Char Default value  "+ch);
   }
}
public class DValApp
{
   public static void main(String x[])
   {
        new D().show();
   }
}
```

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

C:\Program Files\Java\jdk1.8.0_291\bin>java DValApp
Boolean default value false
String default value null
Integer Default value 0
Char Default value  0

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

static function

static function means function can call without object means can call by using class name

```
class D
{  static void show()
   { System.out.println("I am static function");
   }
}
public class DValApp
{
   public static void main(String x[])
   {
         D.show();
   }
}
```

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

C:\Program Files\Java\jdk1.8.0_291\bin>java DValApp
I am static function

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

Important points related with static function?

---

a) We cannot use non static variable/instance variable in static function: instance variable can allocate memory after creating object of class but static function can allocate memory before creating object of class and if we use the instance variable in static function there is possibility instance variable may be not present in memory so instance variable not allowed in static function.


```
class A
{  int no;
  static void show()
   {  System.out.println(no);
   }
}
public class StatAAPP
{
    public static void main(String x[])
    {
         A.show();
    }
}
```

```
Command Prompt

C:\Program Files\Java\jdk1.8.0_291\bin>javac StatAAPP.java
StatAAPP.java:4: error: non-static variable no cannot be referenced from a static cont
ext
   {  System.out.println(no);
                         ^
1 error

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

B) Non static function cannot call from static function or we need to need to create object for call function

```
public class StatAAPP
{
    public static void main(String x[])
    {
          show();
    }
    public  void show()
    { System.out.println("I am show function");
    }
}
```

Note: if we think about left hand side code we have show() is non static function and we try to call it from static block or static function it is not possible by rule so we get compile time error.

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StatAAPP.java
StatAAPP.java:6: error: non-static method show() cannot be referenced from a static co
ntext
          show();
          ^
1 error

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

So if we want to avoid above problem we have two ways

a) Define show as static and call it from static function

```
public class StatAAPP
{
    public static void main(String x[])
    {
          show();
    }
    public static  void show()
    { System.out.println("I am show function");
    }
}
```

```
Command Prompt

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

C:\Program Files\Java\jdk1.8.0_291\bin>java StatAAPP
I am show function

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

b) Create object of class and call show function from static function

```
public class StatAAPP
{
    public static void main(String x[])
    {
          new StatAAPP().show();
    }
    public  void show()
    { System.out.println("I am show function");
    }
}
```

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

C:\Program Files\Java\jdk1.8.0_291\bin>java StatAAPP
I am show function

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

Note: static variable can use in non static function but non static variable cannot use in static function.

Local Variable

Local variable means declared within block or function called as local variable shown in following diagram.

```
class L
{
  static int m; // class variable
   int n ; //instance variable

  void show(int y)        Local variable
  { int z;   //local variable
  }
}
```

Some important points related with local variable

---

a) Local variable cannot access outside of his block and if we try to access it we get compile time error.

```
class L
{
   void setValue(int x)
   {

   }
   void show()
   { System.out.printf("X is %d\n",x);
   }
}
public class StatAAPP
{
    public static void main(String x[])
    {

    }

}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StatAAPP.java
StatAAPP.java:7: error: cannot find symbol
   { System.out.printf("X is %d\n",x);
                                   ^
  symbol:   variable x
  location: class L
1 error

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

B) local variable cannot declare as static

```
class L
{
   void setValue(static int x)
   {
   }

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

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StatAAPP.java
StatAAPP.java:3: error: modifier static not allowed here
   void setValue(static int x)
                 ^
1 error

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

Note: local variable cannot declare as static there is some reasons

1. memory allocation: static variable allocate before object and local variable allocate after function call

2. scope and life: life of static variable present till program in execution as well as scope of static variable present globally in application and local variable present in within is block and local live in program till its block in execution when function block terminate then local variable get destroyed.

3. storage space: static variable store in permanent generation section of memory and local variable store in stack of memory

so we can say behaviour of static keyword and local variable completely opposite of each other so we cannot declare local variable as static

C) We cannot use any access specifier on local variable like as private,public,protected etc

```
class L
{
   void setValue(private int x)
   {
   }

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

Note: access specifier is used for provide security to class member but local variable not need to provide security they already secure and access specifier specially design for apply restriction on class and its member but local variable is not part of class directly so we cannot use access specifier with local variable

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StatAAPP.java
StatAAPP.java:3: error: modifier private not allowed here
   void setValue(private int x)
                 ^
1 error

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

D) If local variable name and instance variable name same then instance variable not access in block where local variable name is same.

```
class SQ
{
   int no;
   void setValue(int no)
   {  no = no;
   }
   void showSquare()
   { System.out.printf("Square is %d\n",no*no);
   }
}
public class StatAAPP
{   public static void main(String x[])
    {
      SQ s1 = new SQ();
       s1.setValue(10);
       s1.showSquare();
    }
}
```

10

Note: if we think about this code we get output square is 0 because we have function which contain void setValue(int no) means local variable no and we have instance variable name as no means here we have local variable name instance variable name is same so if we think setValue(int no) function here instance variable not use because local variable assign self value not allow to enter instance variable in setValue() function and if we think about showSquare() we have statement no*no here we use instance variable so instance variable not copied value from local variable so the default of instance variable is 0 so we get answer Square is 0 shown in following screen shot

Note: if we want to solve this problem we have this reference.

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

C:\Program Files\Java\jdk1.8.0_291\bin>java StatAAPP
Square is 0

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

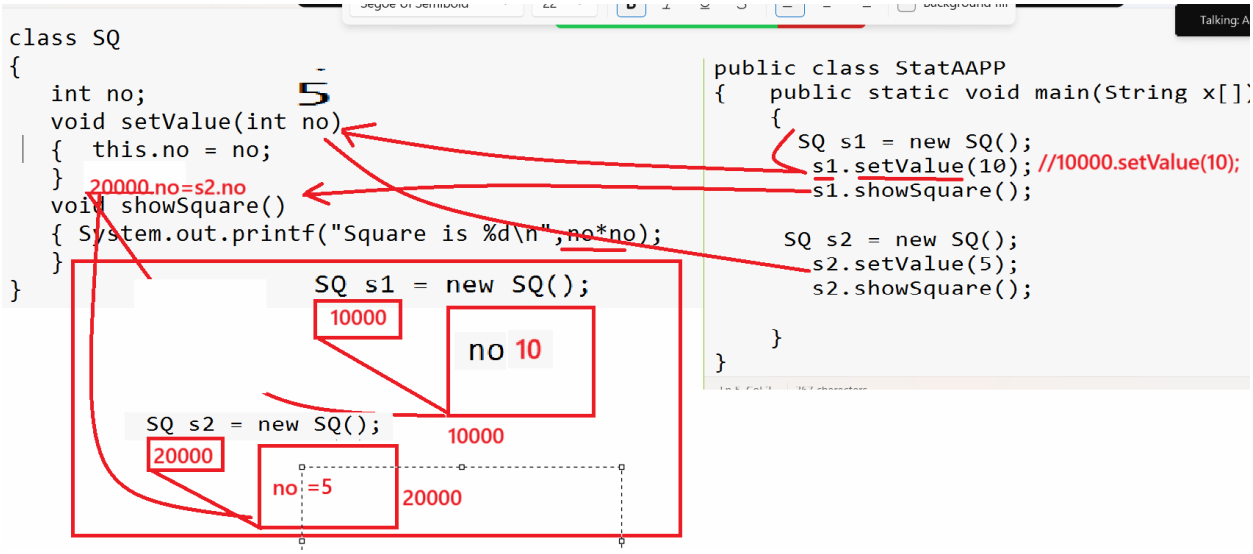
Note: if we want to solve above problem we can use this reference

Q. What is this reference?

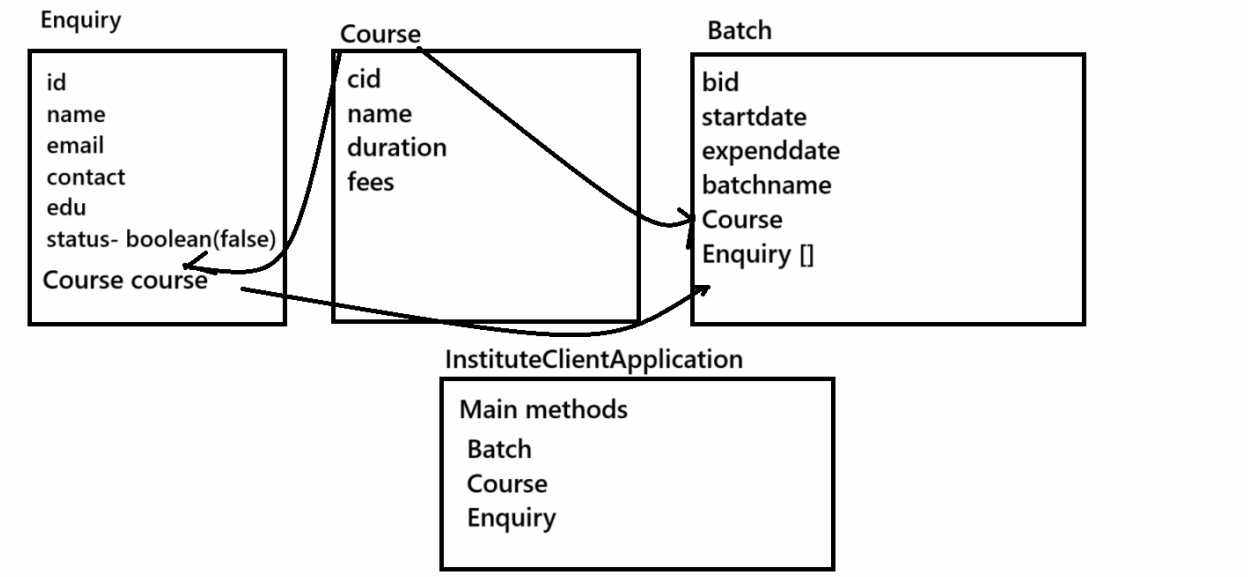
This is internal reference present in every class which is used for point to current running object in memory or working object in memory

Q. How we can identify current running object?

Function calling object called as working object and this always use the address or reference of function calling object.



Example: Institute Management Application using classes and object with array of objects concept.



Case 1: Add New Courses (3 course) : using this case we can create new course using Course class array of object
Note: course name should be duplicate as well as course id should be duplicated

Case 2: Take Enquiry: you have to create 10 enquiry objects using array of object and enquiry id should not duplicate and set all enquiry status by default false and show the all courses to enquiry and allocate course to enquiry using course id

Case 3: Enroll For Admission:
In this case you have to show all enquires and input the enquiry id which is ready for admission and change enquiry status from false to true

Case 4: Allocate Batch: you have to input the batch detail using single object and show the all courses and ask course id for batch then allocate batch to course and fetch all admission according to that course and allocate in batch

Case 5: View All Courses: show the
Case 6: View Course Wise students
Case 7: View Batch Wise Student
Case 8: View All Enquiry
Case 9: View All Admission
Case 10: View Cancel Enquiry

Nested class in JAVA

Q. What is nested class?

---

class within another class called as nested class

Q. Why we need to declare nested classes?

---

a) Some time if we want to segregate or isolate some dependent logic within class then we can use nested classes

```
class Numer
{
   int no;
  void setValue(int no)
  {
  }
  class ArithmeticOperation
  {  int getSquare(){
     }
    int getCube(){
   }
  }
 class GeneralOperation
  {
     int getRev(){
    }
    Boolean checkPrime(){
    }
  }
}
```

B) Normally we cannot declare class as private, protected or static and if we want to declare then we have option known as nested class.

Q. Why we cannot declare outer class as private?

because when we declare any class as private then we cannot create its object as well as cannot inherit it in any another class and as per terminology of OOP if we want to use any class then we have ways using object or using inheritance so declaring class as private it is useless class we cannot declare as private and if we try to declare as private then we get compile time error.

```
private class T
{
}
public class NestApp
{
    public static void main(String x[])
    {
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac NestApp.java
NestApp.java:1: error: modifier private not allowed here
private class T
        ^
1 error

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

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

Note: if we declare within another class then we can declare as private shown in following code.

```
class T1{
private class T
{
}
}
public class NestApp
{
    public static void main(String x[])
    {
    }
}
```

Note: if we think about this code we have class T as private but it is member class of T1 and T1 can mark its member as private so it is possible to mark nested class or child class as private.

```
Command Prompt

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

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

Q. Why we cannot declare outer class as static?

---

if we think about static keyword allocate memory before creating object of class and if we think about class cannot allocate memory without object so this behavior of static and class is opposite of each so we cannot declare outer class as static

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

```
Select Command Prompt

C:\Program Files\Java\jdk1.8.0_291\bin>javac OAPP.java
OAPP.java:1: error: modifier static not allowed here
static class O
       ^
1 error

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

Q. Why we cannot declare outer class as protected?

---

if we think about protected it is specially design to solve some problem in inheritance means when we declare member as protected then member can access by using inheritance only in child class but if we declare class as protected and if class contain public member then public member cannot access in child class if your parent class is protected as well as we cannot create object of protected class so this is major outer class cannot mark as protected

```
protected class O
{
}
public class OAPP
{
    public static void main(String x[])
    {
    }
}
```

```
Select Command Prompt

C:\Program Files\Java\jdk1.8.0_291\bin>javac OAPP.java
OAPP.java:1: error: modifier protected not allowed here
protected class O
          ^
1 error

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

There are four types of nested classes

---

1. Simple Nested class: if your one class directly declared within another class called as simple nested class.

```
class A{

  class B{

  }

}
```

How to create object of simple nested or inner class

Syntax: outerclassname.innerclassname ref = new outerclassname().new innerclassname();

or

outerclassname oref = new outerclassname();

outerclassname.innerclassname iref=oref.new innerclassname();

Example with source code

```
class Num
{   int no;
  void setValue(int no)
  { this.no=no;
  }
  class GenOperation  //simple nested class
  {   int r=0;
    int getRev()
    { int temp=no;
     while(temp!=0)
      { int rem =temp%10;
            temp = temp /10;
            r=r*10+rem;
      }
          return r;
    }
  }
  class Square{
            int getSquare()
            { return no*no;
            }
```

```
  }
}
public class SimInnApp
{
  public static void main(String x[])
  {  Num n = new Num();
     n.setValue(123);
    Num.GenOperation gn=n.new GenOperation();
    int result=gn.getRev();
   System.out.println("Reverse number is "+result);
    Num.Square s = n.new Square();
    result=s.getSquare();
  System.out.println("Square is  "+result);
  }
}
```

2. Static nested class: when we declare inner class as static called as static nested class.

Syntax:

```
 class classname{
    static class classname{
    }
 }
```

Example with source code

```
class SO
{ static class SI
 {
 }
}
```

How to create object of static nested class?

Syntax: outerclassname.innerclassname ref=new outerclassname.innerclassname();

Example with source code

```
class SO
{
  static class SI
  {  void show()
     { System.out.println("I am show method");
     }
  }
}
public class SOSIAPP
{
   public static void main(String x[])
   {
            SO.SI s = new SO.SI();
            s.show();
   }
}
```

3. Local Nested class: if we declare the class within function of outer class called as local nested class and we cannot create object of local nested class outside of his function block.

```
class College
{
   private class Student  //simple nested
   {
      private int id;
      private String name;
```

```
    private String address;
   public void setId(int id)
   { this.id=id;
   }
   public int getId()
   { return id;
   }
   public void setName(String name)
   { this.name=name;
   }
   public String getName()
   { return name;
   }
   public void setAddress(String address)
   { this.address=address;
   }
   public String getAddress()
   { return address;
   }
 }



boolean isTCIssue(int totalFees,int paidFees)
{
     int diff=totalFees-paidFees;
         if(diff==0)
         {  class PrintTC   //local nested class
           {
```

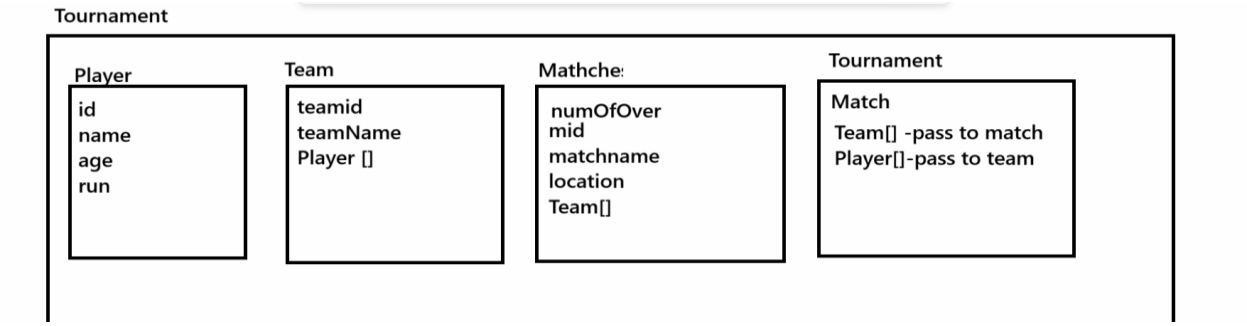
```
            void print(Student s)
            {
          System.out.println(s.getId()+"\t"+s.getName()+"\t"+s.getAddress());
        }
      }

        Student s1 = new Student();
         s1.setId(1);
         s1.setName("Ram");
         s1.setAddress("PUNE");
         PrintTC t=new PrintTC();
          t.print(s1); //pass studnet for leaving certification
        return true;
        }
        else
        {  return false;
        }
  }
}
public class SOSIAPP
{
   public static void main(String x[])
   {   College c = new College();
        boolean result=c.isTCIssue(100000,100000);
        if(result)
        { System.out.println("Best of Luck for future");
        }
        else
```

```
        { System.out.println("Please pay your remaining fees");
        }
    }
}
```

4. Anonymous nested class:

Note: we will discuss this type in inheritance after abstract class and interface topic

Example: WAP to develop the Tournament Application



case1: Add New Teams

case2: Add Players in team

Case 3: Create Match

Case 4: allocate team to match

Case 5: View all teams

Case 6: Show team wise player list

Case 7: show player wise run

Case 8: team wise player run aggregate

Case 9: decide the winner

case10: show the over count played by team

3. Provide encapsulation

Q. What is encapsulation?

Encapsulation means hide the implementation detail from end user at implementation level called as encapsulation means we hide the some implementation logic from end user at code level or logical level and the goal of encapsulation is data security here data consider as variable declared within class means

if we want to achieve encapsulation practically we required declare variable of class as private and access it via public function

```
class Student
{  private int id; //data/variable/state/data member
   private String name;
   private float per;
   public void setId(int id)
   { this.id=id;
   }
   public int getId(){
    return id;
   }
   public void setName(String name)
   { this.name=name;
   }
   public String getName()
   { return name;
   }
   public void setPer(float per)
   { this.per=per;
   }
   public float getPer()
   { return per;
   }
}
public class StudEncApp
{
   public static void main(String x[])
   {    Student s1 = new Student();
        s1.id=100;
   }
}
```

Note: if we think about left hand side code we get compile time error id has private access in Student means we declare id variable/data/state as private we can access it outside of class using its object directly but it is not possible because private variable cannot access outside of class directly so if we want to access private variable of class outside of his definition then we can access via public function
Because function contain logic/behaviour and logic decide data should accessible or not called as data security and it is purpose of encapsulation.

```
C:\Program Files\Java\jdk-23\bin>javac StudEncApp.java
StudEncApp.java:31: error: id has private access in Student
        s1.id=100;
           ^
1 error

C:\Program Files\Java\jdk-23\bin>
```

Example of encapsulation or Can we give scenario for encapsulation or real time example of encapsulation?

Suppose consider we are working for school management system or Institute management system and we have following types of login

1. Admin Login: according to system admin has 100% access of all systems activities
Like as admin can access student data, admin can access student fees data, email, contact as well as all course info and batch info

2. Councilor login: we want to give student name, email and contact access to councilor as well as course info not fees and collection info

3. Teacher login: we want to access only student name and batch detail not provide access of student contact and email to teacher

Example with source code

Student.java

```
public class Student
{
   private int id;
   private String name;
   private String contact;
   private String email;
   private int paidFees;

  public void setPaidFees(int paidFees)
  { this.paidFees=paidFees;
  }
  public int getPaidFees()
  { return paidFees;
  }
```

```
  public void setId(int id)
  { this.id=id;
  }
  public int getId()
  { return id;
  }
  public void setName(String name)
  { this.name=name;
  }
  public String getName()
  { return name;
  }
  public void setContact(String contact)
   { this.contact=contact;
   }
  public String getContact()
  { return contact;
  }
  public void setEmail(String email)
  { this.email=email;
  }
  public String getEmail()
  { return email;
  }
}
```

Course.java

```
class Course
{
 private int id;
 private String name;
 private int fees;

 public void setId(int id)
 { this.id=id;
 }
 public int getId()
 { return id;
 }
 public void setName(String name)
  { this.name=name;
  }
 public String getName()
  { return name;
  }
  public void setFees(int fees)
  { this.fees=fees;
  }
```

```
  public int getFees()
  { return fees;
  }
}
```

Authenticate.java

```
class Authenticate
{  private Course course;
  private Student student;
  void setStudent(Student student)
  {  this.student=student;
  }
  void setCourse(Course course)
  { this.course=course;
  }

  void verifyUser(String userType)
  {
      if(userType.equals("admin"))
          {
System.out.println("Course Info");
System.out.println(course.getId()+"\t"+course.getName()+"\t"+course.getFees());
System.out.println("Student info");
System.out.println(student.getId()+"\t"+student.getName()+"\t"+student.getContact()+"\t"+student.get
Email()+"\t"+student.getPaidFees());
          }
          else if(userType.equals("councilor"))
      {
System.out.println("Course Info");
System.out.println(course.getId()+"\t"+course.getName()+"\t"+course.getFees());
System.out.println("Student info");
System.out.println(student.getId()+"\t"+student.getName()+"\t"+student.getContact()+"\t"+student.get
Email());
          }
      else if(userType.equals("teacher"))
      {System.out.println("Course Info");
System.out.println(course.getId()+"\t"+course.getName()+"\t"+course.getFees());
System.out.println("Student info");
System.out.println(student.getId()+"\t"+student.getName());

      }
      else{
            System.out.println("Invalidate login");
       }
  }
}
```

TestEncapsulation.java

```
public class TestEncapsulation
```

```
{
  public static void main(String x[])
  {
        Authenticate  auth=new Authenticate();
        Student s1 = new Student();
        s1.setId(1);
        s1.setName("ABC");
        s1.setEmail("abc@gmail.com");
        s1.setContact("1234567891");
        s1.setPaidFees(10000);
        Course c1 = new Course();
        c1.setId(1);
        c1.setName("JAVA");
        c1.setFees(10000);

     auth.setStudent(s1);
        auth.setCourse(c1);
        auth.verifyUser("teacher");

  }
}
```