:- Argument Passing in Java :-

In Java, whenever we will pass argument to a Method we can only pass two things

(i) passing Variable / value.

(ii) Pass Reference (Array Refrence, object Reference)

Passing Variable :-

(i) when we pass a Variable as argument to a Method, Java will always pass its value.

(ii) This value is then Received by formal argument declared in the Methods argument list

(iii) Now both actual and formal argument will have the same value but both have different address.

(iv) So if we make any changes in the formal argument, it is not going to effect the actual argument.

(v) So, the Conclusion is that Variable are passed by using Pass By value

29 lect

```
Class Demo
{
    Public ~~Static~~ void increment (int i, int J)
    {
        i++;
        J++;
    }
}
Class UseDemo
{
    Public void Static main (String [] args)
    {
        Demo obj = new Demo ( );
        int i = 10;
        int J = 20;
        obj. increment (i, J);
        Sout ("i", i+J);        → 10, 20.
    }
}
```

local ~~instance~~ memeber

Passing Reference as Argument to Method :-

(i) Whenever we pass any Reference as argument to a Method then Java passes the address of object or Array pointed by that Reference

(ii). This address is then received by a formal Reference declared in the Method's arg list.

(iii) So now, both, the actual and formal are pointing to the same object.

(iv) Thus if we make any changes in the objects value (value of data member of the object) via formal reference then this change will be reflected in original object.

(v) And this might give us an impression that it is pass by reference.

(vi) But this is not true, because if we make any change in the formal reference (ex. making the formal reference point to a new object) then it will have no effect on the original object or the actual Reference passed as argument.

(vii) Thus we can say that in Java even reference are passed using pass by value.

29 March

Passing object Refrence :-

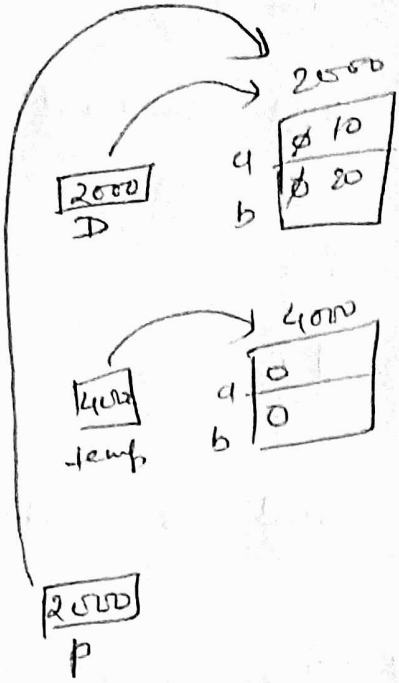
```
Class Data
{
   private int a, b ;

   Public void setdata (int i, int J)
    {
        a = i ;
        b = J ;
    }

   public void Showdata( )
   {
     Sout ("a = " +a+ ", b = " +b) ;
    }

   public void increment (Data P)
   {
       P. a = P. a+1 ;
       P.b = P.b +1 ;
   }
}
```

```
Class Usedata
{
    Public Static void main (String [] args)
    {
        Data D = new Data ();
        D. Setdata (10, 20);
        Sout (" Before increment ");
        D. Showdata ();
        Data temp = new Data ();
        temp. increment (D);
        Sout (" After increment ");
        D. Showdata ();
    }
}
```



Output

Before incrementing

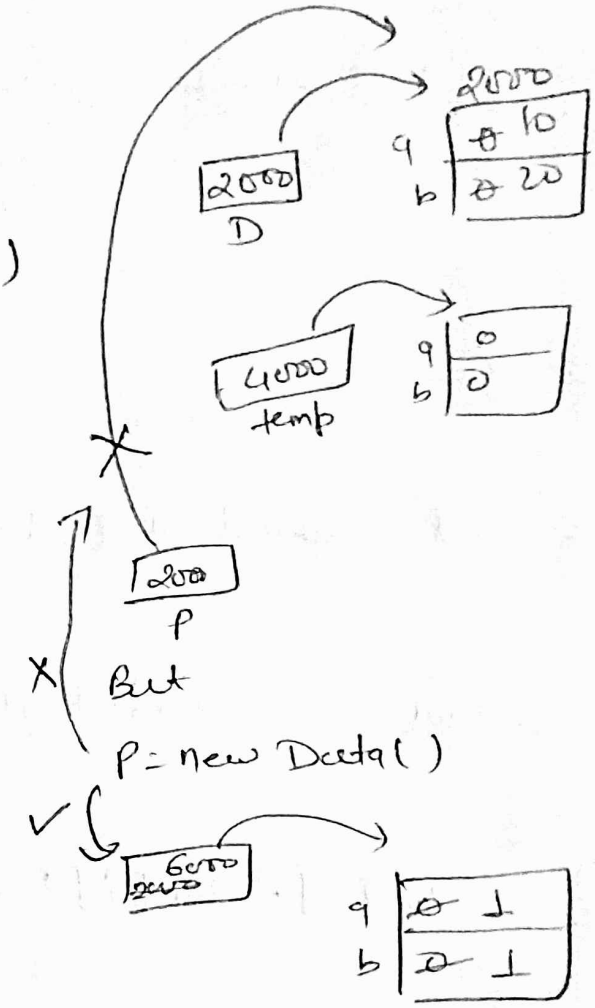
a = 10, b = 20

After incrementing

a = 11, b = 21.

```
Class Data
{
   private int a, b;
   Public void Set-data(int i, int j)
   {
      a = i;
      b = J;
   }
   public void Showdata()
   {
    Sout ("a=" +af ", b=" +b);
   }
    Public vold increment (Data P)
    {
        P = new Data()
        P.a = P.a +1;
        P.b = P.b + 1;
    }
}
```

```
Class Usedata
{
  Public Static void main (String [] args)
  {
    Data D = new Data();
    D. Setdata (10, 20);
    Sout ("Before increment")
    D. Showdata();
    Data temp = new Data();
    temp. increment (D);
    Sout (" After increment");
    D. Show Data();
  }
}
```



Output

Before increment

a=10, b=20

After increment

a=10, b=20

Passing Array as Argument

Syntax.

<accessMod> <returntype> <Methodname> (<datatype>[] <arrayreference>);

```
Class Demo
{
   public void doubler(int [] brr)
   {
      for(int i=0 ; i< brr.length ; i++)
      {
         brr[i] = brr[i] * 2
      }
   }
}
Class UseDemo
{
   PSVM
   {
      int [] arr = {10, 20, 30, 40, 50};
```

```
    Sout("Before doubling");
    for(int x : arr)
    {
       Sout(x);
    }
  Demo obj = new Demo();
    Obj.double(arr);
    Sout("After doubling");
    for(int x : arr)
    {
      Sout(x);
    }
  }
}
```

Output

```
Before doubling
10
20
30
40
50
After doubling
20
40
60
80
100.
```