Deep copy :

* In deep copy two different objects will be created in two different memory locations so, If we modify the content of one object then another object will not be modified.

```
//Program on deep copy

package com.ravi.deep_copy;

class Product
{
    private double price;

    public Product()
    {

    }

    public Product(double price)
    {
        super();
        this.price = price;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    @Override
    public String toString() {
        return "Product [price=" + price + "]";
    }

}


public class ProductDemo
{
    public static void main(String[] args)
    {
        Product p1 = new Product();
        Product p2 = new Product(5000);

        System.out.println("hello");

        //Copy the content of p2 to initialize the p1 data
        p1.setPrice(p2.getPrice());

        System.out.println("After copy");
        System.out.println(" hello");

        p1.setPrice(3000);
        System.out.println("After Modification in p2");
        System.out.println(" hello");

    }

}
```

** Pass By Value :

* Java does not support pointers so java only works with pass by value.

* Pass by value means we are sending the **copy of original data** to the this method.

Diagram for PassByValueDemo1.java

```
package com.ravi.pass_by_value;

class Customer
{
    private double bill;

    public Customer(double bill)
    {
        super();
        this.bill = bill;
    }

    public double getBill()
    {
        return bill;
    }

    public void setBill(double bill)
    {
        this.bill = bill;
    }
}

public class PassByValueDemo1
{
    public static void main(String[] args)
    {
        Customer cust = new Customer(10000);
        System.out.println(cust.getBill());  //10000

        accept(cust);
        System.out.println(cust.getBill());  //10000
    }

    public static void accept(Customer c1)
    {
        c1.setBill(50000);
    }
}
```

Diagram for PassByValueDemo2.java

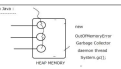
```
package com.ravi.pass_by_value;

class Employee
{
    private double salary = 75000;

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary)
    {
        this.salary = salary;
    }
}

public class PassByValueDemo2
{
    public static void main(String[] args)
    {
        Employee e1 = new Employee();
        System.out.println(e1.getSalary());//75000
        accept(e1);
        System.out.println(e1.getSalary());//75000
    }

    public static void accept(Employee emp)
    {
        emp = new Employee();
        emp.setSalary(90000);

    }

}
```

* In Java, We have immutable objects that means we cannot modify those objects.

* String and all the Wrapper classes are immutable objects so modification is not allowed in the modifing object.

```
package com.ravi.pass_by_value;

public class PassByValueDemo3
{
    public static void main(String[] args)
    {
        String str = new String("Java");
        System.out.println(str);

        accept(str);

        System.out.println(str);
    }

    public static void accept(String s1)
    {
        s1 = "Advanced Java";  //Strings are immutable (in-changed)

    }

}
```

*** Garbage Collector in Java :



It is an automatic memory management technique in java.

In C++ language, A programmer is responsible to allocate as well as de-allocate the memory otherwise we will get OutOfMemoryError.

In java language, Programmer is only responsible to allocate the memory, Memory de-allocation is automatically done by garbage collector.

Garbage Collector is a daemon thread which is responsible to delete the un-used objects from the HEAP Memory. Actually it scans the heap memory and identifying which objects are eligible for Garbage Collection.[THE OBJECTS WHICH DOES NOT CONTAIN ANY REFERENCES, ONLY THESE OBJECTS ARE ELIGIBLE FOR GC]

It internally uses an algorithm called Mark and Sweep algorithm to delete the un-used objects.

As a developer we can also explicitly call garbage collector by writing the following code.

System.gc();