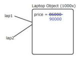
What is a shallow copy in Java ?

* Shallow copy is technique through which a **single object** will be reffered by **multiple reference** variables.

* All the reference variables are **pointing** to same single object so, if we modify the content of an object by using any reference variable then Original object will be modified.

* Always non static methods (insatnce methods) are used to modify the object properties that means IF WE WANT TO PERFORM ANY OPERATION ON THE EXISTING OBJECT THEN WE SHOULD USE NON STATIC METHODS.

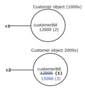


//Programs :

```
package com.ravi.shallow_copy;

public class ShallowCopyExample
{
   void main()
   {
       Laptop lap1 = new Laptop(86000);
       IO.println("Laptop price is :"+lap1.getPrice());

       //Shallow copy
       Laptop lap2 = lap1;
       lap2.setPrice(90000);

       IO.println("After Modification :");
       IO.println("Laptop price is :"+lap1.getPrice()); //9000
       IO.println("Laptop price is :"+lap2.getPrice()); //90000
   }
}

class Laptop
{
  private double price;

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

  public double getPrice()
  {
    return price;
  }

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

}
```

Deep copy :

* In Deep copy two different objects are created in two different memory location.
* We can copy one object data to another object.
* Here we have two different objects are available so, If we modify one object another object will not be modified.




```
package com.ravi.deep_copy;

public class CustomerDemo
{
   void main()
   {
       Customer c1 = new Customer();

       //Creating another object
       Customer c2 = new Customer(12000);


       //Copy the content of 2nd object to first
object
       c1.setCustomerBill(c2.getCustomerBill());

       //Printing both the objects bill
       IO.println(c1.getCustomerBill()); //12000
       IO.println(c2.getCustomerBill()); //12000

       //Modifying the 2nd Object bill
       c2.setCustomerBill(15000);

       IO.println("After modification");
       IO.println(c1.getCustomerBill()); //12000
       IO.println(c2.getCustomerBill()); //15000

   }
}

class Customer
{
    private double customerBill;

    public Customer()
    {
    }

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

    public double getCustomerBill()
    {
        return customerBill;
    }

    public void setCustomerBill(double customerBill)
    {
        this.customerBill = customerBill;
    }

}
```

Pass By Value in Java :

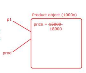
* Java does not support pointers so Java only works with **Pass By Value**
* In Java, Actually we are passing the copy of value from one method to another method.
  For primitive types : The actual value is copied.

  For objects : The reference (address) is copied, not the actual object. So both references still point to the same object in memory.

```
package com.ravi.pass_by_value;

public class PassByValueDemo1
{
    void main()
    {
        int y = 200;
        IO.println("Original Value :"+y); //200
        accept(y);
        IO.println("After calling accept method :"+y); //200
    }

    void accept(int x)
    {
        x = x + 50;
    }
}
```

```
package com.ravi.pass_by_value;

public class PassByValueDemo2
{
    void main()
    {
        int y = 200;
        IO.println("Original Value :"+y); //200
        y = accept(y);
        IO.println("After calling accept method :"+y); //250
    }

    int accept(int x)
    {
        x = x + 50;
        return x;
    }
}
```

```
package com.ravi.pass_by_value;


public class PassByValueDemo3
{
   void main()
   {
      Product p1 = new Product();
      IO.println("Price is :"+p1.getPrice()); //15000
      accept(p1);
      IO.println("Price is :"+p1.getPrice()); //18000
   }

   void accept(Product prod)
   {
      prod.setPrice(18000);
   }
}

class Product
{
    private double price = 15000;

    public double getPrice()
    {
        return price;
    }

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

}
```




```
package com.ravi.pass_by_value;

public class PassByValueDemo4
{
  void main()
  {
      Student raj = new Student();
      IO.println("Raj marks is :"+raj.getMarks()); //85
      accept(raj);
      IO.println("Raj marks is :"+raj.getMarks()); //85
  }

  public void accept(Student s1)
  {
      s1 = new Student(); //4000x
      s1.setMarks(95);
  }
}

class Student
{
    private double marks = 85;

    public double getMarks()
    {
        return marks;
    }

    public void setMarks(double marks)
    {
        this.marks = marks;
    }

}
```