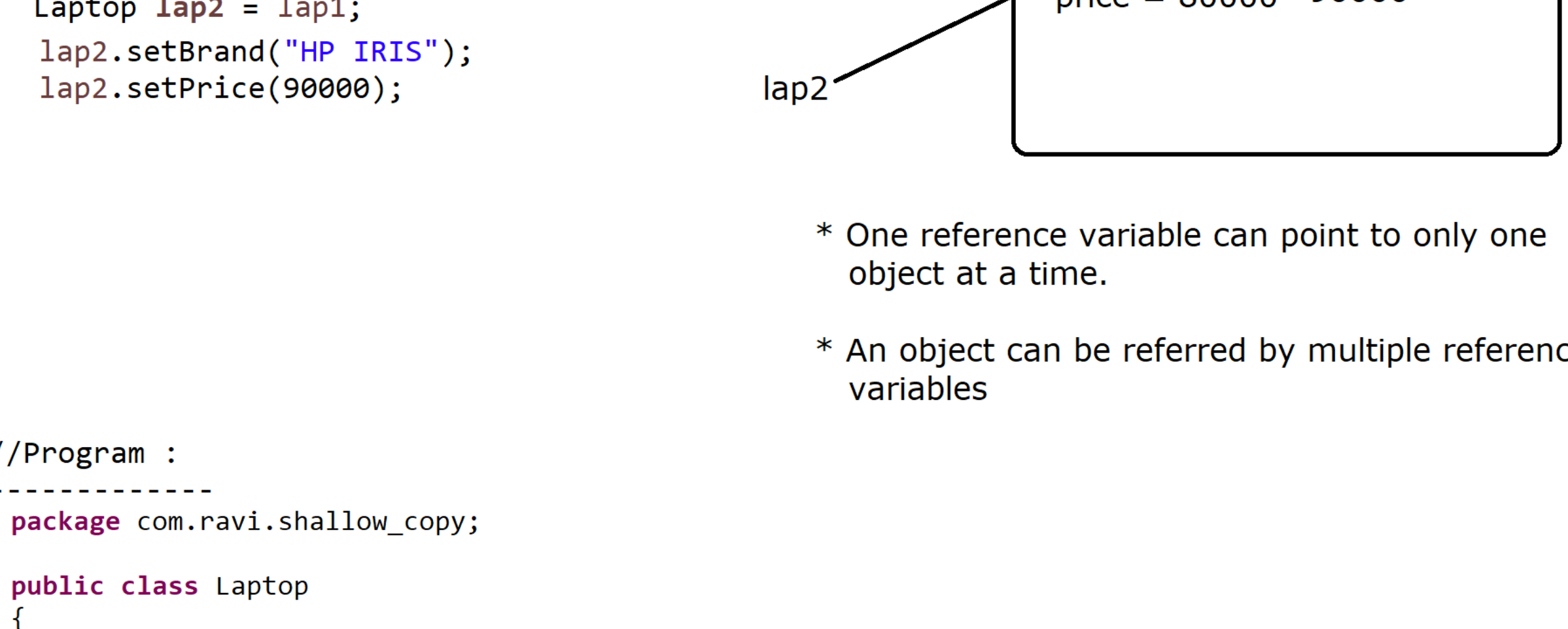


Shallow Copy and Deep copy :

Shallow Copy :

\* In Shallow, We will create **only one object** and same existing object will be referred by multiple reference variable.  
\* In this shallow copy, If we modify the content of one object by any of the reference variable then original object will be modified.



\* One reference variable can point to only one object at a time.

\* An object can be referred by multiple reference variables

```
//Program :
package com.ravi.shallow_copy;

public class Laptop
{
    private String brand;
    private double price;

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

    public String getBrand()
    {
        return brand;
    }

    public void setBrand(String brand)
    {
        this.brand = brand;
    }

    public double getPrice()
    {
        return price;
    }

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

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

package com.ravi.shallow_copy;

public class LaptopDemo
{
    public static void main(String[] args)
    {
        Laptop lap1 = new Laptop("HP", 80000);
        System.out.println("Before Change :");
        System.out.println(lap1);

        Laptop lap2 = lap1;
        lap2.setBrand("HP IRIS");
        lap2.setPrice(90000);

        System.out.println("After Change :");
        System.out.println(lap1);
        System.out.println(lap2);
    }
}
```

Deep Copy :

\* In Deep copy we will create two different objects, Here one object will copy the data of another object.  
\* Here If we modify the content of one object then another object content will not be modified.

```
package com.ravi.deep_copy;

public class Employee
{
    private int id;
    private String name;

    public Employee() //User-defined NO Argument Constructor
    {
        id = 0;
        name = null;
    }

    public Employee(int id, String name)
    {
        super();
        this.id = id;
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + "]";
    }
}

package com.ravi.deep_copy;

public class EmployeeDemo {

    public static void main(String[] args)
    {
        Employee e1 = new Employee(); //id = 0 name = null
        Employee e2 = new Employee(111,"Scott"); //id = 111 name = Scott
        System.out.println(e1);
        System.out.println(e2);

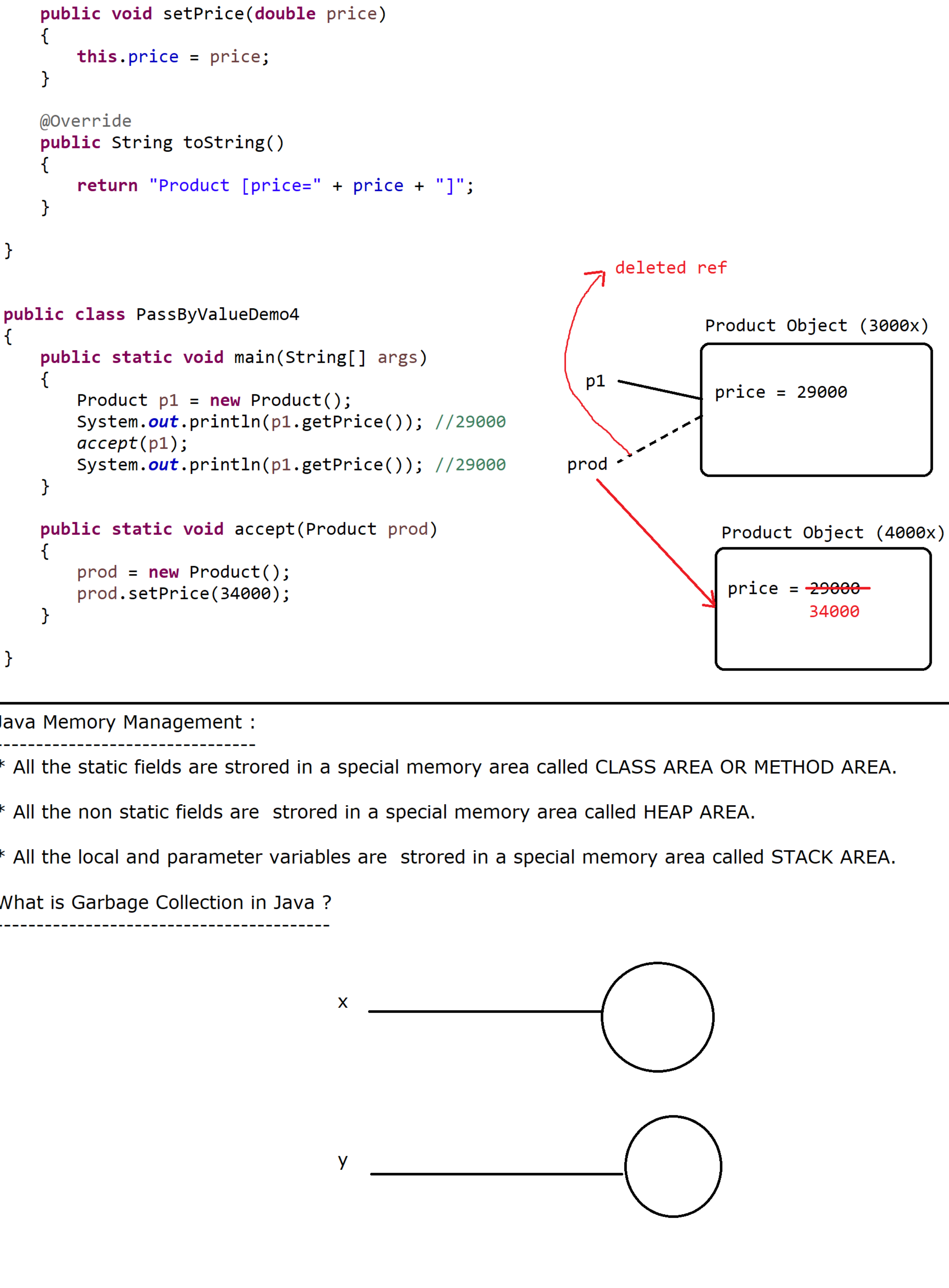
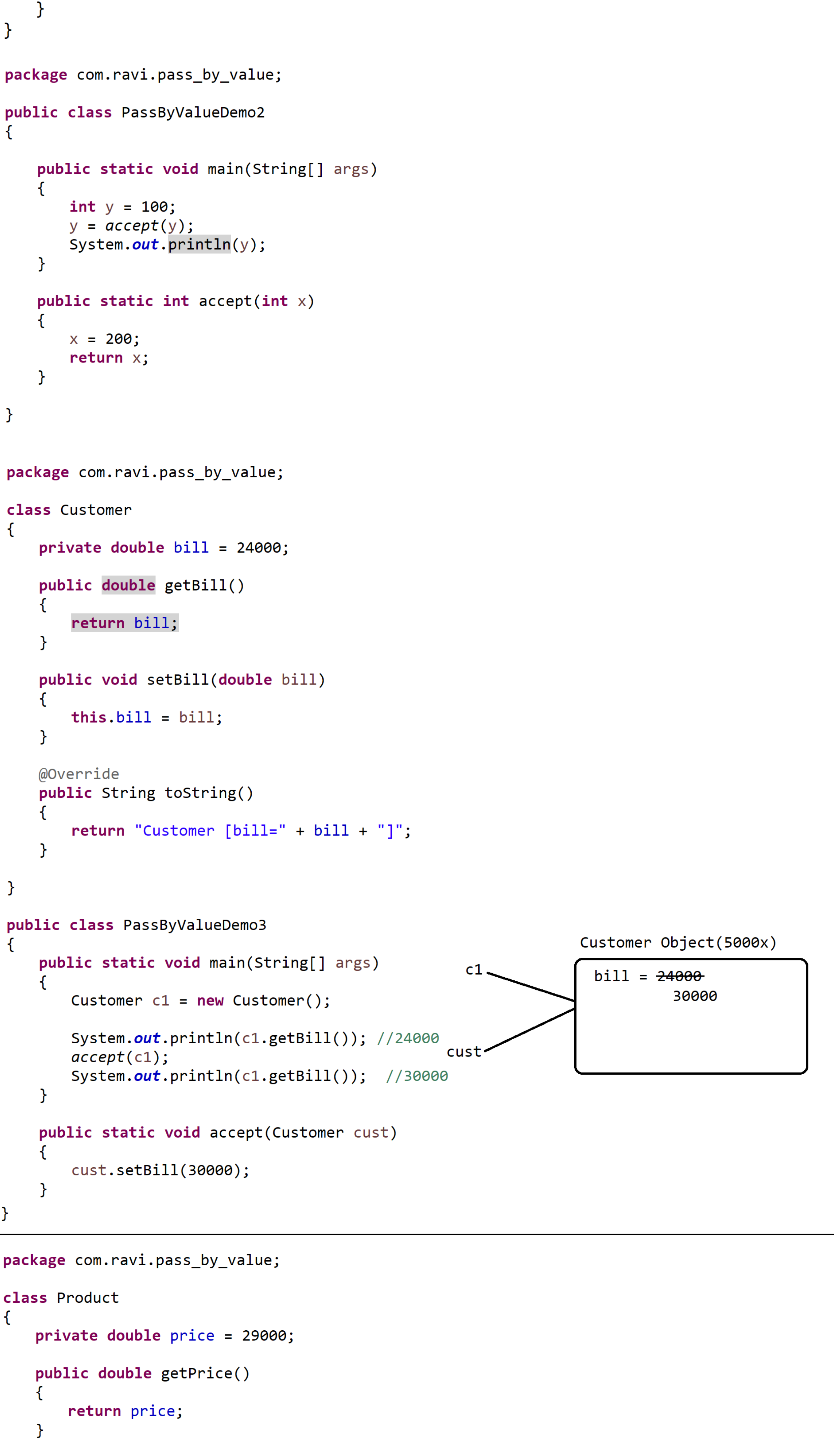
        System.out.println("After deep copy");
        e1.setId(e2.getId()); //e1.id = 111
        e1.setName(e2.getName()); //e1.name = Scott

        e2.setId(999);
        e2.setName("Raj");

        System.out.println(e1);
        System.out.println(e2);
    }
}
```

Pass By Value :

\* Java does not support Pointers so, In java we have only **Pass by Value**  
\* Pass By Value sending the copy of Original Data to the Method.



Java Memory Management :

\* All the static fields are stored in a special memory area called CLASS AREA OR METHOD AREA.

\* All the non static fields are stored in a special memory area called HEAP AREA.

\* All the local and parameter variables are stored in a special memory area called STACK AREA.

What is Garbage Collection in Java ?

