Unknown date | Unknown author

G-Fact 31 (Java is Strictly Pass by Value)

Consider the following Java program that passes a **primitive type** to function.

```
public class Main
{
    public static void main(String[] args)
    {
        int x = 5;
        change(x);
        System.out.println(x);
    }
    public static void change(int x)
    {
        x = 10;
    }
}
```
Run on IDE

Output:

```
5
```

We pass an int to the function “change()” and as a result the change in the value of that integer is not reflected in the main method. Like C/C++, Java creates a copy of the variable being passed in the method and then do the manipulations. Hence the change is not reflected in the main method.

How about objects or references?

In Java, all primitives like int, char, etc are similar to C/C++, but all non-primitives (or objects of any class) are always references. So it gets tricky when we pass object references to methods. Java creates a copy of references and pass it to method, but they still point to same memory reference. Mean if set some other object to reference passed inside method, the object from calling method as well its reference will remain unaffected.

The changes are not reflected back if we change the object itself to refer some other location or object

If we assign reference to some other location, then changes are not reflected back in main().

```
class Test
{
    int x;
    Test(int i) { x = i; }
    Test()      { x = 0; }
}
class Main
{
    public static void main(String[] args)
    {

        Test t = new Test(5);


        change(t);

        System.out.println(t.x);
    }
    public static void change(Test t)
    {



        t = new Test();
        t.x = 10;
    }
}
```

Run on IDE

Output:

```
5
```

Changes are reflected back if we do not assign reference to a new location or object:

If we do not change the reference to refer some other object (or memory location), we can make changes to the members and these changes are reflected back.

```
class Test
{
    int x;
    Test(int i) { x = i; }
    Test()      { x = 0; }
}
class Main
{
        Test t = new Test(5);


        change(t);

        System.out.println(t.x);
    }


    public static void change(Test t)
    {
        t.x = 10;
    }
}
```
Run on IDE

Output:

10

Exercise: Predict the output of following Java program

```
class Main {

   public static void swap(Integer i, Integer j)
   {
      Integer temp = new Integer(i);
      i = j;
      j = temp;
   }
   public static void main(String[] args)
   {
      Integer i = new Integer(10);
      Integer j = new Integer(20);
      swap(i, j);
      System.out.println("i = " + i + ", j = " + j);
   }
}
```
Run on IDE