Lesson on Method Calling

Consider the following program:

```
public class CallByValue
{
    public static void main (String[] args)
    {
        String person1 = "Mark";
        int amountPerson1 = 200;
        String person2 = "Anne";
        int amountPerson2 = 300;

        addMoneyInBank (person1, amountPerson1, 50);
        addMoneyInBank (person2, amountPerson2, 25);

        System.out.println (person1 + " has " + amountPerson1 + " euros.");
        System.out.println (person2 + " has " + amountPerson2 + " euros.");
    }

    public static void addMoneyInBank (String person, int amount, int
amountToAdd)
    {
        amount = amount + amountToAdd;
        System.out.println (person + " now has " + amount + " euros.");
    }
}
```

Run it and see what happens. Does it make sense to you?

Both person1 and person2 have their amount in bank increased by means of the method addMoneyInBank(). But then this money does not appear when the last two lines of method 'main' are executed.

This process is called '**call by value**' and it can be explained thus:

> *Changes to parameters are performed only locally and not globally i.e. changes are performed inside the method only and have **no** effect outside that particular method.*

There is however a different process called '**call by reference**' (known also as 'call by name'. In this technique:

*Changes to parameters are performed both locally and globally i.e. changes performed inside the method **also** hold outside that particular method.*

Now look at the following program:

```
public class CallByValue
{
    public static void main (String[] args)
    {
        System.out.println ("\f"); //clear the screen

        String[] personsNames = new String[2];
        int[] personsAmounts = new int[2];

        personsNames[0] = "Mark";
        personsAmounts[0] = 200;

        personsNames[1] = "Anne";
        personsAmounts[1] = 300;

        System.out.println (personsNames[0] + " has " + personsAmounts[0]
+ " euros.");
        System.out.println (personsNames[1] + " has " + personsAmounts[1]
+ " euros.");

        addMoneyInBank (personsAmounts, 50, 25);

        System.out.println (personsNames[0] + " has " + personsAmounts[0]
+ " euros.");
        System.out.println (personsNames[1] + " has " + personsAmounts[1]
+ " euros.");
    }

    public static void addMoneyInBank (int[] amounts, int amountToAdd0,
int amountToAdd1)
    {
        amounts[0] = amounts[0] + amountToAdd0;
        amounts[1] = amounts[1] + amountToAdd1;
    }
}
```

Run it and see what happens.

Now the changes in the amounts of money performed locally in the method addMoneyInBank() are also visible globally. This is an example of the following rule:

- When parameters are integers changes made locally are not made globally.
- When parameters are arrays changes made locally are also made globally.

In a more general way, we can say:

- *When parameters are primitive variables changes made locally are not made globally.*
- *When parameters are reference variables changes made locally are also made globally.*

Primitive variables

- Integers, doubles, Booleans etc are primitive variables
- They are made of one value
- They are also called 'atomic'.

Reference variables

- Are data structures
- They are made of a group of values
- One type of reference variable is the array.

A **reference** is:

- An address
- Also called 'pointer'

Exercise:

Consider the following program. Perform a dry run to find out what the output will be. Then run the program and see whether you were right.

```
public class CallByValRefEx
{
    public static void main (String[] args)
    {
        int[] numbers = {3, 7, 20, 5};
```

```
        displayArr (numbers);
        int x = 8;
        System.out.println (x);
        method1(x);
        System.out.println (x);
        method2 (numbers);
        displayArr(numbers);
    }

    public static void method1 (int a)
    {
        a = a + 3;
        System.out.println (a);
    }

    public static void method2 (int[] arr)
    {
        int len = arr.length;
        for (int i=0; i<len; i++)
           arr[i] = arr[i] + 6;
    }

    public static void displayArr (int[] arr)
    {
        int len = arr.length;
        for (int i=0; i<len; i++)
           System.out.print (arr[i] + " ");
        System.out.println();
    }
}
```