IST 256
Lab
Week 4 – Tuesday, February 4, 2014

1. Practice with understanding Methods (returning no value), working together with the instructor.

a. Open a new project with NetBeans named something like **TestMethods1.** This will be a Java application, so do not make a jFrameForm with a design window.

b. Put methods definitions into the main class

In the source code for the Main class, put the following code with method definitions into the program. These should go after the line with "public class Main" (the header of the entire class) and before the line with "public static void main … " (the header of the main method).

```
// method to demonstrate passing an integer value
// result is printed as output
 public static void square ( int y) {
    int ysquared;
    ysquared = y * y;
    System.out.println ("The number " + y + " squared is " + ysquared);
 }

 // method to demonstrate passing an String value and an integer value
 // result is printed as output
 public static void repeatText ( String text, int number) {
    // variable to build up the repeated output, initialized to the empty string
    String textout = "";
    for (int i = 1; i <= number; i++ ) {
       textout = textout + text;
    }
    System.out.println ("The repeated text is " + textout);
 }
```

If you copy the method definitions, adjust the indentation as necessary. Look at the definitions of the methods and observe the structure of the class, which now consists of three methods: main, square and repeatText.

c. Test the methods by calling them in the main class

Recall that in a Java class, the main method is the one that is executed. We will write code after the //TODO comment in the main method.

First write a line that prints a message at the start of this program:

```
System.out.println ("Method Testing:");
```

Next declare some variables to hold some actual parameters:

```
int testnumber, number_repetitions;
String testtext;
```

Next assign a test value to the variable testnumber and use the variable as an actual parameter in a method call to the method called square.

```
// define a number and pass it as a parameter to the square method
testnumber = 6;

// method call with one parameter
square(testnumber);
```

Run the program and observe what happens. Try some other values in the variable testnumber.

Next we will add code to assign test values to the other variables and use them as actual parameters in a method call to the method called repeatText.

```
// define a string and a number
//    and pass them as parameters to the repeatText method
testtext = "*";
number_repetitions = 19;

// method call with parameters
repeatText(testtext, number_repetitions);
```

Run the program and observe what happens. Try some other numbers and text values

2. Practice with understanding Methods (returning values), working together with the instructor.

a. Open another new project with NetBeans named something like **TestMethods2.** (Or you can create another Java class in the same project, but then you will have to make your own main method by copying from the TestMethods class in part 1 of this lab.) This will be a Java application, so do not make a jFrameForm with a design window.

b. Put methods definitions into the main class
In the source code for the Main class, put the following code with method definitions into the program. These should go after the line with "public class Main" (the header of the entire class) and before the line with "public static void main … " (the header of the main method).

```
// method to demonstrate passing an integer value
//    and returning an integer result
public static int squareR ( int y) {
   // variable to hold the square of y
   int ysquared;
   ysquared = y * y;
   return ysquared;
}

// method to demonstrate passing a String value and an integer value
//   and returning a String result
public static String repeatTextR ( String text, int number) {
   // variable to build up the repeated output, initialized to the empty string
   String textout = "";
   for (int i = 1; i <= number; i++ ) {
      textout = textout + text;
   }
   return textout;
}
```

If you copy the method definitions, adjust the indentation as necessary. Look at the definitions of the methods and observe how they are different from the previous methods with no return values.

c. Test the methods by calling them in the main class

Again write a line that prints a message at the start of this program:

```
System.out.println ("Method Testing with return values:");
```

Next declare some variables to hold some actual parameters and results:

```
int testnumber, number_repetitions, resultnumber;
```

```
String testtext, resulttext;
```

Next assign a test value to the variable testnumber and use the variable as an actual parameter in a method call to the method called squareR. Note that we must assign the result of the method call to another variable. This value can be used in computation, but here we are just going to print it:

```
// define a number and pass it as a parameter to the square method
testnumber = 6;

// method call with one parameter, saving the result in a variable
resultnumber = squareR(testnumber);

System.out.println ("The number " + testnumber + " squared is " + resultnumber);
```

Run the program and observe what happens.

Next we will add code to assign test values to the other variables and use them as actual parameters in a method call to the method called repeatText.

```
// define a string and a number
//        and pass them as parameters to the repeatText method
testtext = "*";
number_repetitions = 19;

// method call with parameters, saving the result in a variable
resulttext = repeatTextR(testtext, number_repetitions);

System.out.println ("The repeated text is " + resulttext);
```

Run the program and observe what happens.

Now try some experiments in which you use variables or values of the wrong type as parameters or to get the results. Observe the error messages that NetBeans tells you.

Finally, test your program with different values and write one example here:

Value for testnumber	
resultnumber	
Value for testtext	
Value for number_repetitions	
resulttext	

3. Practice writing a method, working on your own.

In this part of the lab, you will add write a third method and add it to the TestMethods2 class. The method definition should go near the top of the class, with the other methods in any order.

Write a method, called *cubed* that will compute the cube of a number. This method will be very similar to the squaredR method, except that it will multiply its argument together 3 times instead of twice.

The cubed method should accept a parameter of type double and return a result of type double.

Write a method call to this method and declare variables of type double to be the actual parameter and result. Add a call to System.out.println() to show the actual parameter and result.

Test several numbers in the *cubed* method, including the suggested values of 2.5 and 9.1.

Value for actual parameter	Value of result
2.5	
9.1	

Hand in this lab sheet with your experiments by the beginning of class on Thursday, Feb 13. Also copy the code for the part of the main class from TestMethods2 that includes all the methods:

squareR, repeatTextR, cubed and main

Print these and attach to your lab.