Ex.1. NetBeans IDE– creating, compiling and running the simple application

a) Run the NetBeans IDE
b) Create the new project (*New/Project* -> *Java Application*)
c) Review the source code of the application **class**. Netbeans creates a program (class) with any name, which contains the main method (it could be considered as equivalent to the main function in C). No declarations of variables or functions (methods) can not be placed outside the class. In the class, you can add your own declarations of variables and their functions (methods) support, which can be used for example in the main method.
d) Complete the generated program – the program should display the words "Hello Java!". Please note that the application source code is stored in a file with the same name as class name (the following code must be written for example in the file **Lab1.java** because a public class is called **Lab1**. NOTE - passwords are case sensitive.

```
public class Lab1 {  //begining of the class

    public static void main(String[] args) { //main method
        //instructions
        //declaration of local variables
       System.out.print("Hello java!!!\n");

    } //end of the main method

} //end of Lab1 class
```

e) Compile and run the project (*Run/Run Main Project*)
f) Complete the program - write the next line: „It is our first program!"

Ex. 2. Declaration of the variables, displaying the values

The primitive data types in Java: double, int, char.
All variables must be declared and they may have assigned an initial value such as:

```
int i,j;
int k=10;
double x=1.25;
char ch='z';
double y;
i=0;
j=333;y=0.001;
```

For our first application (in main method) add the above statements and display the values of all variables. The variables of the same type display on one line, for example:

```
i=0, j=333, k=10
```

The easiest way to display the numeric values is to use the '+' operator as the string concatenation operator, eg:

```
System.out.print("x="+x+", y="+y+".");
```

Ex.3. Formatted output

In order to properly format the displayed data, use the printf function and specify the data format string using for example % d (for integer values), % e or % f (for real values), % c for a single character, % s for string, eg:

```
System.out.printf("\n x=%4.1f, y=%8.2e, j=%6d char:%c",x,y,j,ch);
//x will be displayed on the 4 positions, one digits after the
//decimal point
```

```
    //y will be displayed on the 8 positions of the two digits after
    //the decimal point (in the floating point format)
    //j will be displayed on the 6 positions (space at the beginning)
    //ch will be displayed just as single character
```

The result is:

```
    x= 1.3, y=1.00e-03, j=   333 char:z
```

Modify displaing of the data - all the integer values display on at least five positions, and the real values in exponential form of the five places after the decimal point.

Ex.4.
Control instructions in java:

if-else:
 	
`if (condition)`
`{    //instructions`
`}`
`else`
`{    //instructions`
`}`
	switch:
	
	`switch (const_expression)`
	`{    case ... : ... break;`
	`     case ... : ... break;`
	`     ...`
	`     default : ... ;`
	`}`

while:
 	
`while (condition)`
`{`
`     //instructions`
`}`
	for:
	
	`for (int i=0;i<10;i++)`
	`{`
	`     //instructions`
	`}`

do-while:
 	
`do`
`{`
`     //instructions`
`}`
`while (condition);`	

Write an application that for fixed values of ***n*** (from the range 1 to 10) will display:

a) triangle: **1** **22** **333** **4444** **.** **.** **.** **nnnnnn...n**	b) triangle: **1** **12** **123** **1234** **.** **.** **.** **1234...n**	c) rectangle (*n*- number of rows and columns), eg. for *n*=4: **** **** **** ****

d) triangle (*n* – number of rows), eg. for *n*=4:
*
**
***
****
	e) triangle (*n*- number of rows), eg. for *n=5*:
	    *
	   ***
	  *****
	 *******
	*********	

Ex.5. The multiplication table
Write a program which will calculate and display the multiplication table of numbers from 1 to 10:
a) follow all the action locally in the method main (as in previous exercises)
b) call an special method *multi_table ()*

<u>Notice to the point b)</u>
The definition of such sub method may be placed in the class before or after the main method definition and method declaration may be for example:

```
public class Lab1 {

     //the sub method has no parameters and returns no value - but it
     //must be static:

    static void multi_table(){
              //display the multiplication table
         }
         public static void main(String[] args) {
          //main method calls the sub method:
          multi_table();
         }
     }
```