1. homework assignment; JAVA, Academic year 2014/2015; FER

First: read last page of this document. I mean it! You are back? OK. This homework consists of seven problems.

Problem 1.

Write a program `Rectangle` (place it into package `hr.fer.zemris.java.tecaj.hw1`) which asks user to provide width and height of rectangle (one per line, program reads from standard input). Program calculates and writes rectangle's area and perimeter. If program gets data from command line arguments, use them and do not ask user anything. Here are the rules for situation when program reads data from standard input.

1. If user enters anything, you can assume it is number (leave error checking and exception handling for later when we learn about them).
2. If nothing is provided, write complaint and ask user again to provide data.
3. If user provides negative number, write complaint and ask user again to provide data.
4. Don't copy&paste program segment for reading width and reading height (it's basically the same piece of code with different messages) – either extract code into a new method with appropriate arguments, or use arrays and loop.

Here are examples of program runs (user's input is show in red).

```
C:\tecaj> java -cp bin hr.fer.zemris.java.tecaj.hw1.Rectangle
Please provide width:
Nothing was given.
Please provide width: -43
Width is negative.
Please provide width: 10
Please provide height: -10
Height is negative.
Please provide height: 20.5
You have specified a rectangle with width 10.0 and height 20.5. Its area is 205.0 and
its perimeter is 61.0.
C:\tecaj> java -cp bin hr.fer.zemris.java.tecaj.hw1.Rectangle 25
Invalid number of arguments was provided.
C:\tecaj> java -cp bin hr.fer.zemris.java.tecaj.hw1.Rectangle 25 10
You have specified a rectangle with width 25.0 and height 10.0. Its area is 250.0 and
its perimeter is 70.0.
```

What follows is an example of program which reads a single line from standard input and converts it into a decimal number.

```
package hr.fer.zemris.java.tecaj.hw1;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class Example {

  public static void main(String[] args) throws IOException {
    // Stvori objekt za citanje s tipkovnice:
    BufferedReader reader = new BufferedReader(
      new InputStreamReader(new BufferedInputStream(System.in))
    );
```

```
    // Za citanje svakog retka zovi reader.readLine();
    String redak = reader.readLine();
    if(redak != null) {
      System.out.println("Korisnik je unio: \"" + redak + "\"");
      double broj = Double.parseDouble(redak);
      System.out.println("Broj je negativan: " + (broj<0.0));
    }
  }
}
```

When you read a line from user, use method `trim()` to remove whitespaces in front and after the data; use method `isEmpty()` to determine if what remained is an empty line. These are methods from `String`. Take a look at documentation:

http://docs.oracle.com/javase/8/docs/api/java/lang/String.html

If `reader.readLine()` return `null`, this means that programs standard input is closed and no further reading is possible; this is different from the situation when user just presses enter which will be interpreted as an empty line and empty string will be returned (i.e. "").

Problem 2.

Complete the following program.

```
package hr.fer.zemris.java.tecaj.hw1;

class ProgramListe {

  static class CvorListe {
    CvorListe sljedeci;
      String podatak;
  }

  public static void main(String[] args) {
    CvorListe cvor = null;

      cvor = ubaci(cvor, "Jasna");
      cvor = ubaci(cvor, "Ana");
      cvor = ubaci(cvor, "Ivana");

      System.out.println("Ispisujem listu uz originalni poredak:");
      ispisiListu(cvor);

      cvor = sortirajListu(cvor);

      System.out.println("Ispisujem listu nakon sortiranja:");
      ispisiListu(cvor);

      int vel = velicinaListe(cvor);
      System.out.println("Lista sadrzi elemenata: "+vel);
  }

  static int velicinaListe(CvorListe cvor) {
        // your code
  }

  static CvorListe ubaci(CvorListe prvi, String podatak) {
        // your code
  }
```

```
  static void ispisiListu(CvorListe cvor) {
        // your code
  }

  static CvorListe sortirajListu(CvorListe cvor) {
        // your code
  }
}
```

Problem 3.

Complete the following program.

```
package hr.fer.zemris.java.tecaj.hw1;

class ProgramStabla {

  static class CvorStabla {
    CvorStabla lijevi;
    CvorStabla desni;
      String podatak;
  }

  public static void main(String[] args) {
    CvorStabla cvor = null;

      cvor = ubaci(cvor, "Jasna");
      cvor = ubaci(cvor, "Ana");
      cvor = ubaci(cvor, "Ivana");
      cvor = ubaci(cvor, "Anamarija");
      cvor = ubaci(cvor, "Vesna");
      cvor = ubaci(cvor, "Kristina");

      System.out.println("Ispisujem stablo inorder:");
      ispisiStablo(cvor);

      int vel = velicinaStabla(cvor);
      System.out.println("Stablo sadrzi elemenata: "+vel);

      boolean pronaden = sadrziPodatak(cvor, "Ivana");
      System.out.println("Trazeni podatak je pronaden: "+pronaden);
  }

  static boolean sadrziPodatak(CvorStabla korijen, String podatak) {
    // ...
  }

  static int velicinaStabla(CvorStabla cvor) {
        return 0;
  }

  static CvorStabla ubaci(CvorStabla korijen, String podatak) {
    // ...
  }

  static void ispisiStablo(CvorStabla cvor) {
    // ...
  }

}
```

Problem 4.

Write a program `HofstadterQ` (place it into package `hr.fer.zemris.java.tecaj.hw1`) which calculates *i*-th number of Hofstadter's Q sequence. Use type `long` for calculations. The program accepts *i* as command line argument. This argument must be positive – if not, report an error.

Usage example:

```
C:\tecaj> java -cp bin hr.fer.zemris.java.tecaj.hw1.HofstadterQ 10
You requested calculation of 10. number of Hofstadter's Q-sequence. The requested
number is 6.
```

Problem 5.

Write a program `Roots` (place it into package `hr.fer.zemris.java.tecaj.hw1`). The program accepts three command-line arguments: real part of complex number, imaginary part of complex number, and required root to calculate (natural number greater than 1). The program computes and prints all requested roots of given complex number (also in form: real part plus imaginary part). In case that you need trigonometric functions (or similar), feel free to use methods of `Math` class – the documentation is here:

http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Usage example:

```
C:\tecaj> java -cp bin hr.fer.zemris.java.tecaj.hw1.Roots 3 4 2
You requested calculation of 2. roots. Solutions are:
1) 2 + 1i
2) -2 - 1i
```

Problem 6.

Write a program `PrimeNumbers` (place it into package `hr.fer.zemris.java.tecaj.hw1`). The program accepts a single command-line argument: a number *n* (*n*>0), and computes and prints first *n* prime numbers. We will consider 2 to be the first prime number.

Usage example:

```
C:\tecaj> java -cp bin hr.fer.zemris.java.tecaj.hw1.PrimeNumbers 4
You requested calculation of 4 prime numbers. Here they are:
1. 2
2. 3
3. 5
4. 7
```

Problem 7.

Write a program `NumberDecomposition` (place it into package `hr.fer.zemris.java.tecaj.hw1`). The program accepts a single command-line argument: a natural number greater than 1. The program calculates and prints the decomposition of this number onto prime factors.

Usage example:

```
C:\tecaj> java -cp bin hr.fer.zemris.java.tecaj.hw1.NumberDecomposition 84
You requested decomposition of number 84 onto prime factors. Here they are:
1. 2
2. 2
3. 3
4. 7
```

Please note. You can consult with your peers and exchange ideas about this homework *before* you start actual coding. Once you open you IDE and start coding, consultations with others (except with me) will be regarded as cheating. You can not use any of preexisting code or libraries for this homework (whether it is yours old code or someone else's); you can use classess and methods which comprise standard Java libraries. However, for this homework you can not use any of Java Collection Framework classes or its derivatives. Document your code! If you already know anything about the Java and object-oriented programming, do not use this knowledge in this first homework (i.e. explicitly using constructors is not allowed).

In order to solve this homework, create a **single blank Eclipse Java Project** and write your code inside. You must name your project's main directory (which is usually also the project name) `HW01-`*yourJMBAG*; for example, if your JMBAG is 0012345678, the project name and the directory name must be `HW01-0012345678`. Once you are done, export the project as a ZIP archive and upload this archive to Ferko before the deadline. Do not forget to lock your upload or upload will not be accepted. Deadline is March 18th 2015. at 11:59 PM.