Software Development I – Exercises **Winter Term 2012/2013**

Übungen zu Softwareentwicklung 1 **Assignment 4**

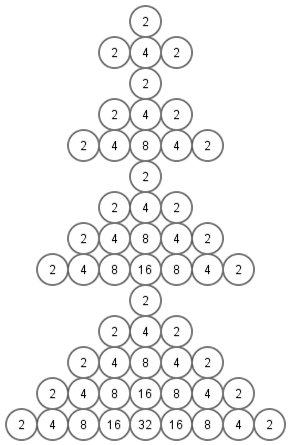
Name: _____ Teaching Assistant: _____

Student ID (Matr.Nr.): _____ Points (max. 24): _____

Group: _____ Deadline: Wed, Nov. 21st, 2012 12:00 noon

Instructor: _____ Editing time: _____

Question 1: A Mathematician's Christmas Tree **12 points**



If we like it or not, Christmas is closing in! To get you and your colleagues into the right mood, write a program to print a 'tree' by stacking number pyramids. Your algorithm will stack (n-1) pyramids, where each pyramid has an increased height, and 'n' is entered by the user. Each element of the pyramid shall be a 2^n result, restarting with 2^1 in each row and incrementing n until the middle of the row is reached. Then n is decreased until 2^1 is reached again at the end of the row.

Note that 'n' is also the maximum number of rows in the last pyramid of the tree. The image on the left side shows a maximum of 5 rows for the last pyramid – find appropriate limits for the user input. Limit the user input appropriately to achieve a Christmas Tree fitting to the max width of your console window.

Hints:

- To print this 'tree' properly you have to calculate the width of the very last row of the last pyramid to start the very first row (only 1 element) in the center. If there are numbers bigger than 9 in a row they will take more space, take this into account when calculating the *deltas* used for centering the 'tree'.
- Foresee 4 digits for each number in the Christmas Tree. Use the `printf()` function of Java to correctly format the printing of numbers.
  Example: `System.out.printf("%"+offset+"d",(int) Math.pow(2,exp));`

Example:

```
Christmas Tree Printing Service
-------------------------------

Height of Base Pyramid: 5

                    2
                2   4   2
                    2
                2   4   2
            2   4   8   4   2
                    2
                2   4   2
            2   4   8   4   2
        2   4   8  16   8   4   2
                    2
                2   4   2
            2   4   8   4   2
        2   4   8  16   8   4   2
    2   4   8  16  32  16   8   4   2
```

Write the program in Java. Provide all the material requested below at the very bottom ("Requested material…")

Question 2: Mini-Calculator **12 points**

Write a program `MiniCalc.java`, which will show each step of a basic calculator, for expressions such as `2 + 3 * 5`. Your program should display each step of the calculation until the final resulting value is displayed, like in the example below.

Your program should be able to input data from the command line. Therefore, the elements of your expression (i.e., values and operators) should be separated by spaces to be available in the `String[] args` array of the `main` method. For example, the following should be a valid way to call your algorithm: `java.exe MiniCalc 1 + 2 * 3 * 4 – 5`

Simplification:

- For simplification only consider the following three operators: `+`, `-`, `*`. However, the calculator should follow the correct operator ordering of operators (* before – before +). You can ignore parentheses.

Example:

```
java.exe MiniCalc 1 + 2 * 3 * 4 – 5

MiniCalc
========

 1 + 2 * 3 * 4 - 5
 1 + 6 * 4 - 5
 1 + 24 - 5
 1 + 19
 20
```

Note:

- Your program shall terminate in case you detect that the given expression is not a valid one.
- Consider also that calculated values may become negative.
- The solution shall be an iterative one!

Hints:

- Process the expression as an array of constituents (values and operators) of type `String`. During each reduction step search for the most binding operator and use the values left and right of the position of the operator for the calculation. Calculate the result of that operation and shrink the array accordingly. You need to convert from strings to decimal values before calculation and reconvert to replace the old constituents in the array with the new values.
- Consider to use methods where appropriate to structure your program.
- The following about arrays may be of help for you:
  - `anArray.length` - gives you the length of *anArray*.
  - `anArray[0]` - gives you the first element of *anArray*.
  - `anArray[anArray.length-1]` - gives you the last element of *anArray*.
- The following `String`-methods may be of help for you:
  - `int aString.length()` - returns the length of *aString*.
  - `char aString.charAt(int pos)` - returns the character at position `pos` of *aString*. The first character in *aString* is found at position 0, the last character is found at position `aString.length()-1`. Make sure to test for valid positions before passing to `charAt()` since invalid positions lead to an exception which must be avoided.
  - `String aString.substring(int beginIndex, int endIndex)` - returns a substring of *aString* including the character found at position `beginIndex` (n.b., first character is found at index position 0) excluding the character found at

`endIndex`. Make sure to test for valid `beginIndex` and `endIndex` before passing to `substring()` since invalid positions lead to an exception, which must be avoided.

- The following conversion-method may be of help for you:
  - `int parseInt(String aString)` - parses the string argument as a signed decimal integer. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '–' (`'\u002D'`) to indicate a negative value. The resulting integer value is returned.

Write the program in Java. Provide all the material requested below at the very bottom ("Requested material…")

Advice for user input:

- **Usage of the class `Input.java` for user input (read operations) is mandatory for assignments 1-5.** Please find the class (source code) in the exercise directory on the webpage: `(http://www.pervasive.jku.at/Teaching/lvaInfo.php?key=352&do=uebungen)`
  This class provides various methods to read values of different data types. In order to read digits and numbers from the console, you may for example, use the following functions:
  - `Input.readInt()` – reads an integer from the console (data type: `int`)
  - `Input.readFloat()` – reads a floating point number (data type: `float`)

  Example:

```
- int x;
- float y;
- System.out.println("x = ");
- x = Input.readInt ();  // read integer x
- System.out.println("y = ");
- y = Input.readFloat();  // read floating point number y
```

Requested material for all programming problems:

- **For each exercise, hand in the following:**
  a) **The idea for your solution written in text form**
  b) **Source code (Java classes including English(!) comments)**
  c) **Test plan for analyzing boundary values (e.g., minimal temperature allowed, maximum number of input, etc.) and exceptional cases (e.g., textual input when a number is required, etc.). State the expected behavior of the program for each input and make sure there is no "undefined" behavior leading to runtime exceptions. List all your test cases in a table (test case #, description, user input, expected (return) values).**
  d) **The output of your java program for all test cases in your test plan**
- Pay attention to using adequate and reasonable data types and meaningful English variable names for your implementation, check the user input carefully and print out meaningful error messages.