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

Übungen zu Softwareentwicklung 1 **Assignment 5**

Name: _____ Teaching Assistant: _____

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

Group: _____ Deadline: Wed, Nov. 28th, 2012 12:00 noon

Instructor: _____ Editing time: _____

Question 1: 2D-Diagrams **10 points**

Develop an algorithm for a Java program that will draw a point on a 2 dimensional grid. The program should accept the ***(x, y)-coordinate*** of the point and place it on a fixed **20 x 15** grid (use final fields, i.e. "constants" in your program for the diagram size). The grid should then be displayed, including the labeled axes like in the example below. Ensure the coordinates of the point are within the limits of the grid, however boundary points are considered valid (for example, x=0 and y=0 is a valid point). Provide a general solutionp that allows the programmer later to change the constants for the diagram size!

Example (this is what the user will see on the display):

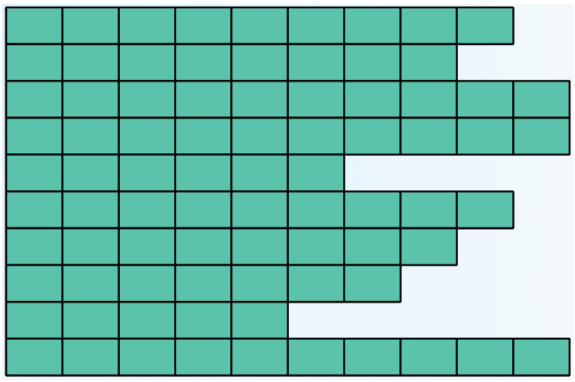
```
This program will place a point on a plot.
Enter the x-coordinate:
5
Enter the y-coordinate:
5

  ^
15+
  |
  |
  |
  |
10+
  |
  |
  |
  |
 5+    *
  |
  |
  |
  |
  +----+----+----+----+>
  0    5    10   15   20
```

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

Question 2: Grid of Emoticons **14 Points**

Write an algorithm that will randomly place 5 of the list of emoticons given in the accompanying file `emoticon.txt` onto a grid. The grid size for this problem is fixed in one dimension to **10**, the second dimension should be randomly assigned ("jagged" array) as indicated in the example below (value range **2 to 10**).



Example of a 2 dimensional "jagged" array

Note:

- The user will select the 5 emoticons they would like to see displayed on the grid.
- The program should follow these rules:
  - Emoticons are always displayed horizontally
  - Only complete emoticons are allowed (do not display part of an emoticon or split an emoticon across rows).
  - Do not display overlapping emoticons.
  - Do not overwrite existing emoticons (there should always be 5 complete emoticons).
  - A user is allowed to select an emoticon more than once.
- Once the board has been generated, it should be displayed to the screen.
- Please use the dash symbol ('-') to mark empty array elements.

Example (this is what the user will see on the display):

```
Please select 5 emoticons from the following list:
1: n_n
2: $v$
3: 8(>_<)8
4: W(^o^)W
5: (=_=)
6: (/_\)
7: >^..^<
8: (~-^)
9: (*-*)
10: <*)))-{
Select emoticon # 1
9
Select emoticon # 2
7
Select emoticon # 3
3
Select emoticon # 4
4
Select emoticon # 5
10

This is the generated grid:
----------
------
8(>_<)8---
--W(^o^)W-
------
--<*)))-{
--
->^..^<
(*-*)--
----------
```

Hints:

- Use the `java.util.Random` package to generate random numbers. You can find help on how to use the `nextInt()` method here:
  `http://docs.oracle.com/javase/6/docs/api/java/util/Random.html`
- Use the two dimensional array of chars in `emoticons.txt`. Keep in mind the array indices begin at element 0 and not 1. (i.e., `EMOTICONS[0][0]='n'`) and each row contains a different number of characters.

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.