2IP67.5 Instruction 4 – Methods

Netbeans

For the following exercises, some starter code is provided in a zip-file on the wiki (`http://www.win.tue.nl/~keesh/dokuwiki/doku.php?id=2ip65_en_2ip70`). Unzip the file, creating a folder Files4. This folder will contain some Java-files. When you need one of the starters, you can copy and paste the text into a Java file the or move the file to the folder `src` in your project folder. Netbeans will recognize the file and include it in your project. Alternatively, you can drag the file to the src-package in the navigation panel of Netbeans.

Exercise 1 – Vicious Circle

Here is a starter for a program that handles a circle. You can find this program in the folder `Files4`.

```
1  class Circle {
2          double radius;
3
4          void setRadius(double r) {
5                  radius = r;
6          }
7
8          // other methods go here
9
10         void demo() {
11                 setRadius(5);
12                 // more will come here
13         }
14
15         public static void main(String args[]) {
16                 new Circle().demo();
17         }
18 }
```

1. Add an instance variable `area` to the class and two methods with headers

   `void printData()`

   and

   `void calculateArea()`

   The first method should output the data of the object (the values of the instance variables) in a decent way. The second method should calculate the area of the circle and put the result in the variable `area`. Try the methods out by changing `demo`.

   Hint: `Math.PI` gives a good approximation of π.

2. Add an instance variable `perimeter` and a method `void calculatePerimeter()` that store respectively calculate the perimeter (*Dutch: omtrek*) of the circle. Change `printData` and `demo` accordingly.

3. * Add a method that draws on the consoel the circle built out of characters of your choice, e.g., starts. Assume that a character is 3 units high and 2 units wide. Of course, circles with a small radius will not look very circle-like.

Exercise 2 – Starry Starry Night

1. Write a class `Stars` with a demo method that calls a method `printStars` that prints 10 stars, like this:

   ****

2. Add a method with header

   `void printStars(int n)`

   that prints n stars on a single line.

   Hint: the method call `System.out.print("*");` prints one star *without a newline*. The call `System.out.println();` prints only a newline. Try out this printStars method.

3. Add the following method to the class:

```
void printRectangle() {
   printStars(5);
   printStars(5);
   printStars(5);
}
```

   When called from the demo method, you should see as output:

```
*****
*****
*****
```

   Change the method that it takes two parameters of type `int` that represent the width and the height of the rectangle respectively and that prints a rectangle of stars of that size. The header (first line of the definition) of this method will look like this:

   `void printRectangle(int width, int height)`

4. Add a method, similar to printRectangle, that prints a triangle of given size. You may decide for yourself how the triangle is exactly shaped and positioned.

5. All these methods print only stars. What does it take to have the methods print a different symbol? Give two ways.

Exercise 3 – Fun With Functions

1. Write a class with a method

   `double f(double x)`

   that computes and returns the value of $x^2 - 2x - 3$.

2. Add a method `table` that prints a table of values for x and the corresponding function value $f(x)$, where x ranges between -3 and 3 with steps of 0.5. So the table looks like this:

```
   x    f(x)
---------------
-3.0   12.0
-2.5   8.25
...
```

   Use tabs (`\t` inside String-expressions) to make columns. Don't worry too much about neat spacing etc.

3. Write a method that helps the user in finding a zero for the function f by means of a question game. The program asks the user a value for x and then gives the corresponding value for $f(x)$. This is repeated until the function value is between -0.001 and +0.001. Then the program stops with a wholehearted compliment. If you like, have it also print the number of tries.

4. * Make the program work for any quadratic function. To achieve this, add instance variables for a, b, and c and have `f(x)` calculate $ax^2 + bx + c$. Before the game starts, the program will ask for values for a, b, and c and set them.