CO520 Further Object-Oriented Programming
Assessment 4: In-Class Test

Stefan Kahrs

1st April 2019

Preamble

For this test, you will be creating a BlueJ project (you can use another IDE if you prefer), from scratch — there is no code download. Submit the completed project at the end on moodle as a .zip file or .jar file — notice that *this must contain* the java source code. You do not have to write any comments for this test.

Meta-comment: Generally, if you cannot do a question, just carry on with the other questions. If some task depends on a method you did not complete, then you could provide a dummy implementation (e.g. returning a fixed value) for the previous task.

Questions

There are overall 4 questions (worth 100 marks overall) with several subparts — if you do not see them all now it is because you would need to turn the page over.

1. In the week 22 exercise there was some code segment for a static method `printFigure`. Turn this into a proper class `Figure`, as follows:

   (a) As in the week 22 exercises, write a functional interface `BiIntPredicate` with a method `test` that has two `int` parameters and that returns a `boolean`. [3 marks]

   (b) Create a new class, called `Figure`. This class should have a non-static field `pred` of type `BiIntPredicate` and (as in week 22) also the static field `BOUND` of type `int`, again set to 7. [5 marks]

   (c) Intialise the `pred` field to a predicate whose test method always returns false. [4 marks]

   (d) Add set/get methods for the `pred` field. [4 marks]

   (e) Add a static method `negate` which has a `BiIntPredicate` parameter `x` and which also returns a `BiIntPredicate`. The predicate it returns should give true whenever `x` gives false, and vice versa. [4 marks]

   (f) Add a method `void invert()` which inverts the image (star to space, space to star) by modifying its `pred` field. [4 marks]

   (g) Write a `toString` method for the class. This should return a single string which, when printed, would produced the same image as `printFigure`. Note: do not forget to add the `\n` in places where `printFigure` uses a `println`. [8 marks]

(h) Write a `printFigure` method which does the same thing as the `printFigure` method from the week 22 exercises, except that it is (i) non-static, and (ii) the predicate used to create the drawing is not passed as a parameter to the method, it is here instead the field `pred` from the class. You can make use of your `toString` method here. [4 marks]

2. Printing figures to `System.out` is a rather specific thing to do with a figure. We may want to keep our options more open regarding that:

   (a) Write an abstract class `AbstractFigure` which is like `Figure`, except that the `printFigure` is here an abstract method. (You can copy/paste the other code.) [6 marks]

   (b) Make `Figure` a concrete subclass of `AbstractFigure`. [4 marks]

3. Instead of printing figures we could render them in a GUI. Here we want a class that is prepared to be placed inside a GUI.

   (a) Write a concrete class `GUIFigure` which is a subclass of `AbstractFigure`, and which has an additional field `area` of type `JTextArea` (from `javax.swing`). [4 marks]

   (b) Write a concrete `setTextArea` method that can be used to set the `area` field. [2 marks]

   (c) Add a concrete `printFigure` method which does the following: it places the text that makes up the figure in the `area` text-component, using that object's `setText` method. [4 marks]

   (d) Override the `invert` method: this should do everthing the `invert` method of the superclass does, but in addition it should also update the text of its text-component to the modified image. [6 marks]

4. Now we want to create a graphical user interface with some components inside.

   (a) Write a class `MyFigure` which extends `JFrame`, and which has fields of type `JButton` called `invert`, of type `JTextArea` called `canvas` and of type `GUIFigure` (see previous question) called `guif`. [6 marks]

   (b) The `invert` button should be initialised and carry the text "invert". The `canvas` area should be initialised to a `JTextArea` object, whose number of rows and columns is both 7 (i.e. `Figure.BOUND`). [8 marks]

   (c) In the constructor, write code that places `invert` and `canvas` in the North/Center region of the frame, respectively, and eventually makes the whole frame visible. [8 marks]

   (d) Set the predicate of the `guif` figure to the smaller predicate on integers, i.e. calling `test(x,y)` on that predicate should give the same value as `x<y`. [6 marks]

   (e) Set the text-component of `guif` to `canvas`. [4 marks]

   (f) Add an action listener to the `invert` button. This should invert the predicate of the `guif` object, and update the text in its text-component accordingly. [6 marks]