Additional Higher Grade Tasks

Object-Oriented Design, IV1350

These tasks are only for deciding the final grade. The requirement to pass the course, with grade E, is to pass all seminars.

Grading

Maximum score from these tasks is 3, which is added to the higher grade score from the seminar tasks. The total max higher grade score is 3 from these tasks and 4 from the seminar tasks, which makes a total of 7. Grade limits are as follows.

- 6 higher grade points give grade A
- 5 higher grade points give grade B
- 3 higher grade points give grade C
- 1 higher grade points give grade D
- No higher grade points are required for grade E

Task 1, Inheritance (1p)

In your *Process Sale* program, use the *Template Method* design pattern for the observers you wrote in task 2a in seminar 4. The template class you create must have the structure described by the pseudocode below. You are allowed to change variable, parameter and method names, and also to add variables, parameters and methods. *The structure with the try-catch block and the abstract methods must however remain unchanged.* The grade is not affected no matter how simple or advanced the view is.

```
// This is the method defined in the observer interface.
public void newSaleWasMade(int priceOfTheSaleThatWasJustMade) {
  calculateTotalIncome(priceOfTheSaleThatWasJustMade); // Calculate
                  //the total amount paid since the program started.
  showTotalIncome();
}

private void showTotalIncome() {
  try {
    doShowTotalIncome();
  } catch (Exception e) {
    handleErrors(e);
  }
}

protected abstract void doShowTotalIncome() throws Exception;

protected abstract void handleErrors(Exception e);
```

Listing 1: Pseudocode for the template class

- In the `Method` chapter of your report, explain how you worked and how you reasoned when implementing the template method pattern.
- In the `Result` chapter of your report, briefly explain source code for all classes you changed when implementing the template method pattern. Include links to your git repository, and make sure the repository is public. Also include a printout of a sample run.
- In the `Discussion` chapter of your report, thoroughly motivate that your code is a correct implementation of the template method pattern. Also thoroughly explain the benefits of using template method in your application. Alternatively, if you can't see any benefits, thoroughly motivate why that is the case.

Task 2, Inheritance vs Composition (1p)

This task relates to the section *Second Reason to Prefer Composition: Inheritance Breaks Encapsulation* in chapter 9.3 in *A First Course in Object-Oriented Development.* Make sure to understand that section before proceeding. The section describes how a class can be adapted using inheritance, and how it can be adapted using composition.

The task is now to adapt any class in the `java` libraries from Oracle. You are free to choose any class to adapt, *except a list*, and also free to choose how you adapt it. Write one new class that adapts using inheritance, and another new class that adapts using composition, as described in the course book section mentioned above. Also write a main method which instantiates your new classes and executes the adaptions. The program must include printouts illustrating how your classes work. A suggestion, if it's hard to find a class to adapt, is to choose `java.util.Random`.

- In the `Method` chapter of your report, explain how you worked and how you reasoned when writing the adapting classes.
- In the `Result` chapter of your report, briefly explain the source code. Include links to your git repository, and make sure the repository is public. Also include a printout of a sample run.
- In the `Discussion` chapter of your report, thoroughly compare adaption using inheritance and adaption using composition. There are comparison criteria in section 9.3 in the course book, and it is fine to use other criteria as well. Also draw a conclusion, is inheritance or composition preferred, or are they equal? You are of course not required to make the same conclusion as is made in the book.

Task 3, Testing Output (1p)

Write unit tests for all methods that print to `System.out` in your *Process Sale* program. This **must** include at least the `main` method, the `View` class and the receipt printout. If you have additional methods printing to `System.out`, also those must be tested. The tests of the `View` class must test all printouts in that class, but you only have to test printout containing information. It's not required to test printouts which exist only to make the view more readable.

- In the `Method` chapter of your report, explain how you worked and how you reasoned when writing the unit tests.
- In the `Result` chapter of your report, tell which tests you have written for this task and briefly explain the tests. Include links to your git repository, and make sure the repository is public.
- In the `Discussion` chapter of your report, thoroughly evaluate your unit tests using assessment criteria related to testing from the document `assessment-criteria-seminar3.pdf`, which is available on the `Seminar Tasks` page in Canvas.