COP 4331 Homework 4

Chapter 5 and Chapter 6

Instructor: Dr. Ionut Cardei

No homeworks will be accepted after the due date

Preparation/Delivery Instructions:

1. Write all your answers, in the order given in the homework file, in ONE PDF file.
   Upload the PDF file on Canvas.

- **Follow this format:**
  - Write your name followed by the section number.
  - For each problem write as a heading the problem number (e.g. "4.1").
  - The problem number must be clearly readable before the problem solution.
  - Java source files must be properly identified: write the file name as a heading, followed by the file content.
  - Make sure Java code and UML diagrams are readable. Make it easy for the grader.
  - Nice color syntax highlighting is not required, but appreciated. Consider using http://hilite.me/ .
  - Proper indentation and code formatting is required.

2. Package the directory with all source .java files (or packages) **in one ZIP file** and upload it on the homework Canvas page.

3. For full credit, your designs and code must follow the course guidelines. The code must compile without warnings and work correctly, as required in the problem description.

4. If you have a question about the problems, send email to the instructor or post on the Homework's Q&A forum.

Remider: it is academic misconduct to submit work that is not your original contribution.
Do not copy code taken from the web.
Understand that is very easy for the instructor to identify code shared by multiple students.

You can (and should) use any helpful code **from the textbook** for your answers.

Other general advice that will help you do well in this class. And build better code, too.

* !! Ask your instructor if you have any questions about the homework
(and anything else related to the class) !!
* Read the rubric posted on the Homework Solutions module.
* Consult the solutions for selected textbook problems, available at
http://www.horstmann.com/oodp2/solutions/solutions.html
* Do exactly what the problem asks you to do. There is no extra credit for unnecessary work.
Points are deducted if design or implementation requirements are not met.
If you have a question about the problems, send email to the instructor or post on the
Homework's Q&A forum.
* Do not rename classes and methods if these are given.
* Do not change method signatures, if specified.
* Design/code your classes for general use. Assume there are other programmers who will use your code.
* Avoid unnecessary side effects. Do not use static fields/methods, unless warranted (e.g. main()).
* check for errors and exceptions.
* Enclose methods that may throw exceptions within a try-catch block.
* Check parameters and variables before you do something in a method. E.g. average = sum/list.size() may throw an ArithmeticException if the list is empty.
* Use class (static) variables only when necessary (e.g. to share a variable between instances,
or for constants).
* Do not define new instance variables when local variables could do the job.
* Use nouns for class names and verbs for methods.
* Follow coding conventions; class names start with capitals, methods and variables start with lowercase, etc.
Here is the official guide: https://www.oracle.com/technetwork/java/codeconvtoc-136057.html

---

Scoring: non-optional problems total = 100 points

Homework problems (Chapter 5):

5.1.
Answer these questions:
a) How does a software engineer use design patterns ?

b) When do you apply the Strategy pattern ?

c) The TitledBorder class can give a title to a border. Consider the code

```
panel.setBorder(new TitledBorder(new BevelBorder(BevelBorder.RAISED), "Confirm Selection"));
```

What design pattern(s) are at work? Explain your answer. (a similar example is in the textbook/notes)

5.2.
You read the following (sparsely commented) Java code written by somebody else (in several files), with irrelevant details omitted:

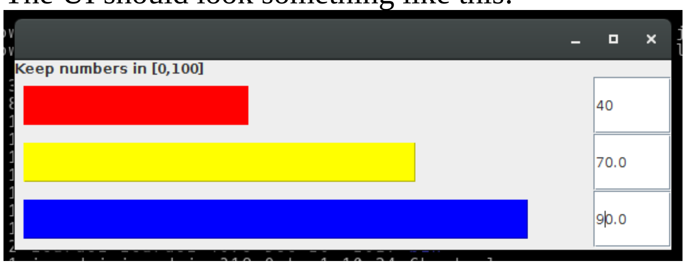
```
1 public interface Videoclip {
2    void play(JComponent comp);     // start playing the videoclip
3    void stop();                    // stop playing the videoclip
4 };
5
6
7 public class LectureVideo implements Videoclip {
8    public LectureVideo(LecturePDFFile notes, Date whenRecorded, Instructor who) {
9       // details not relevant
10   }
11
12   public void play(JComponent comp) {
13      // details not relevant
14   }
15
16   public void stop() {
17      // details not relevant
18   }
19 };
20
21
22 // class CaptionedVideo is a videoclip that also plays closed captions
23 // (stored in class CaptionInfo)
24 public class CaptionedVideo implements Videoclip {
25   public CaptionedVideo(Videoclip clip, CaptionInfo captionsInfo) {
26      this.clip = clip;
27      this.capInfo = captionsInfo;
28      // details omitted
29   }
30
31   public void play(JComponent comp) {
32      this.clip.play(comp);
33      this.capInfo.play(comp);
34   }
35
36   public void stop() {
37      this.clip.stop();
38      this.capInfo.stop();
39   }
40
41   private Videoclip clip;
42   private CaptionInfo capInfo;
43 };
44
45
46 public class Player {
47   public static void main(String[] args) {
48      // ...
49      CaptionInfo capinfo = ....;  // not relevant
50      LectureVideo lectureVid = ....
51      CaptionedVideo capLectureVid = new CaptionedVideo(lectureVid, capInfo);
52
53      JComponent wnd = .....
54      capLectureVid.play(wnd);
55      // ..... not relevant
56   }
57 };
```

a) Write the UML class diagram for the design used in the above code. Include multiplicity, attributes and operations.

b) What design pattern is at work in this application? Analyze the intent of the pattern and look at how objects interact. Explain your answer in detail.

5.3.
Write a program that displays editable bar graphs. The GUI has two horizontal panels.
The right panel contains textfields with numbers. The left panel contains corresponding horizontal bar graphs that graphically show the value of the numbers on the left. Use a rectangle with the width (in pixels) equal to the numbers from the textfield, but no higher than 100 pixels.

The UI should look something like this:



Use MVC and the Observer pattern. Store the numbers in the model. Attach the graph view as a listener. When the user edits one of the numbers, the number controller should update the model and the model should notify the graph view that a changes has occurred. Then, the graph should get the number from the model and repaint itself.

Hints: use method Textfield.getText() to extract the text from a textfield.
Add a key listener to each textfield with method
Textfield.addKeyListener(KeyListener l)

$$$

EXTRA CREDIT PROBLEM : 5 points
5.4
Write a program that displays a disk and three sliders. A JSlider object is a component that lets the user graphically select a value by sliding a knob within a bounded interval.
Each slider in the GUI is labeled with a color name in a JLabel object: red, green, and blue.
Each slider varies between min=0 and max=255. Together, the three sliders provide the RGB numbers necessary to create a Color object.
When the user changes a slider, the disk's RGB color must change accordingly.

Use the MVC architecture and the Observer pattern.
Register with each slider a ChangeListener object (Controller).
When the user moves a slider, the ChangeListener object notifies the model of the change.
The model gets the updated numbers from the 3 sliders and notifies the view.
The view object can be a JComponent subclass that displays the disk in function paintComponent().

JSlider doc:
http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/JSlider.html

Color doc:
http://java.sun.com/j2se/1.5.0/docs/api/java/awt/Color.html

Chapter 6

6.1. Answer these questions:
a) When should abstract classes be used ? Give a general answer and an example.

b) Explain when an abstract class cannot be used in a Java program and an interface is the only choice.

c) The Section class used in a text editor application includes a text attribute, contains a collection of objects of class Section or its subclasses, and has a method called *display* that displays the text of this section object and of all other sections referenced in its collection.
What design pattern does class Section implement ? Explain.

6.2
a) Explain in detail and with your own words why classes Rectangle2D, Rectangle2D.Float, and Rectangle2D.Double implement the Template Method design pattern.
Why is Rectangle2D an abstract class ?

b) Consider the SceneEditor example from Chapter 6, the 3rd version, at
http://wisenet.fau.edu/class/cop4331/notes/Ch6/code/scene3/.
Explain why the CompoundShape.add method is protected.
Explain why CompoundShape.path is private.

6.3
Consider the Employee class hierarchy from the beginning of Chapter 6.
Use the Template Method design pattern to implement a Template Method for the Employee class:

```
String toJson() {...}
```

that generates a JSON string with the employee's name, base salary (the salary field), and the total salary (as returned by getSalary()).

Example:

```
Employee sarah = new Employee("Sarah");
sarah.setSalary(50000);
Manager sandy = new Manager("Sandy");
sandy.setSalary(100000);
sandy.setBonus(1234);

System.out.println(sarah.toJson());
// prints {"class":"Employee", "name":"Sarah", "salary": 50000}
```

```
System.out.println(sandy.toJson());
// prints {"class":"Manager", "name":"Sandy", "salary": 101234, "bonus":1234}
```

a) Describe the mappings (table) from the pattern context to the Employee problem domain.

b) Show the UML class diagram. The diagram needs to describe relationship between the base class Employee and subclass Manager. Include attributes, and methods involved in implementing method toJson().

c) Write the Java code for classes Employee and Manager. Include a main() method in class Manager where you illustrate how toJson() is used for objects of both classes.

6.4
A flight simulator game involves several manned aircraft types, like single-engine prop airplane, helicopter, fighter jet.
A simulated flight consists the following operations, in this given order, for all aircraft types: 1. *takeoff*, 2. *fly*, 3. *land*.
Each aircraft type does these operations in a way that is very specific, as one would imagine.
Executing a simulation involves creating an aircraft object of a particular type, then running the flight scenario.

a) Use the Template Method design pattern to design the architecture of the flight simulator as described above. Include a class that runs the scenario. Create the necessary class hierarchy so that it's easy to add new aircraft types. Which is the abstract class ?
Write the UML class diagram. Make sure to include classes, interfaces (if any), relationships, attributes, and methods (with parameters).
Write the UML sequence diagram for the flight scenario.

b) Implement the architecture in Java.

NOTE: For this problem you are free to identify and name classes as you please. Follow the approach from Chapter 2. You will need a class that puts things together to run a simulation scenario.
Since we don't have sufficient information on how to actually simulate the phases of aircraft flight, we will simply use System.println("….") statements to fake them.