IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2019

MSc in Computing Science
for Internal Students of the Imperial College of Science, Technology and Medicine

PAPER C575

SOFTWARE ENGINEERING

Thursday 2nd May 2019, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

1. Design Patterns I

For this question, look at the code provided to you in the directory Q1.

Look at the code for classes FormattedList, HtmlList and LaTeXList.

A. What design pattern does this code follow?

B. What would be an alternative pattern for factoring out duplication between these list classes?

C. Change the code to implement this pattern instead and show how class FormattedList looks after the changes.

D. Which of these two solutions do you think is preferable, and why?

The four parts carry, respectively, 10%, 10%, 60% and 20% of the marks.

2. Design Patterns II

For this question, look at the code provided to you in the directory Q2.

Look at the code for class HttpRequest, which has quite a few constructor parameters, not all of which are always needed.

A. Name a pattern to make constructing different configurations of an HttpRequest more convenient and elegant.

B. Implement that pattern and show the new code you wrote.

C. Show how a call to your new code would look to create a POST request to http://exams.imperial.ac.uk/575 with body "mark=100" and a header "Date=02-05-2019".

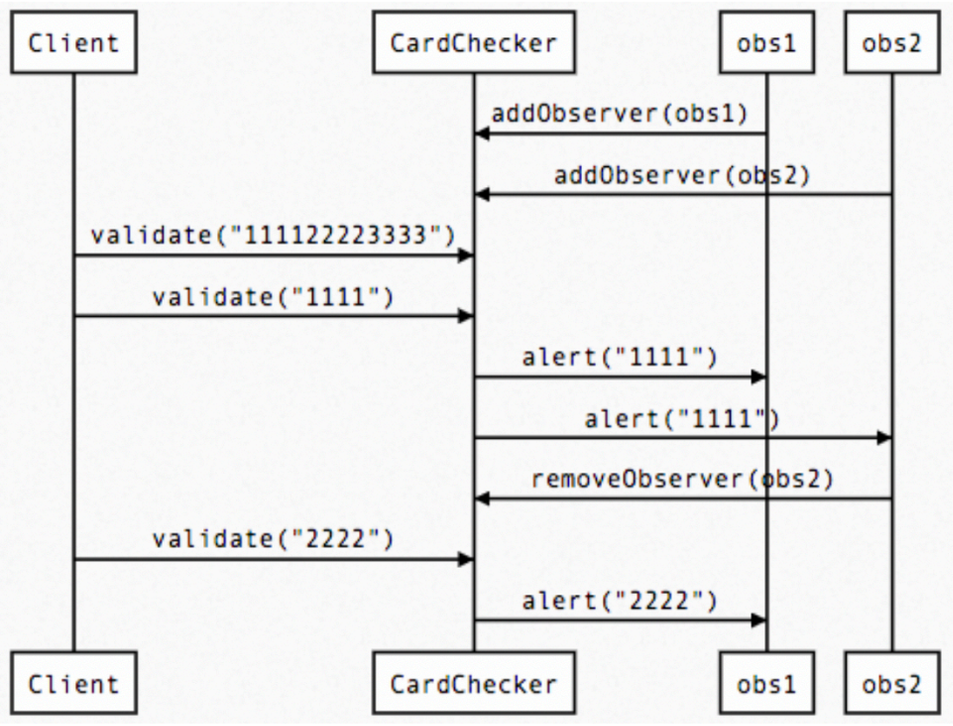
The three parts carry, respectively, 10%, 60% and 30% of the marks.

3. Test Driven Development

For this question, look at the code provided to you in the directory Q3.

Look at the Sequence Diagram below. Using Test-Driven development and mock objects, iteratively develop an implementation of CardChecker according to the scenario illustrated.

The CardChecker notifies all current observers whenever an invalid card number is entered. For this question assume that any card number with 12 digits is valid - all others are invalid.



A. Show your first test and the implementation to make it pass

B. Show your second test and the implementation to make it pass

C. Show your third test and the implementation to make it pass

The three parts carry, respectively, 40%, 30% and 30% of the marks.

4 **Interactive Applications**

For this question, look at the code provided to you in the directory Q4. This code implements a very small GUI statistics app. As you click buttons on the UI, aggregate statistics about the numbers entered are presented.

A) Give two benefits of using a Model-View-Controller architecture for a GUI application.

   i) Explain the first benefit.

   ii) Explain the second benefit.

B) Refactor the code in the Q4 directory to follow the Model-View-Controller architecture. Do not worry too much about totally separating view from controller, most important is the separation of the model.

C) Write at least one unit test for the model in your refactored code.

The three parts carry, respectively, 20%, 60% and 20% of the marks.

Appendix of Code Samples.

These will be given to the candidates on their computer - not needed as part of the printed examination paper.

(Intentionally left blank)

Q1

```
public abstract class FormattedList {

    private final List<String> content = new ArrayList<>();

    public FormattedList(String... items) {
        content.addAll(Arrays.asList(items));
    }

    public void add(String item) {
        content.add(item);
    }

    public void print() {
        System.out.println(formatHeader());
        for (String item : content) {
            System.out.println(formatItem(item));
        }
        System.out.println(formatFooter());
    }

    protected abstract String formatHeader();
    protected abstract String formatItem(String item);
    protected abstract String formatFooter();
}

public class HtmlList extends FormattedList {

    public HtmlList(String... items) {
        super(items);
    }

    @Override
    protected String formatHeader() {
        return "<ul>";
    }

    @Override
    protected String formatItem(String item) {
        return "  <li>" + item + "</li>";
    }

    @Override
    protected String formatFooter() {
        return "</ul>";
    }
}

public class LaTeXList extends FormattedList {

    public LaTeXList(String... items) {
        super(items);
    }

    @Override
    protected String formatHeader() {
        return "\\begin{itemize}";
    }

    @Override
    protected String formatItem(String item) {
        return "  \\item " + item;
    }

    @Override
    protected String formatFooter() {
        return "\\end{itemize}";
    }
}
```

Q2

```
public class HttpRequest {

    public enum Method {
        GET,
        PUT,
        POST,
        HEAD,
        DELETE
    }

    private final String url;
    private final String body;
    private final List<String> params;
    private final Method method;
    private final List<String> headers;

    public HttpRequest(String url, Method method, List<String> params, List<String>
headers, String body) {
        this.url = url;
        this.body = body;
        this.params = params;
        this.method = method;
        this.headers = headers;
    }

    // more code here - not relevant for exam question

}
```

Q3 - (Only the starting point for the tests)

```
public class CardCheckerTest {

    // implement your tests here
}
```

Q4

```
import java.awt.Panel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class SimpleStats {

  private final List<Integer> numbers = new ArrayList<>();
  private int max;
  private double mean;

  private void display() {

    JFrame frame = new JFrame("Simple Stats");
    frame.setSize(250, 350);

    Panel panel = new Panel();

    JTextField currentMax = new JTextField(11);
    JTextField currentMean = new JTextField(11);

    panel.add(new JLabel("Max: value "));
    panel.add(currentMax);
    panel.add(new JLabel("Mean: value "));
    panel.add(currentMean);

    for (int i = 1; i <= 12; i++) {
      final int n = i;
      JButton button = new JButton(String.valueOf(i));
      button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
          numbers.add(n);
          max = Math.max(max, n);
          mean = numbers.stream().mapToInt(val -> val).average().orElse(0.0);
          currentMax.setText(String.valueOf(max));
          currentMean.setText(String.valueOf(mean));
        }
      });
      panel.add(button);
    }

    frame.getContentPane().add(panel);

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }

  public static void main(String[] args) {
    new SimpleStats().display();
  }

}
```

General Information and Computer Instructions

When the exam starts, log in to the Linux computer in front of you, which is in Lexis mode. Use your College username as both your username and password to log in to Lexis. Don't worry if it is slow to start.

AnswerBook

All your answers should be submitted electronically by accessing the *AnswerBook* website at `https://co575.doc.ic.ac.uk/exam` using a standard web browser from the Linux environment. All other access to the network is intentionally blocked. Log in to this website using your normal College username and password (not username twice as with Lexis). You should not use a paper answer booklet.

Refer to the exam paper for the full version of the exam questions.

Skeleton Code

On the computer you will find a directory `/exam/co575-exam` containing code related to each of the questions in the exam. Work on this code on the computer using IntelliJ IDEA.

Using IntelliJ IDEA

You can start IntelliJ from the menu, or by typing `idea &` in the terminal. The skeleton files are set up with an IntelliJ project configuration, so: **click *Open* (not *Import Project*)**, navigate to the `/exam/co575-exam` directory, and click OK.

The lib folder contains all the testing libraries that we used during the course. These should be added into your IDE automatically if you open the project as above. If no JDK is set, select JDK 10. Please ask an invigilator if you have problems with opening the project.

There is no `build.sh` or suite of automated tests/checks for you to pass, but do aim to use good code style and clear formatting in your solutions.

You do not have access to GitLab, but you can use a Git repository locally on your machine if you find it helpful.

When you are happy with your work, paste your code into the relevant boxes on the *AnswerBook* site to answer the questions.

Click the button in the top corner to save your changes.

Make sure all changes are saved before the end of the exam.

At the end of the exam, just log out from your machine.