Family Name _____

First Name _____

Student Number _____

Venue _____

Seat Number _____

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

No form of communication or collaboration allowed during exam.

Computer Science and Software Engineering

EXAMINATION

Mid-year Examinations, 2020

SENG301-S20S1 (C) Software Engineering II

Examination Duration: 180 minutes

Exam Conditions:

Open Book exam: Students may use all (electronic) resources supplied during the course (*i.e.* lecture notes, additional readings, lab material).

Instructions to Students:

Answer all questions. Please write in a readable and intelligible manner, unreadable answers will be discarded.

Read the entire question before answering any part of it. A few questions have associated figures or notes that are crucial to the question.

Check carefully the number of marks allotted to each question. This suggests the degree of detail required in each answer, and the amount of time you should spend on the question.

No form of collaboration is permitted:

You **may not communicate** with anyone during the exam, and you **may not post material** relating to the exam online. This is simply a matter of ethical behaviour.

Your online activity leaves a **footprint we eventually can trace**. Stay ethical and keep your integrity as you sit the exam by making sure everything you do is your own work.

In the unlikely event you would decide to collaborate in any form and we would have any form of suspicion, **we may contact you for an additional oral exam**.

If we have sufficient doubts on the intellectual integrity of your exam answers, **we may open a case to the Registrar** with the potentially detrimental academic consequence this may lead to (*e.g., failing the course, exclusion of the University*).

For Examiner Use Only

Question	Mark

Total _____

Academic integrity:

Academic integrity is a principle at the University of Canterbury whereby both staff and students agree to act honestly, fairly, ethically and with respect for each other in teaching and learning.

For some students, there may be increased temptation to cheat and engage in dishonest academic practices such as:

Plagiarism using someone's ideas and information without acknowledging them as the source;
Self-plagiarism where someone attempts to submit their own writing to two different assessments to gain credit twice;
Collusion copying the work of someone else or allowing someone else to copy your work without disclosing this with the intent to deceive;
Impersonating/Ghost writing having another person or commercial organisation impersonate you and complete an assessment item on your behalf;
Fabrication "inventing", data for example in a lab report or from a publication.

Declaration:

By **taking part to** and **completing** this examination,

1. I confirm that I have **read and understood the rules** of the exam and the **Academic integrity** principle.
2. I understand that the **exam is considered confidential** and I must not discuss the exam materials until the exam is over for all students.
3. I understand that **failure to comply** with these requirements may mean that **the matter will be referred to** the Head of Department, Dean or **Proctor** as appropriate for disciplinary action.

Instructions for submissions (visible on Learn):

You are required to submit the following files separately:

- all your answers into a document file named `Fullname_studentID_answers` of any of the following types: `.doc`, `.docx`, `.odt` or `.pdf`;
- any pictures or drawings as `.png`, `.jepg`, `.jpg` or `.pdf` alongside your answers;
- submit on Learn, **`https://learn.canterbury.ac.nz/mod/assign/view.php?id=1414837`**.

An extra **10 minutes** is given to upload on time. Therefore,

- **no late submissions** will be considered;
- **upload before the scheduled time (12:30PM)** to avoid any last minute stress;
- the cut-off time is **12:40PM**. We **will not accept** any submissions by any other mean than Learn (e.g., email).

As UML drawing means, we recommend the following:

- web- and textual-based *PlantUML*, **`https://plantuml-editor.kkeisuke.com/`**
- *dia*, **`http://dia-installer.de/download/index.html`**
- *umbrello*, **`https://umbrello.kde.org/installation.php`**
- *argoUML*, **`https://github.com/argouml-tigris-org/argouml`**
- pen and paper, and a decent camera or scanner.

Questions regarding the procedure can be asked on: **`https://learn.canterbury.ac.nz/mod/forum/discuss.php?d=416322`**.

Questions Start on Page 3

Question 1 Continuous integration and deployment

10 marks for whole question

Please read below excerpt from a transcript of an interview at *MyDevCorp* where they explain part of their continuous development strategy.

> "We are using a decentralised version control system, such as `git`. We use a scrum sprint backlog to keep track of our work to do. For every story that has been picked up for the current sprint, developers pull the latest development branch into the corresponding feature branch (or create a new one if that feature has not been worked on earlier). They start working in pairs to implement all tasks linked to a feature during the sprint and merge the feature branch to the development branch close to the end of the sprint. Manual tests is carried as the stories are implemented in isolation on each feature branch.
>
> When all the merging tasks starts, all development stops and each pair usually takes half a day to merge the code successfully. A round-robin approach is used for all pairs to merge in turn into the development branch. When all features are merged, each pair starts manually testing the features of the other teams following the manual test plan that is maintained throughout the sprint. If a story and/or feature is discovered to be buggy, it is commented out in the code and will be reworked during next sprint."

This strategy has at least **5** flaws that are either technical (*i.e.* the way the sources are managed) or methodological (*i.e.* the way the team works). Identify them and for each flaw, give actual advice on how to **improve the current strategy**. Focus your answer on the flaws and give solutions on how to improve them, but **do not re-explain a git flow**.

Question 2 Agile Software Development

10 marks for whole question

Question 2.1 Backlogs

5 marks

Explain the two different types of backlogs in a Scrum project. For each of them, explain what they contain by stressing their differences.

Question 2.2 Retrospection

5 marks

What are the goals of a *Retrospective meeting*? What is the expected output of such meeting? How should the output of that meeting be managed on the long run?

Question 3 Code quality and reviews

10 marks for whole question

Here is some sample code. Take a close look at it before answering the next questions.

```
1  public long longestCommonMultiple(int a, int b) {
2    List<Integer> multA = new ArrayList<>();
3    List<Integer> multB = new ArrayList<>();
4    int max = a * b;
5    int current = 0;
6    while (current <= max) {
7      multA.add(a * current);
8      multB.add(b * current);
9      current++;
10   }
11   for (Integer integerA : multA) {
12     for (Integer integerB : multB) {
13       if (integerA.equals(integerB)) {
14         return integerA;
15       }
16     }
17   }
18   return max;
19 }
```

Question 3.1 Unit tests 5 marks

Specify the unit tests you would write for this method. You can express the tests either in plain English or in a mathematical form.

Question 3.2 Code review 5 marks

Write a review for this piece of code as you would do when assessing a merge request.

Question 4 Weekly readings

5 marks for whole question

In week 3 reading, Michaela Greiler discusses some social factors that need to be taken into account when conducting code reviews.

- What are code reviews for?
- Why the way the feedback is given is as important as the feedback itself?

Question 5 Design patterns

36 marks for whole question

You are asked to help design software to control robot vacuum cleaners (RVCs). These operate throughout an environment (which might be anything from a room to a whole building). Some of their activity is autonomous and some is directed by instructions and guidance from the software system. Each RVC has a number of features including sensors, a power system, and a cleaning system. The following design notes provide an overview of the main system features. Your task is to use your knowledge of GoF design patterns to produce a design for the system.

- There's lots of different kinds of RVC — some are for domestic or commercial use while others are for industrial applications.
- Performance and manoeuvrability is affected by the RVCs weight. Small animals, such as cats and budgerigars, like to ride on an RVC and accessories, such as a large capacity bag, may be fitted.
- The system can be monitored in several ways. For example, a web app can provide a real-time display of the system status. It is expected that further monitoring tools, including a mobile app, will be developed in future.
- Coordinated RVCs can operate in groups. This is particularly useful when several RVCs are operating in the same room.
- The software will only ever control RVCs in one building.
- When it is powered on, each RVC formats a report, including its configuration and the state of each of its components, and emails it to the maintenance department. Each kind of RVC will produce a different report.
- The environment's cleaning supervisor can specify high priority tasks for the RVCs to perform. For example, a specialised RVC might be assigned to polish the executive washroom floor. A high traffic area should be cleaned as a high priority. These tasks will be carried out first before the batteries run out. When an RVC has completed a high priority task it logs the task details and time stamp.

Question 5.1 23 marks

For each GoF pattern you identify, name it and show how the elements of the pattern (classes, methods, ...) correspond to the elements of your design. Use the tabular format described in lectures for this. You may also add further explanation if you wish.

Question 5.2 10 marks

Sketch a UML class diagram of your design — you may use suitable software tools if you wish. Your diagram should focus on showing the design patterns present and the ways they are combined. You should include the major elements (classes, interfaces, associations, methods, ...) but may omit detail not directly relevant to the patterns used. You may find it helpful to use the guillemet stereotype notation (e.g. $\ll Iterator \gg$) to clarify pattern roles.

Question 5.3 3 marks for subquestions

An RVC has limited carrying capacity. If a large capacity bag is fitted, then there is room for a budgerigar to ride but not a cat. An RVC with a regular bag can carry a cat or up to 3 budgerigars.

Question 5.3.1 1 mark

What term would best describe such additional requirements?

Question 5.3.2 2 marks

Does your current design allow for these additional requirements to be satisfied? If so, briefly explain. If not, briefly explain why not and suggest a possible refactoring.

Question 6 User stories

5 marks for whole question

Using the domain of the *MyRecipe* app developed throughout term 1, take a look at following story.

"As a user, I want to share my own recipes on social media like Facebook and Instagram so that my friends can get inspired by my recipes."

What principles of the INVEST rule are violated in the above user story? For each principle, state if it is fulfilled or not and justify why you think so.

Question 7 Design principles and code smells

3 marks for whole question

In your own words, clearly and concisely explain what *large class smell* is. Imagine you are given an unfamiliar Java code base and asked if you think it contains any classes with *large class smell*. Briefly explain how you would identify any classes which might have that smell and describe how you would determine whether or not they actually did.

Question 8 Definitions and concepts

10 marks for whole question

Answer to the following questions in a concise way. Do not use bullet points. A typical answer should be formed by 2 to 3 sentences.

Question 8.1 Requirements 2 marks

How are *"persona"* and *"user profile"* identified? What is their common goal?

Question 8.2 Bad behavioural pattern 2 marks

Why coming to a stand-up unprepared is a bad behavioural pattern? Relate to (one or more) Scrum value/s.

Question 8.3 Resilience 2 marks

The *"visibility"* principle (also known as the *need-to-know*) is critical when you write code. What is it? Why is it important?

Question 8.4 Metrics 2 marks

What is a *"sprint interference"* chart? Why is it used for?

Question 8.5 Delivery 2 marks

What are *"smoke tests"*? When do we typically perform them?

Question 9 Contracts

11 marks for whole question

You are asked to advise on the design of a software system that controls music players. Part of the system is shown in the class diagram of Figure 1.

- MP3Player has a play() method. If the media source is available and an album has been selected, then calling this method plays all tracks on the album in order.
- PayPerPlay is used in jukeboxes: it also has a play() method. If the media source is available, an album has been selected, and a coin has been inserted, then calling this method plays all tracks on the album in order until the album is finished or 20 minutes have elapsed.
- ShufflePlayer also has a play() method. If the media source is available, then calling this method plays all tracks on the album in order until the album is finished and then plays all other albums by the same artist.

Question 9.1 2 marks

In your own words, clearly and concisely describe what is meant by the contract of a method. How are contracts expressed? What, if any, relationships should there be between a method and one which overrides it (such as play() in Figure 1)?

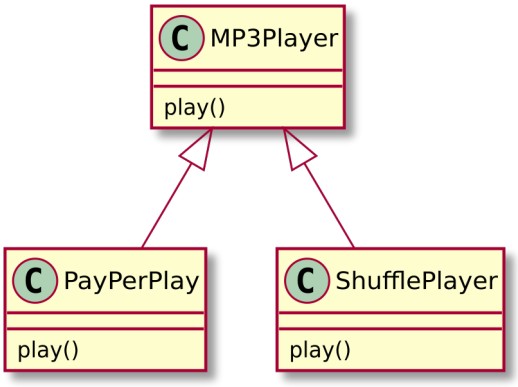

Figure 1: Part of the MP3 player system

Question 9.2 6 marks

What are the contracts of the three play() methods in Figure 1? You should focus on the key points: these are more important than minor detail and formal notation.

Question 9.3 3 marks

Use your knowledge of contracts to comment on the quality of the design. Briefly explain any good and bad features. If you identify any problems, briefly indicate how they might be fixed.

— END OF EXAMINATION —