COMP2511 21T2 Final Exam

There are three parts,

- *Part 1: Multiple Choice (20 marks)*.
- *Part 2: Short Answer (25 marks)*.
- *Part 3: Programming Questions (55 marks)*.

Change Log

- *Part3 Q3)* Changed part3q3/types to just part3q3 in filenames
- *Part3 Q2)* Fixed assertEquals for refactoring function (just redownload zip or look at ED)

Exam Conditions

- You can start reading the exam at **09:00 Tuesday 17 August 2021** Sydney time (AEST).
- You can start typing at 09:00 Tuesday 17 August 2021 Sydney time (AEST).
- You have until **12:00noon Tuesday 17 August 2021** Sydney time (AEST) to complete this exam.
- Only submissions before 12:00noon Tuesday 17 August 2021 Sydney time (AEST) will be marked.
- Except, students with extra exam time approved by Equitable Learning Services (ELS) can make submissions after 12:00noon Tuesday 17 August 2021 Sydney time (AEST) within their approved extra time.
- You are **not permitted to communicate** (email, phone, message, talk, social networks, etc.) with anyone during this exam, except COMP2511 staff.
- Again, please note that you are not permitted to get help from anyone except COMP2511 staff during this exam.
- You are **not permitted to access web pages or other internet resources**, except the following web pages (for the exam, the lecture notes and documentation):
    1. you can access and read this exam paper!
    2. you can access the course material available on the course webpage and your course work (your tut/lab solutions, assignment and project work).
    3. you can also access Java API available online at https://docs.oracle.com/en/java/javase/11/docs/api/index.html .
- You are **not** permitted to access any other papers, books or computer files, except for the following:
    - this exam, your tut/lab solutions, assignment and project work for this course.
- You are **not** permitted to use code-synthesis tools such as GitHub Copilot and other similar tools.
- Importantly, please make sure that you **submit your original work** and **don't copy!** We will use sophisticated plagiarism software/techniques to detect any possible breaches.
- Even after you finish the exam, on the day of the exam, on the day of the exam do **not** communicate your exam answers to anyone. Some students have extended time to complete the exam.
- Do **not** place your exam work in any location, including file sharing services such as Dropbox or GitHub, accessible to any other person.
- Ensure during the exam **no other person** in your household can access your work.
- Your zpass should **not** be disclosed to any other person. If you have disclosed your zpass, you should change it immediately.
- **Deliberate violation of exam conditions will be referred to Student Integrity as serious misconduct.**

- **During the exam:**
    - if you have a question or need clarification during the exam, please post a **PRIVATE post on the Ed forum**. Do not send a message to the lecturer (or tutors), you may not get a response. When posting a message to the forum, it's important that you provide all the required details. Failure to do this may result in delays in responding to your queries.
    - if there is a correction, we will post a notice on the class webpage. So please check your email and also check the class webpage for possible corrections (if any) during the exam period.

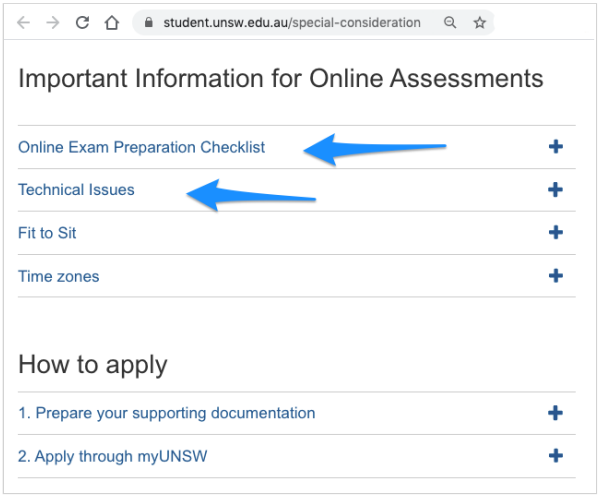
Notes:

- Answer **all** questions. Questions may **not** be worth equal marks. Questions may be answered in any order.
- All answers must be submitted online using the provided instructions in the respective questions.

- Please note that the topics covered in the final exam **may be different** to the topics covered in the *Practice Exam* structure and/or *Practice Questions*. Also, the marks distribution across topics may also vary.
- You do **not** need to draw any UML diagrams for this exam.

Important Information for Online Assessments

Before your final exam, you must read the section "*Important Information for Online Assessments*" available on the Special Consideration webpage. It offers information on what you should do if you experience a technical issue that is beyond your control and impacts on your ability to complete an assessment, and the other related topics. In particular, how and what to document for a special consideration application.



Please also read the check list provided by UNSW Student Services & Systems, **click here** .

During the Exam

If you have a question or need a clarification during the exam, you can make a **PRIVATE post on the Ed forum**. Do not send a message to the lecturer (or tutors), you may not get a response.

When posting a message to the forum, it's important that you provide all the required details. Failure to do this may result in delays in responding to your queries.

To ensure that you are as prepared for your online exam as you can be, make sure you check each point listed in the "Online Exam Preparation Checklist". In particular, read the following sections:

- Fit to Sit
- Technical Issues
- Communication during the exam
- Sharing answers with others or posting them online

Part 1: Multiple Choice (20 marks)

There will be 10 multiple choice questions. Each question is worth two marks. Part 1 uses Moodle Quiz module. You can attempt multiple times. Your final selections will be marked. Make sure to "**Finish attempt**" and also "**Submit all and Finish**".

Go to Part 1: Multiple Choice on Moodle (questions 1 to 10)

Part 2: Short Answer (25 marks)

Part 2 uses Moodle Short Answer module. You need to submit your answers following the required instructions on Moodle.

Please carefully read the instructions provided in the questions. For example, you may be required to briefly explain/justify one or two or three most important points in your answer. You do not need to write a long description in your answer.

Go to Part 2: Short Answer on Moodle (questions 11 to 16)

Part 3 (of 3): Programming Questions (55 marks)

You can use VLab or your own machine. You need to submit your answer files using the provided instructions in the respective questions (using the provided give command or via WebCMS). All answers must be submitted online.

Question One (16 marks)

Download the following zip file, it contains the starter code for this question.

https://cgi.cse.unsw.edu.au/~cs2511/21T2/FinalExam/21T2_Part3_Q1.zip

If you are on CSE you can also use the following command:

```
unzip /web/cs2511/21T2/FinalExamResources/21T2_Part3_Q1.zip
```

Back when people used to hold meetings and conferences in person, organisations would need to book out large venues to hold the conference. UNSW with its large lecture theatres is an excellent space for this, and you have been asked to write their Conference Room booking system ready for when people return to campus, ensuring COVID safety by tracking room capacity limits. The system has the following requirements:

- A room has a name, a room ID and a capacity
- The cost of hiring a room (in dollars) is the capacity multiplied by 10 per day
- Bookings can be made for either a single room or multiple rooms
- The bookings have a start date and time and an end date and time
- When the booking is created, the number of people in the conference is specified. If the number of people in the conference exceeds the total room capacity, the booking fails.
- A room cannot be booked out at the same time by two bookings
- A booking should be able to be cancelled
- Bookings can optionally include zero or more of the following services:
    - Lunch, which adds an extra $15 per person
    - Dinner, which adds an extra $25 per person
    - Multimedia Facility, which adds $500.

Model an Object-Oriented Design for a possible solution. If suitable, you must use a Design Pattern(s) taught in the course in your solution.

You will need to provide the following:

- Interfaces (with brief Javadoc comments)
- Classes (with brief Javadoc comments) and
- Method signatures (with brief Javadoc comments)
- In a file `Q1.txt`, write a brief rationale explaining your design choices, examples of adhering to design principles taught in the course, any reasonable **assumptions** your design makes about the requirements, and a justification for any Design Patterns you may use in your solution.

For this question:

- You do NOT need to implement methods; you are simply writing class and method declarations as stubs.
- You do NOT need to draw a UML diagram.

You need to submit the required files Q1.txt, *.java using the following give command

```
2511 submit exam q1 Q1.txt *.java
```

or via WebCMS.

Question Two (12 marks)

Download the following zip file, it contains the starter code for this question.

https://cgi.cse.unsw.edu.au/~cs2511/21T2/FinalExam/21T2_Part3_Q2.zip

If you are on CSE you can also use the following command:

```
unzip /web/cs2511/21T2/FinalExamResources/21T2_Part3_Q2.zip
```

A service provider offers checkout services to the two most popular supermarkets in Australia: Coles and Woolworths.

The starter code provided models a fully correct software solution for the checkout process. However, the code has significant design flaws.

You need to do the following:

- In a file `Q2.txt`, briefly describe the Design Smells you identified, justify a series of refactoring choices both at low-level (code level) and high level (use a suitable design pattern, if required).
- Refactor the code to remove the Design Smells, and if required, use a design pattern(s) discussed in the course for refactoring.

For this question:

- We will provide a complete suite of tests for you. After refactoring, the behaviour of the system should remain unchanged (i.e. the tests should still pass unchanged)

You need to submit the required files Q2.txt, *.java using the following give command

```
2511 submit exam q2 Q2.txt *.java
```

or via WebCMS.

Question Three (15 marks)

Download the following zip file, it contains the starter code for this question.

https://cgi.cse.unsw.edu.au/~cs2511/21T2/FinalExam/21T2_Part3_Q3.zip

If you are on VLAB you can also use the following command:

```
unzip /web/cs2511/21T2/FinalExamResources/21T2_Part3_Q3.zip
```

Typically, in commercial programs, we want to carry out actions only when a set of rules are true. As a shorthand, these are called *business rules*.

Let's assume we have the following operators and variables available:

Operators:

- group operators (`AND`, `OR`).
- comparison operators (`IS NOT BLANK`, `IS GREATER THAN`).
- transformation operators (`/`, `*`).

Variables:

- for example: `"email"`, `"phoneNumber"`, `"mark"`, `"responses"`, `"invites"`, etc.
- variables can be one of the following types: Double, Integer, String.
- variables are looked up using a `Map<String, Object>` a link to the Javadoc's for Map is below. If the Map doesn't contain the variable, its value is null.

https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

Possible business rules could be as simple as below:

(1) The following is true if `"email"` variable is not blank (blank meaning that it consists purely of whitespace, is empty string, or is null), or `"phoneNumber"` is not blank.

```
("email" IS NOT BLANK) OR ("phoneNumber" IS NOT BLANK)
```

(2) The following is true if *"responses"* variable is greater than the *"invites"* variable divided by 2 and either *"email"* is not blank or *"phoneNumber"* is not blank.

```
("responses" IS GREATER THAN ("invites" / 2)) AND (("email" IS NOT BLANK) OR ("phoneNumber" IS NOT BLANK))
```

Business Rules always evaluate to a `boolean` value. For simplicity, we also only support very few transformations/operators in this example (the ones stated above). All transformations/groups/operator's behaviour is explained in detail in the files

- part3q3/BusinessRuleGroupTypes.java
- part3q3/BusinessRuleOperators.java
- part3q3/BusinessRuleTransformations.java

For simplicity, we construct rules using a factory (which you need to implement, a skeleton is provided). Please note that you don't need to parse rules.

▶ BusinessRulesTest.java

Your task is to design a solution that allows a user to create arbitrary rules as shown above. You must design your solution using one or more of the design patterns discussed in the course (**in addition** to the factory pattern provided) such that it could be easily extended (for additional operators).

You must also provide brief justification for your design choices in Q3.txt. Please note that it doesn't have to be very long, just a paragraph should suffice. You will be awarded marks for proper justifications.

For this question:

- You need to **implement** methods in BusinessRuleFactory.java. You've been provided a basic factory outline in BusinessRuleFactory.java, please do not change the signatures of any of these methods.

- As per your design choice(s), you need to **implement** the required classes. If you prefer, you can add interfaces and Enums.

Please make sure that you satisfy the following criteria:

- All files created must be in the same package that is `package part3q3;`
- You are allowed to modify the Enum members to give them constructors, but you are not allowed to add new members or remove existing ones. I.e., the code written in the tests needs to still compile.
- You are not allowed to use any libraries outside of the Java standard library.
- **Importantly**, if you don't use a suitable design pattern in your solution, as discussed in the lectures, you will not be awarded marks, even if you pass the tests.
- We will run additional tests on your code. To ensure your solution passes our additional tests, you will need to write your own tests that account for cases which the given tests do not account for.

You need to submit the required files Q3.txt, *.java using the following give command

```
2511 submit exam q3 Q3.txt *.java
```

or via WebCMS.

Question Four (12 marks)

Download the following zip file, it contains the starter code for this question.

https://cgi.cse.unsw.edu.au/~cs2511/21T2/FinalExam/21T2_Part3_Q4.zip

If you are on VLAB you can also use the following command:

```
unzip /web/cs2511/21T2/FinalExamResources/21T2_Part3_Q4.zip
```

The class `Cycle` represents an ordered, infinitely repeating sequence of items of type `E`. Henceforth in this question, we denote a cycle as a finite sequence of elements separated by commas within angle brackets, which will be repeated infinitely in the denoted cycle. For example:

```
<1, 2, 3> represents the infinite-length cycle ..., 1, 2, 3, 1, 2, 3, ...
```

A `Cycle` is instantiated with a `List` of items of finite size, which will be repeated infinitely. This means that shifting the cycle to the left or right will still result in an equal cycle. For example, the following are equal cycles:

```
<1, 2, 3> is equal to <3, 1, 2>
<5, 6>    is equal to <6, 5, 6, 5, 6, 5, 6, 5>
```

Additionally, a cycle which can be obtained by "simplifying" another cycle is an equal cycle. A cycle is considered fully simplified when its internally stored finite-length list of elements is as short as possible, and repeating this internal list infinitely forms an equal cycle to the original cycle. For example, the following are equal cycles:

```
<1, 2, 1, 2> is equal to <1, 2>
<1, 2>       is equal to <1, 2, 1, 2>
<1, 2, 1, 2> is equal to <2, 1>
<2, 1>       is equal to <1, 2, 1, 2>
<1, 2, 1, 2> is equal to <1, 2, 1, 2, 1, 2> (since both are equal to <1, 2>)
<1, 2, 1, 2> is equal to <1, 2, 1, 2, 1, 2, 1, 2>
```

And so on...

An iterator generated for a `Cycle` should repeat infinitely if the `Cycle` is non-empty. If the cycle is empty, calling `iterator.next()` should throw a `java.util.NoSuchElementException`, and calling `iterator.hasNext()` should return `false`. If the cycle is non-empty, calling `iterator.next()` should return the next item in the cycle (wrapping around to the beginning if the end of the `Cycle` is reached), and calling `iterator.hasNext()` should return `true`. You will need to implement your own implementation of the `java.util.Iterator` interface and return an instance of this from your `Cycle.iterator` method:

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Iterator.html

Your task is to implement the following methods in the `Cycle` class:

- The `iterator` method (2 marks)
- The `equals` method (10 marks). **You may find this question challenging. We recommend leaving this towards the end of the exam and to do it only if you have time.**

As part of completing the iterator method above, you should also implement the methods marked as TODO in the file CycleIterator.java.

For the `equals` method, you MUST ensure the contract of `Object.equals` is adhered to.

Hint: when comparing two `Cycles` to check if they are equal, you may wish to calculate a fully simplified finite-length sublist of each cycle, such that if this simplified sublist is repeated infinitely it forms a cycle equal to the cycle it was generated from. For example:

```
A fully simplified, finite-length sublist of the cycle <1, 2, 1, 2> is    [1, 2]
A fully simplified, finite-length sublist of the cycle <1, 2, 1, 2, 5> is [1, 2, 1, 2, 5]
```

You should read the provided file `TestCycle.java` to understand how `Cycle` objects are created and tested. Make sure that your solution successfully passes the tests in `TestCycle`. You should thoroughly test your solution for additional test cases - we will be using different test cases to test your submission.

You do not need to write a rationale for this question.

Please make sure that you satisfy the following criteria:

- All files created must be in the same package that is `package part3q4;`
- You are not allowed to use any libraries outside of the Java standard library.
- We will run additional tests on your code. To ensure your solution passes our additional tests, you will need to write your own tests that account for cases which the given tests do not account for.

You need to submit the required files Cycle.java, CycleIterator.java using the following give command:

```
2511 submit exam q4 Cycle.java CycleIterator.java
```

or via WebCMS.

End
