The University of Melbourne
School of Computing and Information Systems

Final Exam

Semester 1, 2020

SWEN20003 Object Oriented Software Development

Exam Duration: 2 hours

Total marks for this paper: 100

This paper has 9 pages

Continued overleaf ...

Instructions to students:

- The exam has 4 questions across 3 sections, and all questions must be attempted.
- This is an online, open-book exam and you will have access to your subject material and Internet resources.
- You are required to provide your own answers to questions; copying directly from subject material or Internet sources is not acceptable and will be considered plagiarism.
- The mark allocation for each question is listed along with the question. Please use the marks as a guide to the detail required in your answers while keeping your answers concise and relevant. Point form is acceptable in answering descriptive questions. Any unreadable answers will be considered wrong.
- Worded questions must all be answered in English, and code questions must all be answered in Java.
- Please follow the instructions below to complete the exam:
  - Clone the solution template from the repository `<username>-final-exam` in `gitlab.eng.unimelb.edu.au` using the appropriate: `git clone` command.
  - In this repository you will find a single file named `<username>-final-exam.txt`. You are required to provide the answers to questions in the appropriate section of this file, through typing them in; all answers must be in this file and no other files should be added to the repository. You are not permitted to upload hand-written scanned files, and such solutions will receive zero marks.
  - Upon attempting each sub-question commit the changes to your files (answers) using:
    `git add .` and
    `git commit -m '[commit message]'`
    performed from the cloned folder in the above step.
    e.g. Once you complete Question 1(a) you can execute the following commands to commit the changes:
    ```
    git add .
    git commit -m 'completed question 1(a)'
    ```
  - You may choose to do `git push` as often as you would like as a way of backing up, but are not required to push until you complete the exam.
  - At the end of the exam, you are also required to submit the solution file, `<username>-final-exam.txt`, as a Canvas Turnitin Assignment submission.
    Note: It is important to carefully follow the instructions above. Abnormal commits will receive scrutiny because it could indicate possible plagiarism.
- Your lecturers will be available online via Zoom, during the first 15 minutes of the exam (Zoom link will be made available via Canvas). During this time you may ask clarification questions. You will be allowed to start answering questions during this time if you choose to do so (this is not a strict reading time as in normal written exams).

This paper will not be reproduced and lodged with Baillieu Library.

Continued overleaf . . .

1 Short Answer (20 marks)

Question 1. **(20 marks)**

Answer the following questions with **brief**, **worded** responses. Your answers should contain no more than **four** dot points, **not** essays.

a) Explain two forms of polymorphism supported in object-oriented software development, using an example for each form from your project (in your answer, you are allowed to include code snippets directly from your own project but the explanations must be your own). (4 marks)

b) What is **open-closed principle** in design? Demonstrate how you have applied this principle in your designs using an example from your project (in your answer, you are allowed to include code snippets directly from your own project but the explanations must be your own). (4 marks)

c) Explain how the event-driven programming paradigm, used for error handling in object-oriented programming (through the use of Exceptions), has enabled better error handing, compared to how it is done in non object-oriented programming. Your answer must provide justification based on the four advantages of the event-driven programming paradigm taught in the lecture. (6 marks)

d) Describe the **purpose** and **behaviour** of the following stream pipeline. Give a **real-world example** where you might use this code. Be sure to address each line of code in your answer. (6 marks)

```
List<String> tweetUserNames = tweets.stream()
                                .map(tweet -> tweet.getUser())
                                .filter(user -> user.getCountry().compareTo("AU") == 0)
                                .filter(user -> user.isVerified())
                                .map(user -> user.getName())
                                .collect(Collectors.toList());
```

2 System Design (30 marks)

Question 2. **(30 marks)**

After completing your degree you have chosen to join a game development company and you have been asked to design their next game *ShadowQuest*. *ShadowQuest* is a server-based, multi-player game; players will connect to a server via the Internet to interact with other game entities.

ShadowQuest has two main types of game entities: monsters, and players. All game entities are defined by their position in the world, and the mesh (or 3D model) that represents them. All game entities can also move.

Monsters can be passive or aggressive. All monsters have a maximum speed. Aggressive monsters can attack players when they get close to each other; players can attack other players and aggressive monsters. All units that can attack have a value for power, which starts at a maximum value and decrements on each attack; when power is down to zero, the unit is destroyed (not active), otherwise the unit is active. Passive monsters, who do not attack, start active with a Time to Live (TTL), and when this time lapses they die (not active).

All players keep a list of enemies (players and monsters) they have destroyed. Players also share a count of how many players are active in the game. Players can be either *human* or *computer controlled*. All human players have a username and computer controlled players have no further characteristics.

Finally, human players can be *controlled* by the person playing the game.

For the questions below, you must rely **only** on the specification provided; you may make design decisions about method arguments, but do **not** make assumptions about behaviours that haven't been specified.

a) Using **only** the description given above, design a class diagram for the *ShadowQuest* system and write Java class skeletons for your class design that show the attributes (including type) and methods signatures that are implied from the problem description. You must only submit the skeleton code for your classes not the class diagram.

Your classes must show privacy, class relationships, associations and have the correct data types to handle the multiplicities indicated in the class design. You do **not** need to show getters, setters, or constructors.
Note: You may assume that `Position`, `Mesh`, and `Input` are provided to you as libraries; you do not need to include them as separate classes in your UML diagram. (20 marks)

b) Choose a design pattern you have learnt this subject that will be most appropriate to use in the design of the game and describe how you would use it mentioning the name and role of classes - only a description is required not class diagrams. (6 marks)

c) Describe **two** test cases you might write to test your implementation, stating specifically what *behaviour/component* you are testing, what an *input* might be, and the expected *output/result.* Do **not** write any Java code for this question. (4 marks)

Invalid example: *Test whether X object is created correctly/test whether Y variable is given a value.* These tests are not related to your design or implementation, they are testing your ability to write code that works.

Invalid example: *Test that a castle can't be given a new owner if it is already owned.*
This is not specified, or even vaguely suggested by the description provided, nor is it a logical assumption.

Valid example:
Description (1 mark): ensure a human player cannot be instantiated without a name.
Input (0.5 mark): construct a human player with an empty string as a name.
Output (0.5 mark): throw an exception.

You cannot use this example in your answer.

Continued overleaf ...

3 Java Development (50 marks)

Question 3. **(25 marks)**

For this question you will implement the classes required to develop a simulation, `InfectionSpread.java`, which simulates infection spread using a simple infection propagation model. The simulation consists of locations, and people who move between locations. You will implement the two main classes, `Person` and `Location` using the specifications given below.

a) Implement the `Person` class with the following: (8 marks)

i. Four attributes:
   - `id` - a unique identifier for the person, known at the time of object creation, and not modifiable afterwards.
   - `infectionRisk` - a value between 0 and 1, which indicates the infection risk level of the person, known at the time of object creation, not modifiable afterwards.
   - `infectionLevel` - an integer with a value between 0 and `MAX_INFECTION_LEVEL`; 0 indicates that the person is not infected and as soon as a person gets infected the value is set equal to `MAX_INFECTION_LEVEL`, which then decrements by 1 each time step.
   - `locationId` - identifies the current location of the user (the unique `id` for the location).

   If the data type is not explicitly specified, you are required to choose an appropriate type.

ii. Appropriate constructor(s), getter(s) and setter(s).

iii. A method, `isInfected`, that returns a boolean `true` if the person is infected, `false`, otherwise.

iv. A method with signature:

```
public boolean updateInfectionLevel(double locationRisk)
```

   for a person who is currently infected, the method should decrement the `infectionLevel`; for an uninfected person, if the product of `locationRisk` and `infectionRisk` is greater than 0.2, set the `infectionLevel` to `MAX_INFECTION_LEVEL`. In both cases, the method should return `true` if the person is infected, `false` otherwise.

v. A `toString` method that returns all four attribute values for a person.

b) Implement a `Location` class with the following: (17 marks)

i. Four attributes:
   - `id` - a unique identifier for the location, known at the time of object creation, and not modifiable afterwards.
   - `neighbourLocations` - a map of `Locations`, with location id as key; `neighbourLocations` are locations a person can move to.
   - `people` - a map of `Persons`, with id as the key, which contains all the people at the location,
   - `numInfected` - number of infected people at the location,

   Note: If the data type is not explicitly specified, you are required to choose an appropriate type.

ii. Appropriate constructor(s), getter(s) and setter(s).

iii. A method, `addPerson`, that adds a `Person` to the location.

iv. A method, `removePerson`, that removes a person from the location.

v. A `toString` method that returns the location id, the number of people and the number of infected people at the location.

vi. A method, `nextState`, which is called by the simulation for each location, every time step, which must:

- Compute the `locationRisk` as:
  $locationRisk = \tanh(numInfected \times 0.1)$
- For each person at the location, calculates the new `infectionLevel` using the `updateInfectionLevel` method in `Person` class.
- For each person at the location, updates the next location; the next location for the person could be the current location, or ones of the neighbouring locations, chosen randomly with equal probability of choosing one of these locations.

Hint: The methods `tanh` and `random` in the Java `Math` class will be useful when implementing the above logic.

vii. A method, `movePeople`, that moves each person to the new location chosen in the `nextState` method.

Skeleton code for the `InfectionSpread.java` is shown below to help you better understand how the above classes you develop are used in the simulation; the code below is only additional information, and you are not required to use or modify it. You should be able follow the above instructions and implement the methods, even if you do not read or understand the code below.

```java
import java.util.HashMap;

public class InfectionSpread {
    public static void main(String[] args) {
        final int NUM_TIME_STEPS = 100;
        HashMap<Integer, Location> locations =  new HashMap<Integer, Location>();

        // Code to populate the locations map with location
        // and initial people information shoud be included here

        // Simulate each time step
        for(int t = 0; t < NUM_TIME_STEPS; t++) {
           System.out.println("=====================");
           System.out.println(("T = " + t));
           System.out.println("=====================");
           // Compute the next state for all locations
           for (Integer locId: locations.keySet()) {
              locations.get(locId).nextState();
           }
           // Move people to the new location calculated
           // in the previous step
           for (Integer locId: locations.keySet()) {
              locations.get(locId).movePeople();
           }
           // Print information for the location
           for (Integer locId: locations.keySet()) {
              System.out.println(locations.get(locId));
           }
        }
    }
}
```

Continued overleaf . . .

Question 4. **(25 marks)**

In this question you will implement a small object oriented system using generics and use it.

a) Implement a class named, `TwoKeyHashMap`, which is a hash map able to store values based on two keys. (17 marks)

Your class definition should begin with:

```
public class TwoKeyHashMap<K1, K2, V>
```

and must have the following methods implemented.

`TwoKeyHashMap()`	Constructs an empty public TwoKeyHashMap().
`boolean containsKey (K1 k1,  K2 k2)`	Returns `true` if this map contains a mapping for the specified key pair.
`V get(K1 k1, K2 k2)`	Returns the value to which the specified key pair is mapped, or null if this map contains no mapping, for the key.
`Set<Pair<K1, K2>> keySet()`	Returns a `Set` view of the key pairs contained in this map.
`V put(K1 k1, K2 k2, V v)`	Associates the specified value with the specified key pair in this map.
`boolean remove(K1 k1, K2 k2)`	Removes the mapping for the specified key pair from this map if present.
`int size()`	Returns the number of key-pair-value mappings in this map.

The class must use a single `HashMap` to store the values, and the key to the `HashMap` must be a class `Pair`, which can store the key pair. You are required to write the code for the `Pair` class as well.
Your class definition for the `Pair` class must begin with:

```
public class Pair<K1, K2>
```

and must contain the necessary methods to support the methods in `TwoKeyHashMap` class given in the table above.

b) Write a Java program for the following specification: (8 marks)

Your program, named `BigramCounter.java`, must print the frequencies of *word bigrams* in lines, stored in a file named `myFile.dat`; word bigrams are defined as two consecutive words occurring within a line.

You can assume that the words are separated by a single space. Your program must convert all words to lower-case, and remove trailing single commas(,) and full stops(.) in them before generating the bigrams.

Skeleton code for the class `BigramCounter.java` provided below, reads a line at a time, stores it in an array, and calls the `countBigramFrequencies` method. You are required to implement the `countBigramFrequencies` method, which must count and print the bigrams as per above specification.

Example:
If the file, `myFile.dat`, contains the following lines:

```
Software design is complicated, but fun.
software design is important.
I love object-oriented software design.
```

Your program must have the following output:

```
key: [is complicated] value: 1
key: [but fun] value: 1
key: [design is] value: 2
key: [complicated but] value: 1
key: [object-oriented software] value: 1
key: [i love] value: 1
key: [software design] value: 3
key: [is important] value: 1
key: [love object-oriented] value: 1
```

Important Notes:

- The keys may be displayed in any order you like, and does not have to be the same order as the example above.
- You may use Java provided classes as well as classes mentioned in part (a) of this question.
- Even if you did not attempt part (a), you can answer this question assuming the classes mentioned in part (a) exist.
- Inefficient code (e.g unnecessary loops, unnecessary memory usage) will result in mark penalties.

```java
import  java.io.*;
import java.util.ArrayList;
import java.util.Set;

public class BigramCounter {
    public static void main(String[] args) {
        try {
            File file = new File("myFile.dat");
                        BufferedReader br = new BufferedReader(new FileReader(file));
            ArrayList<String> lines = new ArrayList<String>();
            String line = null;
            while ((line = br.readLine()) != null) {
               lines.add(line);
            }
            countBigramFrequencies(lines);
            br.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found");
        } catch(IOException e) {
            System.out.println("Exception occurred");
    }
}
```

— *End of Exam* —