The University of Melbourne
Department of Computing and Information Systems

Final Examination

Semester 2, 2016

SWEN20003 Object Oriented Software Development

Exam Duration: 2 hours

Total marks for this paper: 120

This paper has 8 pages.

Authorised materials:
Students may NOT bring any written material into the room.
Students may NOT bring calculators into the room.

Instructions to Invigilators:
Each student should initially receive a script book.
Students may NOT keep the exam paper after the examination.

Instructions to Students:

- Attempt *all* questions.
- The exam has five questions, which are in the Section titled "Questions". All questions must be attempted. The marks 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.
- The Section titled, "Appendix: Java Documentation", gives the documentation for the Java classes that you can use in your questions. You not required to use all the listed classes and methods, just select the ones as needed.

This paper will be reproduced and lodged with Baillieu Library.

Page intentionally left blank

Questions

Question 1 [24 Marks]

1. **(4 marks)** Write an immutable `Person` class in Java which provides the following.

   - Two attributes: `lastName` and `firstName` of type String.
   - Appropriate initialization, accessors/mutator methods.

2. **(6 marks)** Write a `Student` class that is a subclass of `Person` class (in the previous question) which provides the following.

   - Two attributes: A unique `studentID` of type String and `GPA` of type float; the `studentID` is initialized when a `Student` object gets created, and cannot be modified thereafter.
   - Appropriate initialization, accessors/mutator methods.

   Note: `GPA` attribute stores the student's Grade Point Average.

3. **(14 marks)** Write a `Course` class in Java, which provides the following.

   (a) Two attributes: `courseName` and `year` of type String.

   (b) Information regarding all students enrolled in the course (information regarding each student is stored in the `Student` class from the previous question).

   (c) Appropriate initialization, accessors/mutator methods.

   (d) Ability to add a new `Student`.

   (e) Ability to remove a student based on the `studentID`.

   (f) Ability to calculate and return the average student GPA for the course. Average GPA is the sum of GPAs for all students divided by the number of students in the class.

   (g) Ability to display the list of students using two different strategies.

   - Name-based Strategy must display the list of students ordered based on last name.
   - GPA-based Strategy must display the list of students ordered based on the GPA.

   You must use the Strategy pattern to implement the display capability. You must write Java code for the classes to implement the strategies as well. However, when implementing the two strategy classes you are not required to write the code for sorting the student list, instead the method should just display a message to indicate that it is the display method of the particular strategy.

You may use any of a Java classes that are provided in the Appendix.

Question 2 [20 Marks]

A game studio is preparing to release Shadow Quest Deluxe for physical distribution, and due to the amazing videos, graphics and music, have found that all of its files combined adds up to several gigabytes in size, and needs to be shipped on multiple DVDs. Each DVD can hold n megabytes of data. There is an integer array s which contains the size of each file in megabytes, sorted from largest to smallest – some files are hundreds of megabytes and others are just a few megabytes, but no file is larger than n megabytes. To save costs, the studio has decided to try to minimise the number of DVDs that the game will require by placing the files according to the following algorithm:

Select the largest file and copy it onto the first disc. For each file thereafter (in decreasing order of size), copy it onto the first used disc that it will fit on, and if none exists, start a new disc. Continue this process until all files have been copied onto a disc.

For example if n = 700 and s = 600, 550, 400, 300, 120, 50, the solution is 3 (the DVDs will be packed with [600 + 50, 550 + 120, 400 + 300]).

Write a Java method that calculates the number of DVDs that the studio will end up using:

```
public static int num_discs(int n, int[] s);
```

Note: This is not necessarily the most efficient method, but that is the method the studio has chosen so you must follow it.

You may use any of a Java classes that are provided in the Appendix.

Question 3 [24 Marks]

1. **(4 marks)** Write a generic Java class, named `Pair`, that can store two values of different types. Your class must have a single constructor, which has two inputs parameters, each input parameter specifying the initial value of one of the attributes. The class must also have getter and setter methods for each attribute.

2. **(20 marks)** Write a generic Java class, named `TwoDHashMap`, that can store values as key-value pairs similar to a regular `Hashmap`, but the key is composed of two attributes. In addition to appropriate constructors required to support the design of your class, `TwoDHashMap` class must have the following methods.

`public boolean containsKey(K1 k1, K2 k2)`	Tests if an object with key pair (k1, k2) exists. in the map; returns true if the key pair exists and false otherwise.
`public V get(K1 k1, K2 k2)`	Returns the value associated with key pair (k1, k2).
`public void put(K1 k1, K2 k2, V v)`	Maps the specified key pair (k1, k2) to value v.
`public ArrayList<Pair> getAllKeys()`	Returns all the keys in the `TwoDHashMap`.

You may use any of the classes provided in the Appendix and the `Pair` class from part 1 of the question even if you did not attempt part 1.

Question 4 [20 Marks]

You have been asked to design a software system, using the object-oriented design paradigm, for a private medical practice that wishes to computerise its patient records system.

A patient must register with the practice and the system needs to store their name, address and contact telephone number. Each patient is given a unique 10 digit patient number at the time of registration, which is also stored in the system. The system will keep a count of the number of patients the practice currently has.

The practice employs two types of staff: Receptionists and Doctors. The system needs to record their details; which for all staff includes a four digit employee number, name, address, gender and contact telephone number. For doctors the system must store their 7 digit registration number. All receptionists must go for a training course every year and the system must record the date of when they last attended the course and the name of the course provider.

Patients can book an appointment with a particular doctor; the system needs to store the date of the appointment and if the patient attended. After the appointment, the doctor updates the system with the cost of the treatment undertaken.

A list of appointment statistics is required at the end of each week. This will be a summary of how many patients turned up and how many were no-shows. If a patient repeatedly misses two appointments he/she will be charged a fixed cost.

Using the description given above, draw a UML class diagram for the medical practice. In your class diagram show the attributes (including type) that are implied from the problem description. Methods are not required. You must show class relationships, association directions and multiplicities.

Question 5 [32 Marks]

1. **(6 marks)** Give the meaning of the following features in relation to the object-oriented design paradigm.

   - Abstraction
   - Inheritance
   - Delegation

2. **(10 marks)** *"A good design is one that can be extended, adapted and re-used."*

   Discuss how the object-oriented features abstraction, inheritance and delegation support good design as per above statement.

3. **(16 marks)** You have been asked to design an object-oriented game development framework to be used by developers developing role play games (RPGs) such as the *Shadow Quest* game developed in project 2. Discuss how you would use abstraction, inheritance and delegation to design this framework showing examples of UML class diagrams to support your design.

Appendix: Java Documentation

HashMap

```
public class HashMap<K,V>
    extends AbstractMap<K,V>
    implements Map<K,V>, Cloneable, Serializable
```

The `HashMap` class, in `java.util` package, implements the Map interface, which maps keys to values. Any non-null object can be used as a key or as a value.

`HashMap()`	Constructs a new, empty HashMap with a default initial capacity (11) and load factor (0.75).
`HashMap(int initialCapacity)`	Constructs a new, empty HashMap with the specified initial capacity and default load factor (0.75).
`boolean contains(Object value)`	Tests if some key maps into the specified value in this hashmap.
`boolean containsKey(Object key)`	Tests if the specified object is a key in this hashmap.
`boolean containsValue (Object value)`	Returns true if this hashmap maps one or more keys to this value.
`V get(Object key)`	Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
`Set<K> keySet()`	Returns a Set view of the keys contained in this map.
`Set<Map.Entry<K, V>> entrySet()`	Returns a Set view of the mappings in the map.
`V put(K key, V value)`	Maps the specified key to the specified value in this hashmap.
`V remove(Object key)`	Removes the mapping for the specified key from this map if present.
`int size()`	Returns the number of elements in this list.

ArrayList

```
public class ArrayList<E>
   extends AbstractList<E>
   implements List<E>, RandomAccess, Cloneable, Serializable
```

The `ArrayList` class, in `java.util` package, is a resizable-array implementation of the List interface.

`ArrayList()`	Constructs an empty list with an initial capacity of ten.
`boolean  add(E e)`	Appends the specified element to the end of this list.
`void  add(int index, E element)`	Inserts the specified element at the specified position in this list.
`boolean  equals(E element)`	Compares the specified object with this list for equality.
`Element  get(int index)`	Returns the element at the specified position in this list.
`E remove(int index)`	Removes the element at the specified position in this list.
`E set(int index, E element)`	Replaces the element at the specified position in this list with the specified element.
`boolean remove(Object o4)`	Removes the first occurrence of the specified element from this list, if it is present.
`Iterator<E> iterator()`	Returns an iterator over the elements in this list in proper sequence.
`int lastIndexOf(E e)`	Returns the index in this list of the last occurence of the specified element, or -1 if the list does not contain this element.
`ListIterator listIterator()`	Returns an iterator of the elements in this list (in proper sequence).
`ListIterator listIterator(int index)`	Returns a list iterator of the elements in this list (in proper sequence), starting at the specified position in the list.
`int size()`	Returns the number of elements in this list.

Iterator/ListIterator

```
public interface ListIterator<E>
    extends Iterator<E>
```

`boolean hasNext()`	Returns true if this list iterator has more elements when traversing the list in the forward direction.
`boolean hasPrevious()`	Returns true if this list iterator has more elements when traversing the list in the reverse direction.
`E next()`	Returns the next element in the list.
`void remove()`	Removes from the list the last element that was returned by next or previous (optional operation).