CPSC 210
Sample Final Exam Questions

Note: the questions in this document do not constitute an actual final exam. They are provided to you to give you some indication of the **kinds** of questions that could be asked. However, there are other kinds of questions that could be asked and other topics that could be covered. You should therefore not use these sample questions as your primary source of study material.

Q1. Draw an intra-method control flow diagram for the *iterative version* of the method:

```
MyListNode<E> find(int index) throws IllegalIndexPosition;
```

in the `ubc.cpsc210.list.linkedList.MyLinkedList` class of the project `\lectures\LinkedListComplete`.

Q2a) What does it mean for the design of a class to be robust?

b) Design and implement a class that represents an *unchecked* exception that will be thrown by the following method of a `MyArrayList<E>` class when the index is not valid:

```
public E get(int index);
```

The class must provide a constructor that takes the invalid index as a parameter and uses it to construct a meaningful error message that can be displayed when the exception is caught.

Q3. Suppose that a friend of yours has designed a type hierarchy where `ClassB` is a subclass of `ClassA`. `ClassB` overrides the method `doSomething` defined in `ClassA` and throws a checked exception that is not thrown by the method in `ClassA`. The code does not compile. In terms of a design principle studied this term, explain why you would not want such code to compile.

Q4. This question refers to the class `ubc.cpsc210.list.linkedList.MyLinkedList` from the `\lectures\LinkedListComplete` project.

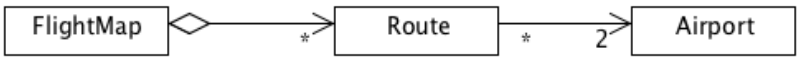
Complete the implementation of the following member of the `MyLinkedList` class. Assume that `Collection<E>` is from the `java.util` package.

```
/**
 * Returns true if this list contains all the elements in the
 * collection c, false otherwise.
 */
public boolean containsAll(Collection<E> c) {
```

Q5. Provide an implementation for the classes shown in the UML diagram below. You must include any fields or methods necessary to support the relationship between the classes in addition to appropriate getters and setters. Note that a route has two associated airports: the departure airport and the arrival airport. Each airport has a unique code (e.g. "YVR" represents Vancouver International, "LHR" represents London Heathrow and "PEK" represents Beijing) which cannot be changed.

Assume that once set, the arrival and departure airports for a particular route cannot be changed. Further assume that routes can be added to or removed from a flight map but the same route cannot be added to the flight map more than once. We consider two routes to be the same if they have the same departure and arrival airports. Two airports are the same if they have the same code.

Note: You can use Eclipse to generate any hashCode() and equals() methods you might need.



Q6. For each of the scenarios below, identify which collection from the Java Collections Framework you would use and briefly justify your answer.

a) Suppose you want to simulate line-ups at a bank. There can be anywhere from one to several tellers available at any given time and each teller has their own line-up of customers. Tellers are numbered sequentially starting at position 0. You want to be able to get the line-up for a particular teller by specifying the teller's position number. Assume that there is a `Customer` class in the system. How would you represent the collection of line-ups?

b) Suppose you are designing a course registration system. Assume that there is a `Student` and a `Course` class in the system. How would you store the students and courses so that you can quickly retrieve the courses in which a given student is registered?

Q7. You have been asked to alter the `lectures/PhotoAlbumBase` system to make it possible for a method containing the following code to compile and execute correctly:

```
Album anAlbum = new Album("My Album");
// Put lots of photos into album
//…
for (Photo p: anAlbum) {
   // do something with each photo
}
```

For each interface, class, field or method that must be changed or added, describe the change or addition in as close to correct Java syntax as you can.

Q8. You have been given a class `ConsoleWriter` that can write a given string to the console.

```
public class ConsoleWriter {

   private writeToConsole(String s) {
      System.out.println(s); }
}
```

You have been asked to alter the functionality of the `lectures/PhotoAlbumBase` system to write to the console the name of any photo added or deleted from an album in the system. You have been told you must use the Observer design pattern to implement this new functionality.

a) Draw a UML class diagram that provides an overview of the changes and additions needed to `PhotoAlbumBase` to support the use of the Observer design pattern to provide the desired functionality. You need not reproduce the entire UML class diagram for the system. Just show those parts of the UML class diagram that must change. Indicate fields and methods that must be changed or added in the UML class diagram.

b) For each interface, class, field or method that must be changed or added, describe the change or addition in as close to correct Java syntax as you can.

Q9. Consider the following Java class.

```
public class Clock {
   private Integer startTime;
   private Integer numHours;

   public Clock( Integer start, Integer hrs ) {
      startTime = start;
      numHours = hrs;
   }
}
```

Modify this class and introduce any other code required, but without using any standard Java Collection Framework classes or interfaces (e.g., `List`), so that you can iterate over an instance of `Clock` using a for-each loop. For example, the following code should print the numbers 2, 3, 4, and 5. You do not have to consider the case where the hours go past 23. Assume the input you're given will produce output within the same day (i.e., values between 0-23 only).

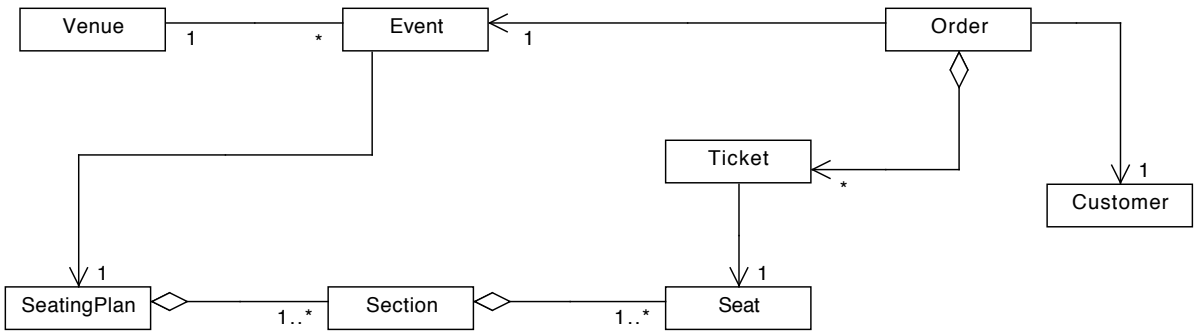
```
for (Integer i: new Clock(2,4)) {
   System.out.println(i);
}
```

Q10. Suppose you are designing an Android app that will allow the user to perform task management. The user can add tasks to the system and can group tasks together into projects. Projects can be added as sub-projects of other projects, nested arbitrarily deep. Each task has an estimated time for completion that is specified when the task is constructed. You want to be able to treat individual tasks and projects in the same way. In particular, you want to be able to get the time needed to complete a task or a project. The time taken to complete a project is the total time needed to complete all the tasks in the project or in sub-projects of that project.

a) Consider the partially completed code in the `lectures/TaskManager` project. Draw a UML diagram that includes all the classes in the `ca.ubc.cpsc210.taskmanager.model` package and that shows how you will apply the composite pattern to the design of this system.

b) Write the complete implementation of the `Task` and `Project` classes. Note that all the tests provided in `ca.ubc.cpsc210.taskmanager.tests.TestProject` should pass. Space is also provided on the following page for your answer to this question.

Q11. The following UML class diagram represents the design of a "TicketWizard" system that allows tickets to be sold for different events held at venues across the country.



Answer the following questions based on the design presented in the UML diagram above and **not** on what you think the design should be! Circle **True** or **False** and briefly explain your choice. Note that you will earn no marks if your explanation is not correct.

a) True or False: given a customer object, you can retrieve the orders placed by that customer.

b) True or False: it is possible that an order has no tickets associated with it.

c) True or False: given a ticket object, it is possible to retrieve the section object to which the associated seat belongs.

Q12. This question refers to the `lectures/CompositeDrawingEditorComplete` project. Note that a `Drawing` stores the figures drawn by the user in a list in the order in which they were drawn. In this question you must assume that the user has drawn three figures in the order listed here: a rectangle, an ellipse and a square. Now assume that the user chooses the "select" tool and clicks the ellipse figure. In so doing, the ellipse figure changes from "not selected" to "selected".

Draw a *sequence diagram* for the method `SelectionTool.mousePressedInDrawingArea`, called as a result of the user clicking the ellipse figure with the "select" tool. It is not necessary to include calls to any methods in the Java library. Use the fact that you know which figures have been drawn to resolve calls to abstract methods. If you encounter a branch in the code, it is necessary to include only those methods that will actually execute in the scenario described above. You may abbreviate the names of types and methods provided you do not introduce any ambiguity into your diagram.