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 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`. Note, we didn't cover the implementation of linked lists this term but we think you should probably be able to follow what the code is doing.

Q2. Draw a call graph starting at `DrawingMouseListener.mousePressed` in the `Drawing` class of the `\lectures\Decorator-Lecture-Lab-Complete` project. Assume that the active tool is a `RectangleTool` and use this information to resolve any calls to abstract methods in the `Tool` class. Include calls to methods in the `ca.ubc.cpsc210.drawingEditor.*` packages only, including methods inherited from super-types.

Q3a) 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 the `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.

Q4. Suppose that a friend of yours has designed a type hierarchy. At one point in the hierarchy a subtype overrides a method and throws a checked exception that is not thrown by the overridden method in the super-type. The code does not compile. In terms of a design principle studied this term, explain why you would not expect such code to compile.

Q5. This question refers to the class `ubc.cpsc210.list.linkedList.MyLinkedList` from the `\lectures\LinkedListComplete` project.

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

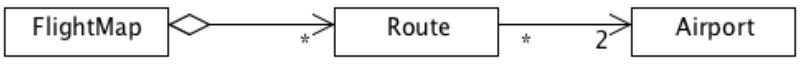
```
/**
 * Returns true if this list contains all the elements in the
 * collection c, false otherwise.
 */
public boolean containsAll(Collection<E> c) {
```

b) (Hard Question: This is harder than what you will find on the final but doing it will improve your knowledge of how object-oriented code works. You will have to first really understand how the code in `MyLinkedList` works.) Provide an iterative and recursive implementation of the following member of the `MyLinkedList` class:

```
/**
 *  Returns the index of the given element or -1 in the case when the
 *  element is not in the list.
 */
public int indexOf(E element) {
```

Q6. 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.



Q7. 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?

Q8. You have been asked to alter the `lectures/PhotoAlbum` system (may be called PhotoAlbumBase in repository) 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
}
```

a) Draw the portion of the UML class diagram that provides an overview of the changes and additions needed to `PhotoAlbum` to support the code above. You need not reproduce the entire UML class diagram. 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. 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/`PhotoAlbum` 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 `PhotoAlbum` 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.

Q10. You have been asked to add new functionality to the `lectures/Pacman-refactored` code. Each time the board is redrawn (in `Board.tickBoard()`), you must write to the Java console, the number of monsters active in the system. You have been asked to design this feature by using the Observer design pattern.

Extract a design (UML Class diagram) for Pacman-refactored and alter it to show how you would support this functionality using the Observer design pattern. Include any classes or interfaces you would need to add to the design. Include any new or changed methods on new or existing classes and interfaces in the design that are needed to support the desired functionality.

Write a few sentence description of how the system would work to support this new functionality.