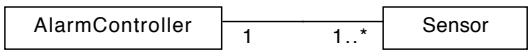
CPSC 210
Sample Final Exam Questions

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. You should also review the sample questions from the midterm exams, given that the final will be comprehensive.

Q1. Extract a UML class design diagram for all types in the `ca.ubc.cpsc210.photo` package in the `PhotoAlbumBase` system found on GitHub. Note that this system is similar to, but not exactly the same as, the photo editing application that you worked with in lab. Do not include any methods or attributes in your diagram. Do not include any types from the Java libraries.

You may add any notes you wish to describe your choice of relationship(s) in the diagram. You are not required to add any notes if you think the choice of relationship(s) between any two types is straightforward.

Q2. For each of the UML class diagrams shown below, indicate the name used to describe the relationship (e.g., an inheritance relationship) between the classes (or instances of those classes) and describe *in point form* what the diagram communicates about the relationship.



i. Name used to describe this relationship:

Point form description of what diagram communicates about relationship:

ii. Name used to describe this relationship:

Point form description of what diagram communicates about relationship:

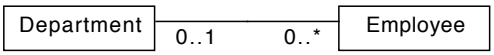
Q3 a) Choose a suitable concrete (so do not use interfaces or abstract classes) data structure from the Java Collections Framework (JCF) for each of the following problems:

i. You are writing a system to manage a hockey pool. For each participant in the pool, you must be able to track a team of players. What data structure will you use to represent the team of players? Why?

ii. You are writing a system to model line-ups at the bank. Each teller has their own line-up. What data structure will you use to store all the people in line at all of the tellers?

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?

Q4. This question relates to the `HRSystem` found on GitHub. The following UML class diagram represents the relationship between objects of type `Employee` and `Department`.



a) Draw an object diagram that contains one department and at least three employee objects. Two of the employees must be associated with the department and one employee must not be associated with any department.

b) Complete the implementation of the `Employee` and `Department` classes. When you are done, all the provided tests must pass. Note that an employee's ID can never be changed once set but an employee can change their name or department. Two employee objects with the same ID but different names or departments are considered to be the same employee. Think very carefully about what happens if an employee changes department – you might want to add a department to your object diagram and think about what changes must be made to maintain the relationship shown in the UML class diagram above.

Q5. 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 found in the repository `LinkedListComplete` on GitHub.

Q6a) 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.

Q7. This question refers to the class `ubc.cpsc210.list.linkedList.MyLinkedList` from the `LinkedListComplete` project.

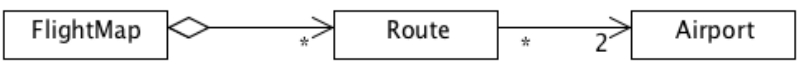
Complete the implementation of the following member of the `MyLinkedList` class. Assume that `Collection<E>` is from the `java.util` package.

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

Q8. 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 IntelliJ to generate any hashCode() and equals() methods you might need.



Q9. You have been asked to alter the `PhotoAlbumBase` project 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.

Q10. You have been given a class `ConsoleWriter` that can write a given string to the console.

```
public class ConsoleWriter {

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

You have been asked to alter the functionality of the `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.

Q11. 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).

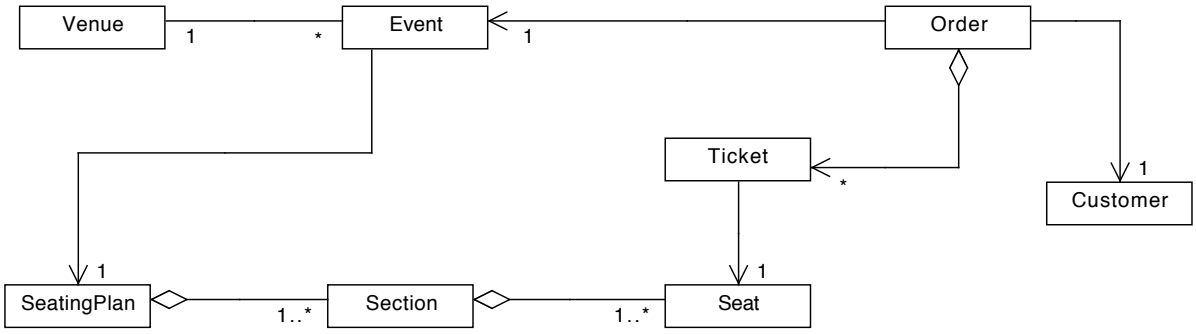
```
for (Integer i: new Clock(2,4)) {
   System.out.println(i);
}
```

Q12. 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 `TaskManager` repository on GitHub. 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 if your implementation is correct, 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.

Q13. 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.

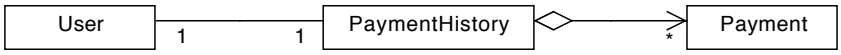
Q14. This question refers to the `CompositeDrawingEditorComplete` project. Note that a `Drawing` stores the figures drawn by the user in a list in the order in which they were drawn. In this question you must assume that the user has drawn three figures in the order listed here: a rectangle, an ellipse 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.

Q15. This question uses the `EMailManager` project found on GitHub.

Draw a sequence diagram for the method `AddressBook.addGroup` within the package `ca.ubc.cs.cpsc210.addressbook`. If you have to loop over a collection, you must assume that there are exactly two objects in that collection. Include calls to *all* methods in the `EMailManager` project *except* constructors. You must also include the first level of calls to the Java library, if any. Be sure to include a legend if you abbreviate class names.

Q16. Consider the UML class diagram shown below:



It represents the design for a small part of an online storage system. Users have to pay for the service and a history of payments is maintained in the system. Write the code for the `PaymentHistory` class. You must include fields and methods that are necessary to support relationships between the other classes shown on the UML diagram but it is not necessary to include any others. Assume that you can add a Payment to the `PaymentHistory` but a `Payment` cannot be removed. Further assume that the payments must be stored in the order in which they were added and that the `PaymentHistory` must not contain duplicate `Payment` objects. Assume that the constructor of the `User` class creates the corresponding `PaymentHistory` object.

Q17. Draw a UML class diagram to represent the design of all the types in the `addressBook` and `email` packages of the `EMailManager` system checked out from the lectures repository.

Q18. This question uses the `SnakeStarterLecLab` project found on GitHub.

a) Draw a UML class diagram to represent the design of all the types in the `model` package. It is recommended that you check out a new copy of this code and assume that the `Food` class has been implemented according to the given specification.

b) Draw a UML sequence diagram to model the call to `Snake.move`. Include the first level of calls to the Java library. Model only the code found in the *first* case in any `switch` statement that you encounter.