CS2102, D15

Exam 1

Name:

You have 50 minutes to complete the problems on the following pages. There should be sufficient space provided for your answers.

If a problem asks you to create a class hierarchy, we are looking for the interfaces, classes, and abstract classes that you would create for the problem. In particular:

- Include **implements** and **extends** statements
- Include field names and types
- Include method headers (names, return type, and input parameter types)
- Full credit requires that all types and implements/extends relationships are clear. Be sure your work is clear if you use class diagrams instead of Java syntax.
- You may omit constructors
- You may omit method bodies
- You may omit the `Examples` class (examples of data and test cases) unless a question asks otherwise

---

3. **Topic: Encapsulation**

   (This starts from the same code as question 2–start with the code afresh for this question, ignoring any edits you made for question 2). You are writing a course scheduling tool that helps students search for classes with open seats. Your current code is as follows:

```
// captures hour when a course starts, how many hours it lasts, and a term.
// A one hour class starting at 3pm in A-term, is   new TimeSlot(3, 1, "A")
class TimeSlot {
  int startHour;
  int duration;
  String term;
}

// a Course in the registration system
class Course {
  String dept;
  int number;
  TimeSlot when;
  int seatsLeft;
}

class Scheduler {
  private LinkedList<Course> catalog;
  private LinkedList<TimeSlot> allTimes;

  // return list of courses that start after given hour in given term
  // and that have seats available
  LinkedList<Course> availableAfter(int hour, String inTerm) {
    LinkedList<Course> result = new LinkedList<Course>();
    for (Course c : catalog) {
      if (c.when.startHour >= hour &&
          c.when.term.equals(inTerm) &&
          c.seatsLeft() > 0) {
        result.addFirst(c);
      }
    }
    return result;
  }
}
```

   (a) (10 points) On the code itself, box off the code fragments that need to change in order to let you encapsulate (change) the data structure used for the catalog in the `Scheduler` class. You only need to draw boxes to answer this part.

(exam continues next page)

(b) (20 points) Provide classes and/or interfaces you would add in order to encapsulate the `catalog` data. Provide complete interfaces. For classes, provide variables and headers (names and input/output types) for methods that are not required by their interface. Include only variables and method headers needed to encapsulate the code on the previous page (don't add new features). You do not need to copy chunks of code from the previous page: a good method name will indicate the corresponding code.

```
interface ICatalog {
  LinkedList<Course> findByTime (int hour,
                                  String inTerm);
}

class CatalogList implements ICatalog {
  LinkedList <Course> catalog;

}
```

(c) (10 points) Should any specific code that is currently in the `availableAfter` method move to the `Course` or `TimeSlot` classes? Either identify code that should move (and tell us where to), or justify why no changes are necessary. You do not need to write or edit new code.

the comparisons on c.when should become a method in the TimeSlot class.

(end of exam)