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

---

Grading Summary

Exam starts on the next page

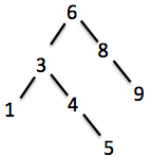
Topic	Max Points	Score
Q1: Data structures	36	
Q2: Java programming	40	
Q3: Program Design	40	

(Each of these is a separate score in the corresponding course theme)

(exam starts on the next page)

1. **Topic: Data Structures**

   (a) (18 points) Consider the following two binary trees. For each data structure listed in the table, either say "yes" if the tree satisfies the requirements of the data structure, or write the number that starts the smallest subtree that violates the requirements for that data structure.

		
BST		
AVL Tree		
Heap		

   (b) (18 points) You are building software to manage orders at a fast-food restaurants. Orders should get filled in the order that they were placed. You need to select a data structure for organizing the orders.

   In the first row of the following table, put a mark (x, check, etc) under each data structure that is appropriate (structurally) for managing orders. In the second row, mark the ONE data structure that you prefer for this problem. Justify your preference (and non-preferences from the appropriate structures) in a few words.

	Stack	Queue	BST	AVL tree	Heap
Appropriate for Problem					
Prefer for Problem					

   Justification:

(exam continues next page)

2. **Topic: Class Hierarchies (Classes, Abstract Classes, Interfaces) and Access Modifiers**

(This starts from the same code as question 3–start with the code afresh for this question, ignoring any edits you made for question 3). 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;
  }
}
```

Your current scheduler does not handle Labs. Each lab has the course it is part of, the TimeSlot in which it is offered, and the number of seats remaining in the course.

(a) (30 points) Edit the class hierarchy and code to allow the catalog to contain both courses and labs. Your class hierarchy should share fields and methods where appropriate. You may add (abstract) classes and interfaces (define interfaces in full, define classes at the level of detail of `TimeSlot` and `Course`). You do not need to add any features, methods, or encapsulation—just make the edit requested in this problem. Mark edits on the original code, using the space around the code and on the next page for any new classes/interfaces.

(exam continues next page)

(space for answer to part (a))

(b) (10 points) Assume you are writing an `Examples` class for the Scheduler, along the following lines:

```
class Examples {
  LinkedList<Course> catalog2015 = new LinkedList<Course>();
  catalog2015.addFirst(new Course(...));

  Scheduler S = new Scheduler(catalog2015);
  int spots = catalog2015.getFirst().seatsLeft;
}
```

i. Will the line that defines `S` compile without error, given the type of `catalog2015` and the edits you made in part (a)? Why or why not?

ii. Will the line that defines `spots` compile without error, given that `catalog` is private in the `Scheduler` class? Why or why not?

(exam continues next page)

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.

(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.

(end of exam)