CS2102, B12

Exam 2

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 an interface, you should provide a complete interface, including method headers and argument types.

If a problem asks you to create a class:

- Include **implements** and **extends** statements
- Include field names and types
- Omit constructors
- Omit methods unless a problem asks otherwise

Omit the `Examples` class (examples of data and test cases) unless a question asks otherwise.

If a problem asks you to choose a data structure, provide the type of content as well as the name of the data structure (for example, a LinkedList<String>, a HashMap<String,Dillo>, a graph with People as nodes). You may indicate the content types either in code or in prose.

Grading Summary

Exam starts on the next page

Topic	Max Points	Score
Q1: Create data	15	
Q1: Encapsulate data	5	
Q2: Choose appropriate data structure	30	
Q3: Test when multiple correct answers	20	
Q4: Create a class hierarchy	10	
Q4: Abstract over data/types/functions	20	

(exam starts on the next page)

1. **Topic: Creating and Encapsulating Data**

   Recall the banking system we worked with earlier in the term: each `Account` contained a `Customer` and each `Customer` contained a list of his `Accounts`. The classes appear below, this time with constructors.

```
class Customer {
  String name;
  int password;
  LinkedList<Account> accounts;

  Customer(String name, LinkedList<Account> accounts) {
    this.name = name;
    this.accounts = accounts;
    this.password = genPassword();
  }
}

class Account {
  int number;
  Customer owner;
  double balance;

  Account(int number, Customer owner) {
    this.number = number;
    this.owner = owner;
    this.balance = 0;
  }
}
```

   Someone named Alice wants to register as a new customer and simultaneously open a new account. Write code to do this using calls to the constructors provided. If you need code beyond calls to the constructors, (a) explain why, and (b) show the code. For encapsulation points, write the extra code using appropriate new methods and state the purposes of those methods (you do not need to write the methods themselves, just purposes).

(exam continues next page)

2. **Topic: Data Structures**

An online bookstore needs to manage sales information about books. They have a `Book` class storing the title, author, and a unique identification number for each book. For each of the following two problems, we list three data structures. For each data structure, briefly state why you would or would not use it for the given problem. Indicate types of contents on any data structures that you would use.

(a) The sales department sometimes offers deals on books, such as a discount if a customer buys another specific book at the same time. At any one time, there is <u>at most one deal for each book</u>. Consider each of these data structures for storing the current deals.

List

Hashmap

new variable in the Book class

(question continues next page)

(b) Some deals are based on which books are frequently bought together. For example, there might be a deal on `book1` and `book2` that are not often purchased together, but that are each often purchased together with `book3`. Consider each of these data structures for storing information on which books are frequently bought together.

Graph

Hashmap

new variable in the Book class

(exam continues next page)

3. **Topic: Testing Methods with Multiple Correct Answers**

With users feeling overwhelmed by too much information, a social network decides to introduce a new "snapshot" feature. The snapshot shows each user only a random sample of her friends' recent posts. A snapshot for a given user on a given date must meet the following criteria:

- It contains at most 10 posts, each one posted by one of the user's friends
- Every post was made within the last 3 days
- Posts were posted by at least 5 different people (you may assume everyone in the network has at least 5 friends)

The following code shows the essential fields and methods in the relevant classes; the `getSnapshot` method in the `SocialNetwork` class generates snapshots.

```
class Member {
  LinkedList<Member> friends;
}

class Post {
  String contents;
  Date postedWhen;
  Member postedBy;
}

class SocialNetwork {
  LinkedList<Post> getSnapshot(Member forMember, Date onDate) {
    ...
  }
}
```

Outline how you would test `getSnapshot`. A good answer will (a) describe any additional methods you need to write and (b) give one concrete example of a `checkExpect` that illustrates your testing approach. For each method, indicate which class it goes in and provide only the input/output types and a purpose statement (**DON'T write the full method**). For the `checkExpect`, show Java expressions for the actual and expected answers, but ignore all the other surrounding syntax. If you need more space, use the next page.

(exam continues next page)

(Additional space to answer Question 3)

(exam continues next page)

4. **Topic: Class Hierarchies; Abstracting over Data, Types, and Functions**

In class, we discussed that programs separate their user-interface code from their core data structures, in part to enable different kinds of interfaces. Imagine that a programmer is building a system with two kinds of user interfaces: one that prints options to the screen and one that reads options aloud. In her initial design, the programmer has created the following four classes (text on left, audio on right):

```
class TextOption {
  String message;

  void presentOption(){
    System.out.println(message);
  }
}
-----------------------------------
class TextConsole {
  LinkedList<TextOption> options;

  void presentAllOptions() {
    for (TextOption t : options) {
      t.presentOption();
    }
  }
}
```

```
class AudioOption {
  AudioFile theAudioFile;

  void presentOption(){
    theAudioFile.play();
  }
}
-----------------------------------
class AudioConsole {
  LinkedList<AudioOption> options;

  void presentAllOptions() {
    for (AudioOption o : options) {
      o.presentOption();
    }
  }
}
```

Seeing a lot of similarities in the code, you decide to clean it up.

(a) Show the class hierarchy you envision for your cleaned up code. Include `extends` and `implements` statements, and method headers (in interfaces and classes), but no method bodies.

(question continues next page)

(b) For each of the following Java abstraction mechanisms, explain briefly why you did or did not use it in part (a). Answer this based only on the given code, not considering any other methods that might be in such a system.

- Add a new Abstract class

- Add a new Interface

- Add generics

- Create a method that takes another method as an argument

(end of exam)