CS2102, D15

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 needs you to give an example of a data structure (like a list, graph, or hashmap), you do **NOT** need to write these in full code (even in a test case). You may use reasonable abbreviations such as

- a picture for a graph
- `List[2,3]` for a list of numbers
- `HM[``Kathi'' -> 5118, ``Chris'' -> 5776]` for a hashmap from names to extensions

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: Testing	30	
Q2: Testing	30	
Q3: Java Programming	30	
Q4: Data Structures (optional)	24	

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

(exam starts on the next page)

1. **(30 points) Topic: Designing Good Test Cases**

   Assume you are writing a `Graph` class to capture routes between cities (as we did in lecture). You wrote a method that finds the shortest route that (a) goes between two cities while (b) passing through a third city:

   ```
   LinkedList<Node> shortestRouteThru(Node from, Node to, Node through)
   ```

   Describe three (3) interesting test cases for this method, each for a situation where a route DOES exist between the `from` and `to` cities. Draw picture(s) of the graph(s) you would create (showing names of nodes and edges between them), write the calls to `shortestRouteThru` (including inputs), and briefly describe the goal of the test. If you draw multiple graphs (having only one is fine), indicate which to use for each test case.

   Here's an example of the expected format (though for a case where the route did not exist; picture not included)

   ```
   // check case where no route exists
   shortestRouteThru(Boston, Worcester, Chicago)
   ```

   Do NOT write full `checkExpects`. Do NOT write the actual method.

(exam continues next page)

2. **(30 points) Topic: Testing Methods with Multiple Correct Answers**

A new internet startup plans to show ads based on users' interests. Each ad stores the client (i.e., company) who has paid to post the ad, the number of times that the client wants the ad shown, the number of times the ad has already been shown, the contents, and a list of interests that the ad targets (e.g., "music", "sports"):

```
class Ad {
  Client sponsor;
  int timesToShow;
  int numTimesShown;
  String contents;
  LinkedList<String> forInterests;
}
```

They have too many ads to show all at once, so they wrote a `chooseAds` method that selects and returns a given number of ads to display. Each returned ad should (a) target at least one interest from the given list of interests, and (b) have been shown fewer times than the client paid for.

```
LinkedList<Ad> chooseAds(int howMany, LinkedList<String> forInterests)
```

Outline how you would test `chooseAds`. A good answer will (a) illustrate your approach by showing one sample test case including `checkExpect` and (b) describe any additional methods you would need (including input/output headers, a brief text description, and which class the method would go in). The sample test case can use shorthand for data and show just the essential code around the `checkExpect` (i.e., omit/assume the `tester` argument, `test` method header, etc).

You do not need to write a full set of test cases, bodies of your additional methods, or the chooseAd method. If you need more space, use the next page.

(exam continues next page)

(Additional space to answer Question 2)

(exam continues next page)

3. **(30 points) Topic: Creating Cyclic Data**

The same internet startup from question 2 (which places ads on behalf of clients) has a class for each of ads and clients. Objects of each class reference objects from the other: the client contains a list of ads, and each ad references its sponsoring client (the `Ad` class is the same as in question 2).

```
class Ad {
  Client sponsor;
  int timesToShow;
  int numTimesShown;
  String contents;
  LinkedList<String> forInterests;
}

class Client {
  String name;
  LinkedList<Ad> ads;
}
```

(a) (10 points) Consider the problem of setting up the mutual references between clients and their ads (when creating objects from these classes). On the code above, draw boxes around the fields whose values should be provided as arguments to the constructor for each class (unboxed fields would be given default values within the constructor). You only need to draw boxes for this part.

(b) (20 points) Assume you had constructors that match your boxes (from part (a)). Write code that (1) creates WPI as a new client who (2) pays for 100 showings of a new ad with the content "great school" and targets those interested in "goats". If you need methods beyond the constructors (a) use them as if they existed, and (b) indicate which class they would go in.

(required exam ends here – optional question on next page)

4. **(24 points) Topic: Data Structures (Optional Question to makeup for Quiz 2)**

   Our same internet company (from the previous two questions) needs to store its ads in such a way that it can quickly locate which ones to display based on both users' interests and the number of times an ad still needs to be shown. For each of the following data structures, briefly describe either how you might use it for this problem, or explain why it doesn't make sense. Good answers illustrate that you know the key features of the data structure you are discussing.

   (a) Lists (if you would use, describe the type in the list)

   (b) HashMap (if you would use, describe the types of keys and values)

   (c) Heap (if you would use, describe the ordering criterion)

(end of exam)