This document provides some sample questions that might appear on a CPSC 310 midterm. It is not representative of the length of a 50 minute midterm (i.e., it would be way too long).

This document is not representative of all the topics that might be covered on the midterm. See the lecture guide for details on what the midterm will cover.

As part of the midterm you can assume you will be given:

- A UML reference guide showing notation for extends (inheritance), implements, association and aggregation.
- The structural parts of any design pattern referenced in the exam.

True/False (if false explain why)

1. Agile methods are different from the Spiral model because development is organized as iterations instead of one single implementation phase.

3. User stories should always be accompanied by automated tests (such as JUnit), so that testing can be performed efficiently.

4. The Daily Scrum meeting allows developers to show a demo of their work, so the product owner can determine if progress is on track.

5. Managers should attend Daily Scrum meetings so that they can give feedback at the meeting about whether a product is achieving a company's business objectives.

6. In Extreme Programming, actual customers should be involved in the creation of acceptance criteria for user stories.

7. User stories describe implementation tasks that developers must complete.

8. A sequence diagram models the static relationships between objects.

9. It is important for software development teams to estimate task durations so that they can work on the shortest duration tasks first.

Short Answer

8. Describe why a traditional waterfall process could be preferred over a spiral model, in cases where the project requirements and implementation were well understood and highly predictable?

9. In the area of Agile methods, what does the acronym INVEST stand for?

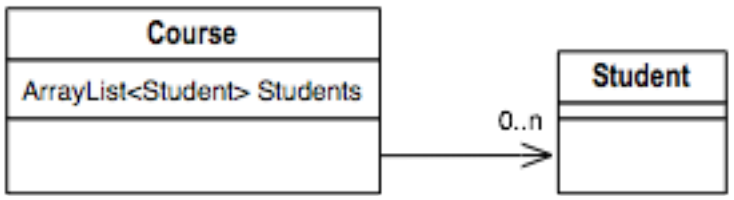
10. What is one of the major problems caused when user stories are too big?

11. What is the purpose of the Scrum Master in the Scrum methodology?

12. What is one problem that is addressed by Collective Ownership of code in Extreme Programming?

13. What is the glaring error in this diagram:

14. Imagine a system (Holiday Shopper) that covers the user story: “As someone planning my



holiday shopping, I would like to be able to have a list of people to shop for and be able to add gift ideas under their names”. Write a user instruction to test a paper prototype of your application.

15. Draw a UML class diagram for the following user story: As a user I want a tool that lets me take a photo of an item I found in a location so that the photo and the location of my GPS is uploaded as a “found item” to the central lost and found service which adds it to a list of all the found item.

16. Imagine that the Holiday Shopper application is a client-server application, where all of the data is held on the server (to allow for updates from multiple clients).
(a) Construct a RESTful API indicating, for each resource, the GET/PUT/POST/DELETE responses. Assume multiple users for this application
(b) Indicate how Get/Put/Delete are idempotent
(c) Indicate how the server is stateless
(d) Indicate how the client does not need to recall the structure of the server side resources

18. You are designing an email application for which you want to provide information to the user on their workflow.

The email application will recognize certain states: *overload* (more than 100 unread emails), *deluge* (in the last half hour, there has been an email every twenty seconds) and *all-is-well* (less than 5 unread emails).

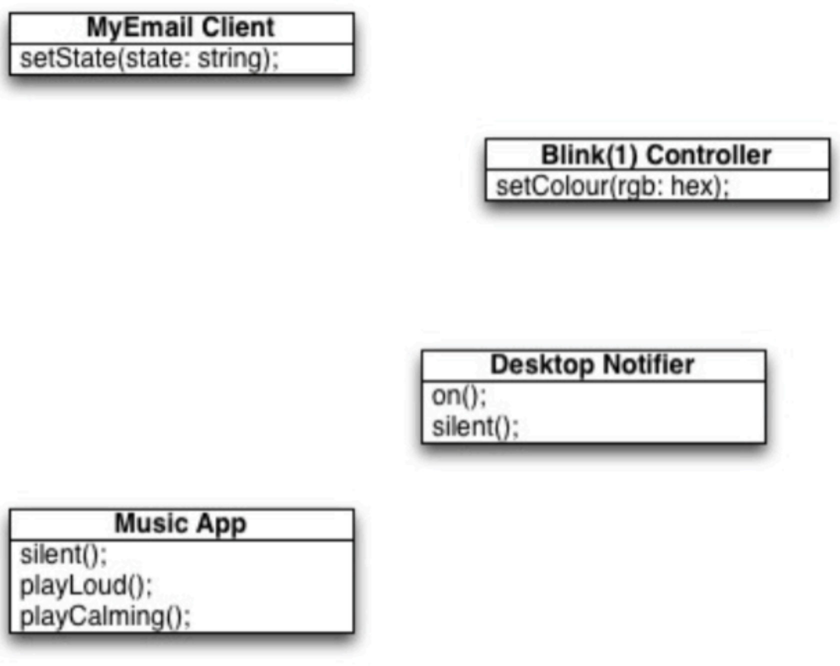
When in *overload* state, a blink(1) will turn red. When in *deluge* state, desktop notifications will be silenced. When in *all-is-well* state your computer will play calming music.

You have been given the class diagram below, which is missing all relationships between classes. Imagine you have been given Java interfaces representing the Observer design pattern. You will be implementing this class diagram in a language like Java or Typescript that allows a class to implement many interfaces. Apply the Observer design pattern to add this new functionality into the system described by the class diagram.

a) What class(es) will play the role of the Subject?
b) What class(es) will play the role of the Observer?
c) Add the necessary relationships into the given class diagram (adding the Subject and Observer interfaces needed) to use Observer to provide the desired behavior.
d) In what method(s) will the Observer(s) be registered with the Subject? Why?
e) In what method(s) will the Subject notify Observers? Why?
f) Do you need to add any methods to any classes in the diagram? If so, which methods and

why?

g) Describe what objects are involved and calls to objects are needed to have a blink(1) device turn red when in overload state? (You could use a sequence diagram to depict this but you can also use words.)


MyEmail Client
setState(state: string);
Blink(1) Controller
setColour(rgb: hex);
Desktop Notifier
on();
silent();
Music App
silent();
playLoud();
playCalming();