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.

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. **False, both perform development in iterations.**

3. User stories should always be accompanied by automated tests (such as JUnit), so that testing can be performed efficiently. **False, testing of some user stories cannot be automated**

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. **False, demos are provided at the Sprint Review meeting not the Daily Scrum meeting.**

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. **False, meeting is guided by three questions (1. What did you do yesterday? 2. What will you today? 3. What obstacles are in your way?)**

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

7. User stories describe implementation tasks that developers must complete. **False, user stories describe a desired feature from the user's viewpoint.**

8. A sequence diagram models the static relationships between objects. **False. A sequence diagram provides dynamic 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. **False. Teams typically work on the highest priority tasks first, but completion times facilitate scheduling.**

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? **No midstream changes makes for an efficient process**

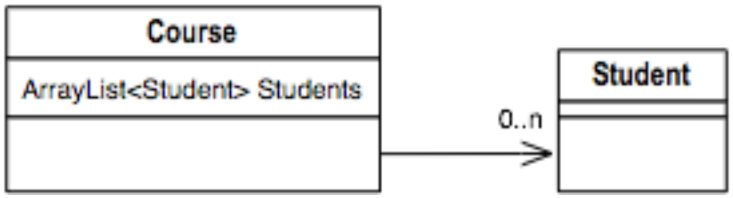
9. In the area of Agile methods, what does the acronym INVEST stand for? **Independent, Negotiable, Valuable (to users, etc…), Estimable, Small, Testable**

10. What is one of the major problems caused when user stories are too big? **Effort estimate could be inaccurate**

11. What is the purpose of the Scrum Master in the Scrum methodology? **“Facilitates Scrum process” OR “Helps resolve impediments” OR “Shields team from external interferences”**

12. What is one problem that is addressed by Collective Ownership of code in Extreme Programming? **Over specialization by individual risks major problems if that person leaves.**

13. What is the glaring error in this diagram: **The field “Students” is not needed, as its already present in the association to multiple students.**



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.

Hello user!

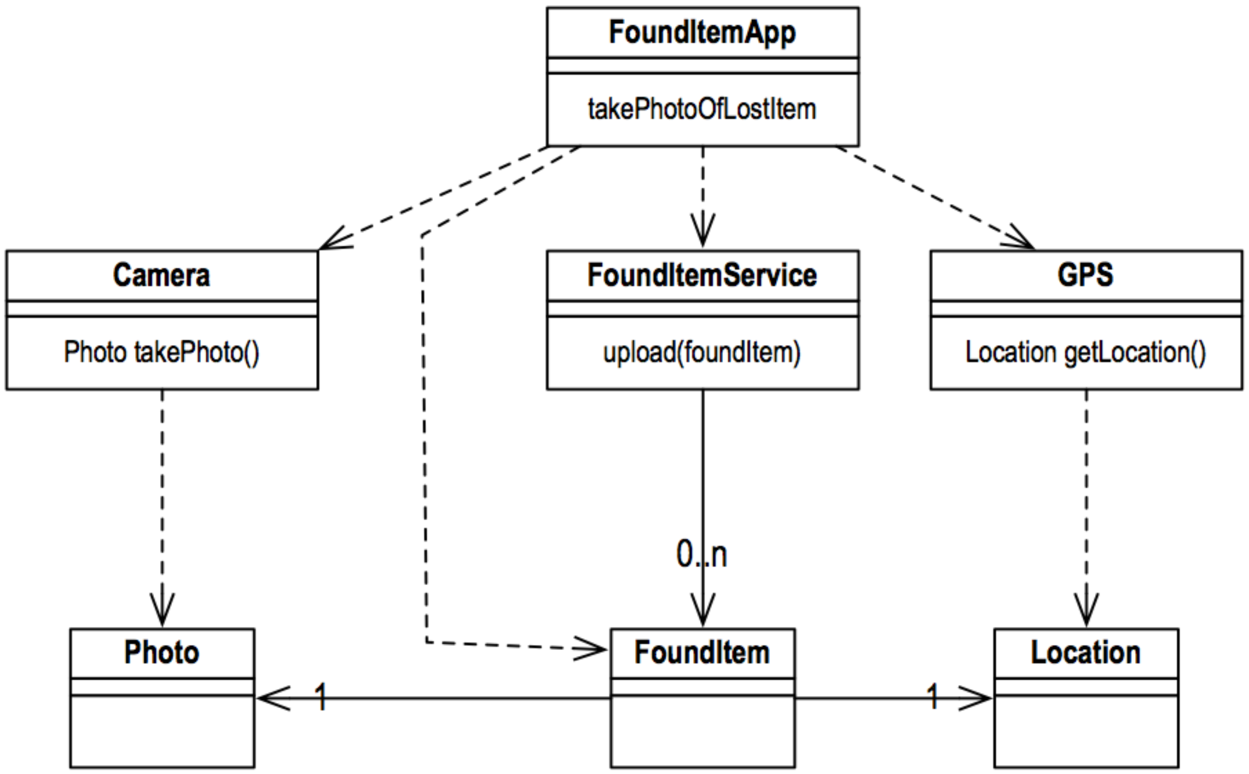
We have created an app that lets you plan your holiday shopping, such that you can have a list of people to shop for, and add gift ideas for each of them.

Please use this application to add a friend to your friends list, and add two gift ideas for that friend.

Please remember to think out loud while you work.

Thanks so much!

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 items



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

RESOURCE	GET	PUT	POST	DELETE
users/	returns list of user IDs as links	not implemented	make a new user, add it to the list, return the user's ID	deletes entire list
users/userID	returns URI for friends list	not implemented	not implemented	deletes user
users/userID/friends	returns list of friendIDs	not implemented	makes a new friend, adds it to the list, returns the new friendID	deletes entire friends list
users/userID/friends/friendID/	returns the URI for the giftIdeas list	not implemented	not implemented	deletes whole friend
users/userID/friends/friendID/ giftIdeas	returns list of idea IDs.	not implemented	makes a new giftIdeaID, adds it to the list, returns giftIdeaID	deletes whole giftIdeas list
user/userID/friends/friendID/ giftIdeas/ideaID	returns the text stored at ideaID	replaces entire contents of ideaID	not implemented	deletes ideaID

b) Get Put and Delete can be repeated without any cumulative effect. For instance, putting new contents to an ideaID will replace the contents of the ideaID, resulting in the same final state for ideaID after every put request.

c) The server maintains no knowledge of what stage the client is in their process. It has no "next" request, for instance.

d) The server returns all links needed for the client to navigate around the current resource structure. There is no need for the client to remember how the resources are stored. For instance, starting from a user, the client can get a series of links back from the server to navigate down to particular gift ideas.

17. 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? MyEmailClient
b) What class(es) will play the role of the Observer? Blink(1) Controller, Desktop Notifier and MusicApp
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. See diagram
d) In what method(s) will the Observer(s) be registered with the Subject? Why?
For lack of anywhere better, we could have the Constructor in the MyEmailClient add an Observer of each Observer class.
e) In what method(s) will the Subject notify Observers? Why?
Methods in the MyEmailClient class will recognize a given state and call the “setState” method of MyEmailClient. The setState will then call “notifyObservers”.
f) Do you need to add any methods to any classes in the diagram? If so, which methods and why?

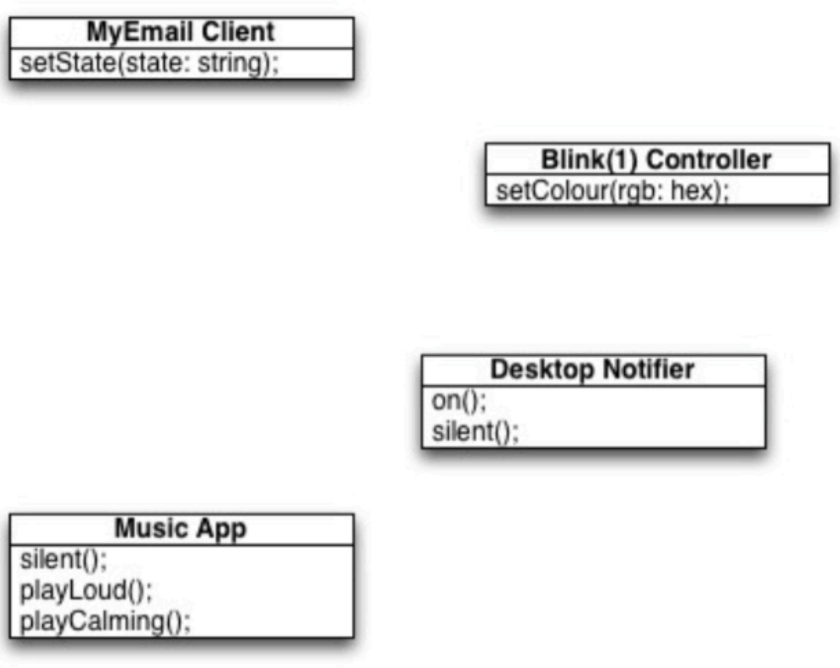
   We need to implement any methods defined by interfaces that classes now extend, such as registerObserver, notifyObserver in MyEmailClient and update in all of the classes implementing Observer (i.e., Blink(1) Controller, DesktopNotifier and MusicApp.

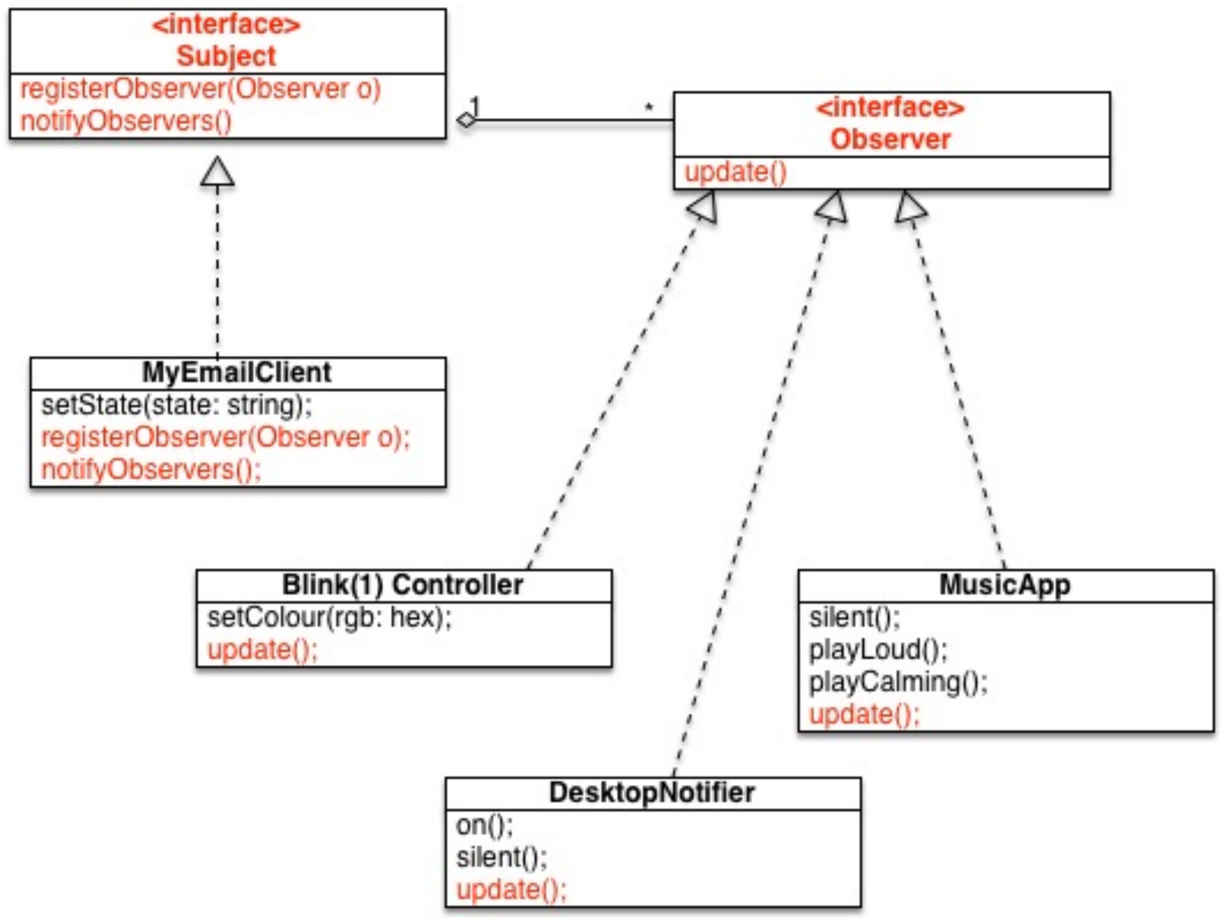
g) Describe what objects are involved and calls to objects are needed to have a blink(1) device turn red when in overload state? (You could use a sequence diagram to depict this but you can also use words.)

   As part of the constructor for the MyEmailClient class, an object of the Blink(1) Controller class will be created and registered as an Observer of MyEmailClient (i.e., there will be a call to registerObserver on MyEmailClient).

   At some point, an overload state will be recognized as part of methods defined on MyEmailClient and setState() will be called with the Overload state indicated as a parameter. setState() will call notifyObservers passing “Overload” as the string parameter value.

   The notifyObservers method will call update on all registered observers, which will include the object of Blink(1)Controller, with “Overload” as the string parameter. Only the Blink(1)Controller update method will recognize the Overload state and will call setColour to display red.


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


<interface>
Subject
registerObserver(Observer o)
notifyObservers()
1
*
<interface>
Observer
update()
MyEmailClient
setState(state: string);
registerObserver(Observer o);
notifyObservers();
Blink(1) Controller
setColour(rgb: hex);
update();
MusicApp
silent();
playLoud();
playCalming();
update();
DesktopNotifier
on();
silent();
update();