CMPUT 301 Winter 2014 Final

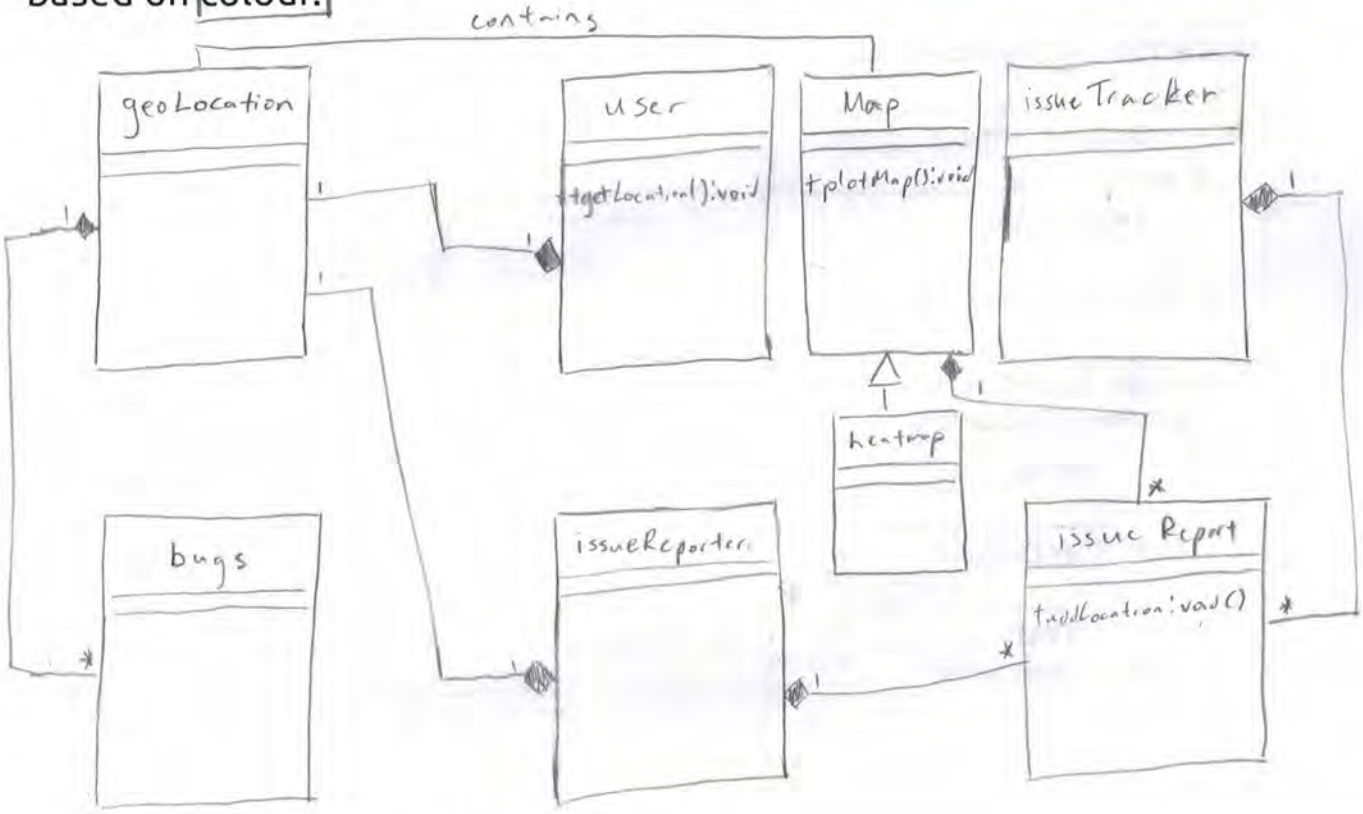
Name:_____

CCID:_____

Object Oriented Analysis: Potential Classes and Methods [2 marks]

Read the following paragraph and **draw** a UML class diagram of this scenario. This is about the domain, the requirements, not the final design. **Label** relationships. **Highlight** the nouns that become classes with **squares**, and the verbs and relationships with **circles**. Provide the basic abstractions, attributes, methods, relationships, multiplicities, and navigabilities as appropriate.

Our company specializes in geo-location based software. By geo-location we mean the current position (latitude and longitude) on Earth of the user of the software. This leads to a strange class of bugs that depend on the location of the user at the time. Some bugs only occur at certain locations. We want to augment our existing issue tracker by adding the issue-reporter's geo-location to the issue-tracker's issue report of our geo-location based software. We distribute more than 1 product that has geo-location capabilities. We want to plot a map of the geo-locations of issue reports. We also want a heat-map view of the map showing the frequency of issues based on colour.



CMPUT 301 Winter 2014 Final

Name:_____

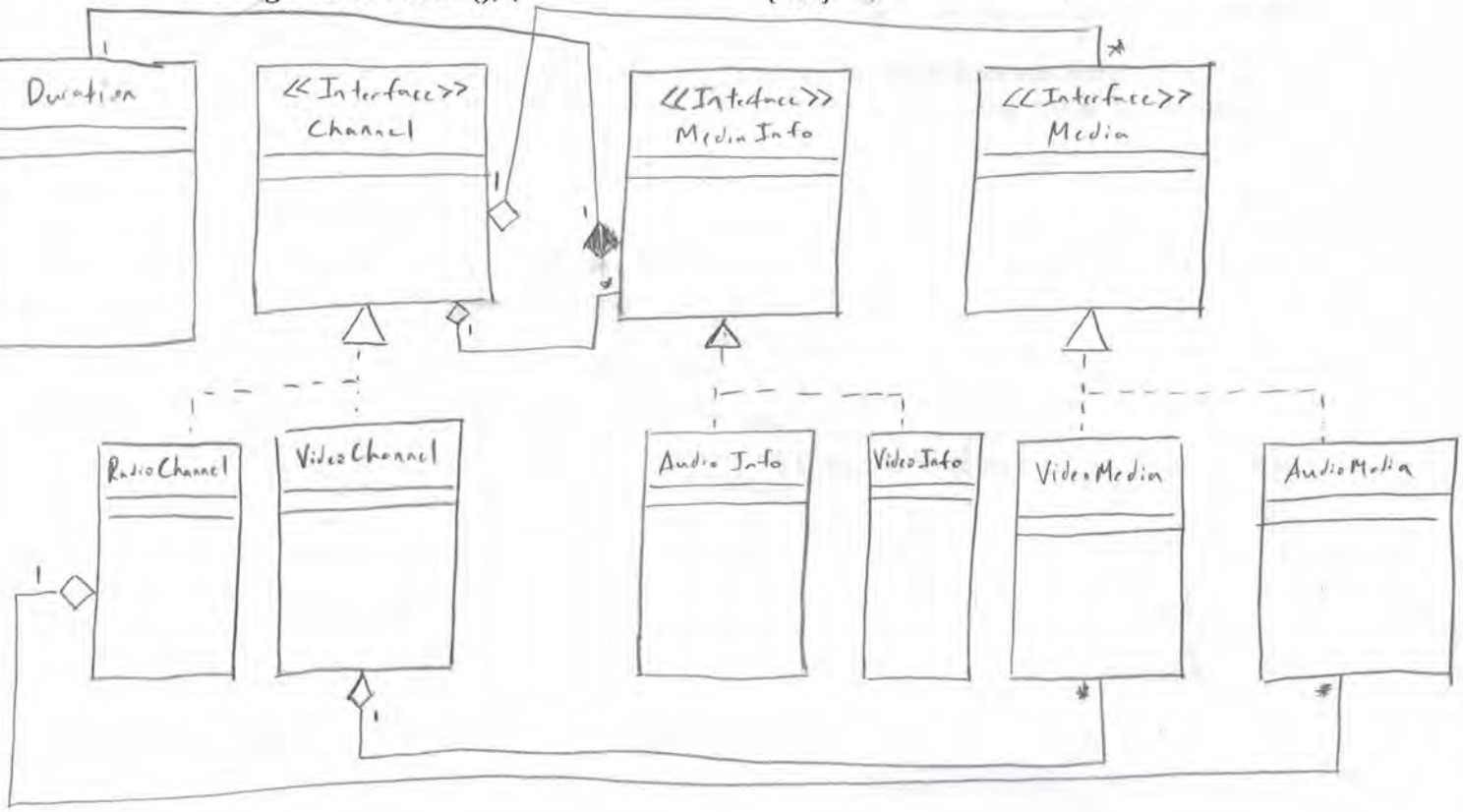
CCID:_____

UML: Association, Aggregation, Composition? [2 marks]

Convert this Java code to a **UML class diagram**. This Java code meant to represent a multi-media streaming system. Draw a well-designed **UML class diagram** to represent this information. Provide the basic abstractions, attributes, methods, relationships, multiplicities, and navigabilities as appropriate.

```
public interface Channel {
   public Media currentMedia();
   public String name();
   public MediaInfo nextMedia(); }
public interface MediaInfo {
   public String name();
   public Duration length(); }
public Duration {
   public Duration( Time start, Time end ) {...}
   public Time start() {...}
   public Time end() {...}    ...}
public interface Media {
   Audio getAudioStream();
   Video getVideoStream(); }
```

```
public class VideoMedia implements Media
{...}
public class AudioMedia implements Media
{...}
public class VideoChannel implements
Channel {
   Collection <VideoMedia> videos;    ... }
public class RadioChannel implements
Channel {
   Collection <AudioMedia> tracks;    ... }
public AudioInfo implements MediaInfo
{ ... }
public VideoInfo implements MediaInfo
{ ... }
```



CMPUT 301 Winter 2014 Final

Name:_____

CCID:_____

Use Cases and Use Case Diagram [2 marks total]

What are the titles of **three** primary use cases of the following situation:

Background:

Music is often encoded using a visual musical notation allowing musicians to read, interpret and play music on their instruments. A musical score is a song encoded using musical notation.

Description:

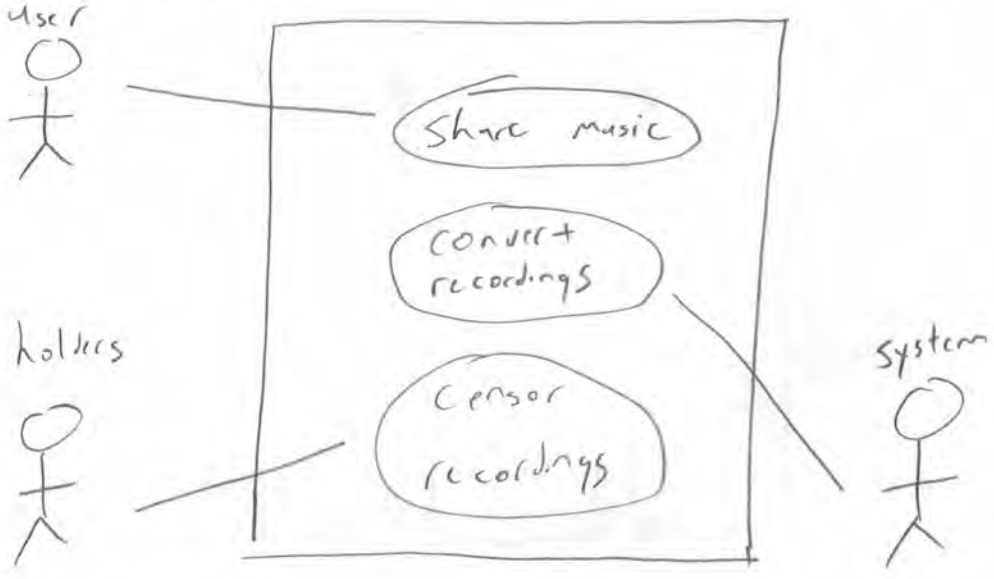
I want to make a system that allows users to record and share themselves playing music. The system can convert these recordings to musical notation automatically and annotate the recordings with these musical scores. Due to various intellectual property holders concerns, intellectual property holders can censor recordings and scores that they claim that they own.

Use case 1: User shares music

Use case 2: System can convert recordings

Use case 3: Intellectual property holders censor recordings

Now complete this **UML use case diagram**, including boundary, actors, use case bubbles and relationships between actors and use case.



CMPUT 301 Winter 2014 Midterm

Name:_____

CCID:_____

Use Case: [2 marks]

Convert this scenario or part of it into a single **use case** related to updating Fridge Tablet and getting relevant recipes.. **R**emember to include of all the actors. And cover common **exceptions**. You can use the back of the page if you need space.

Scenario: updating Fridge Tablet and getting relevant recipes.

I want to make something using the ingredients in my fridge, so on my **fridge's tablet** I click, "Update Fridge Contents". **Fridge tablet** shows me the last list of fridge contents. I click on recommend recipes. **Fridge tablet** shows me the ingredients that it will use in its query and then **Fridge tablet** queries the **recipe server** and gets a list of relevant recipes. Since I have lots of eggs the tablet recommends that I make an eggplant omelet or a tofu turkey omelet or a tofu turkey eggs benedict. I select eggplant omelet. The recipe is shown to me. once **I**'m done **I** tell **fridge tablet** to update my fridge contents to reflect the items that were consumed to make that recipe.

Use Case Name: getRecipe

Participating Actors: Fridge tablet, User, recipe server

Goal: Fridge Tablet gets relevant recipes and updates its contents

Trigger: user selects "update fridge contents" and selects recommended recipes

Precondition: Fridge Tablet knows fridge contents

Postcondition: recipe is shown and fridge contents is updated at user prompting

Basic Flow (back page use is OK):

1. user clicks "update fridge contents"
2. Fridge tablet shows fridge contents
3. User clicks recommend recipes
4. Fridge tablet shows ingredients in query and queries recipe server
5. Recipe server returns list of relevant recipes to tablet
6. user selects recipe
7. Fridge tablet shows recipe
8. user selects done on fridge tablet when finished making recipe and contents are updated

Exceptions (back page use is OK):

1. There are no matching recipes
   1.1 recipe server returns error
2. Fridge has no contents
   2.1 Fridge tablet displays error

CMPUT 301 Winter 2014 Final

Name:_____

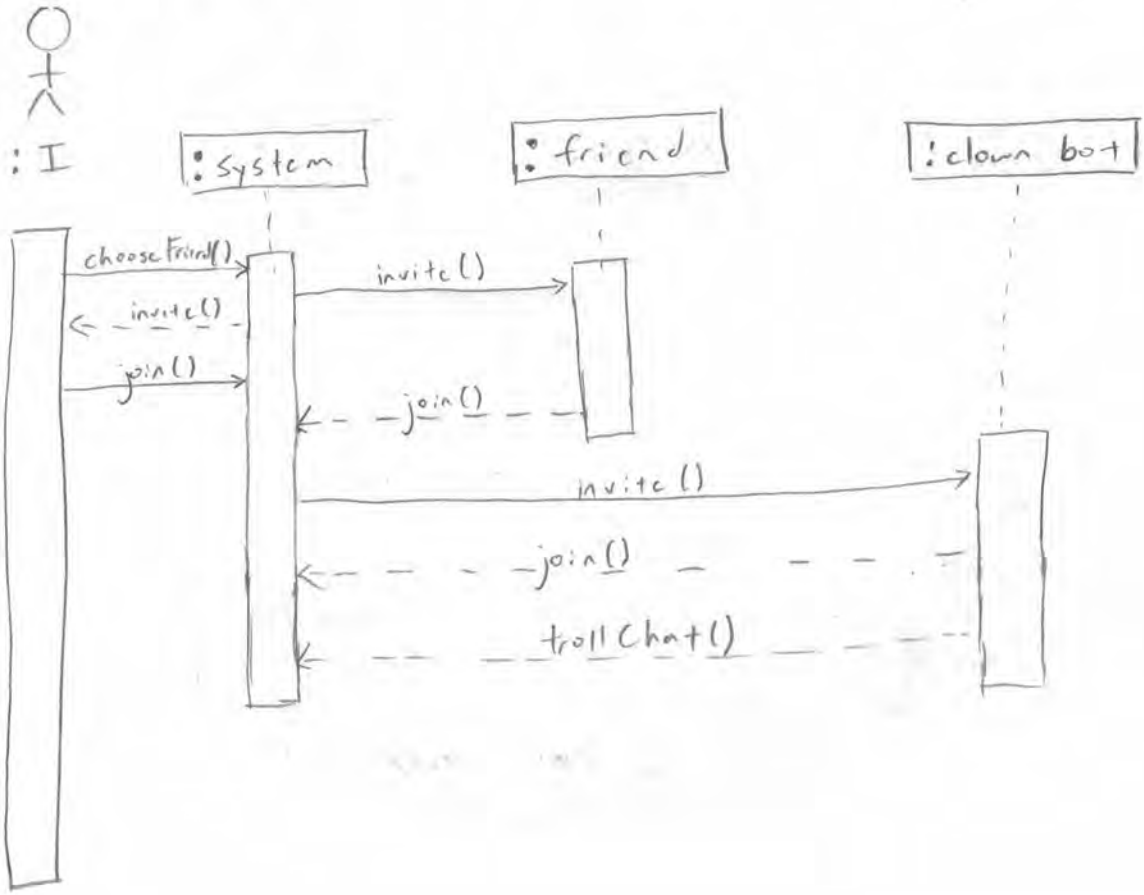
CCID:_____

UML Sequence Diagrams: [3 marks]

Convert this use case sequence of steps into a **sequence diagram,** remember to include all the **actors,** the **roles**, the **components**, the **lifelines**, and **activations!** and use good names for the methods.

Use Case Sequence: Setting up Video Clown Chat

1. I choose my friend from my friend list.
2. I click "invite to video chat" beside my friend.
3. The system invites my friend and I to a video chat.
4. I select join video chat
5. My friend selects join video chat.
6. The system connects my friend and I to a shared video chat.
7. A virtual clown bot is joined to the video chat by the system to make it less boring.



CMPUT 301 Winter 2014 Final

Name:_____

CCID:_____

Software Processes: [3 marks]

[1 mark] In SCRUM what is a daily standup meeting and what are the questions asked during the standup meeting?

daily standup meeting: short meeting where everybody stands up so that they're uncomfortable

questions: what did you do?
what are you going to do?
what is blocking you?

[1 mark] Using Git repositories **how** would you enable or help track an iterative software development process?

by keeping branches of each cycle of development

[1 mark] How does test first development work? How does test first development affect the design of software?

writing tests for the code before actually developing the software

test first development allows for immediate feedback in coding and can save time on debugging

CMPUT 301 Winter 2014 Final;

Name:_____

CCID:_____

Human Error and User Interfaces: [2 Marks]

[1 mark] Some traffic lights in Edmonton are sideways (horizontal, left green, right red) while most are up and down (vertical, bottom green, top red).
A) **Which** subset of the population will be challenged by a sideways traffic light configuration?
B) **How** would you **redesign** these light switches?

A) Color blind people who have memorized the pattern of traffic lights

B) for sideways make the left red and right green for consistency

[1 mark] What is a mode error? How does one prevent mode errors in software?

When you think something is one state but it's is another

to prevent mode errors, make the mode more distinctive

CMPUT 301 Winter 2014 Final;

Name:_____

CCID:_____

Design Paterns: [3 Marks]

Read the following problems, then choose and a)**NAME** the design pattern and b)**EXPLAIN** why this design pattern is the most appropriate solution.

1) You are making a shared canvas paint program where multiple users draw on the same shared canvas. The users can paint strokes, draw pencil lines, and erase elements all together on the same canvas.

Composite because elements can be added to the canvas and can be used all together

2) You're making a system that can respond to natural language queries such as "I want some horse radish". This system provides responses through a series of dynamicly loaded plugins that can be loaded and unloaded by the user at any time. → dynamically

observer because we are dynamically removing and adding plugins

3) You're making a role playing game and it has an inventory system where by boxes, sacks, chests, and bags can hold other containers. Some of the containers have magical properties that imbue the items contained within with properties like fire or lightning.

Composite because an inventory is a tree like structure holding containers that can hold items

CMPUT 301 Winter 2014 Final;

Name:_____

CCID:_____

OO Principles: [2 marks]

[1 Mark] **Explain** how the **replace conditional with polymorphism** refactoring applied to the **switch statement** bad smell increases or decreases **coupling**?

it increases coupling since there is more dependence on other classes as you are inheriting from other classes

[1 Mark] **Explain** how coding to the **specification** rather than the **implementation** increases or decreases **coupling**.

coding to specification increases coupling since it is not following implementation which is structured to account for design challenges for things such as high coupling

CMPUT 301 Winter 2014 Final;

Name:_____

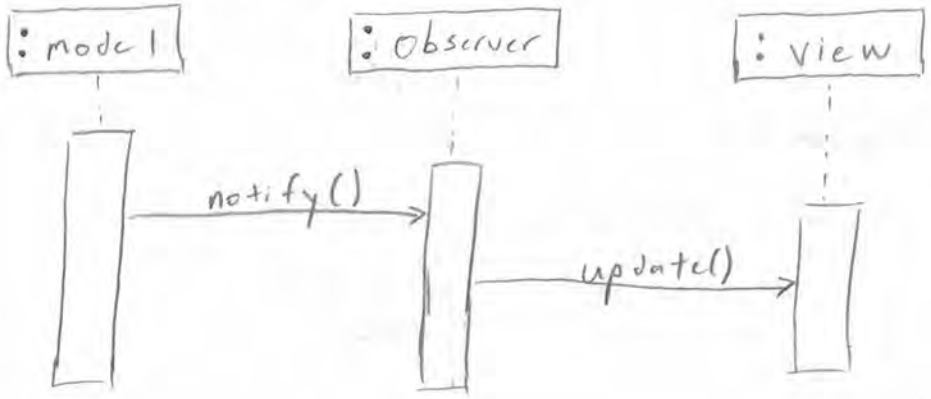
CCID:_____

MVC and Observer Pattern: [3 Marks]

[1 Mark] **How** does the observer pattern **decouple** a model from views? Do not define model, do not define view. Tell me **HOW** this pattern works and why it **DECOUPLES**.

this pattern has a single observer that represents all other observers and updates the subject. It decouples because it separates all the other observers from the subject.

[2 Mark] **Draw** the **UML Sequence Diagram** for the observer pattern when the model has been changed. In your sequence diagram show how an abstract model instance will update all of the listening views.



CMPUT 301 Winter 2014 Final;

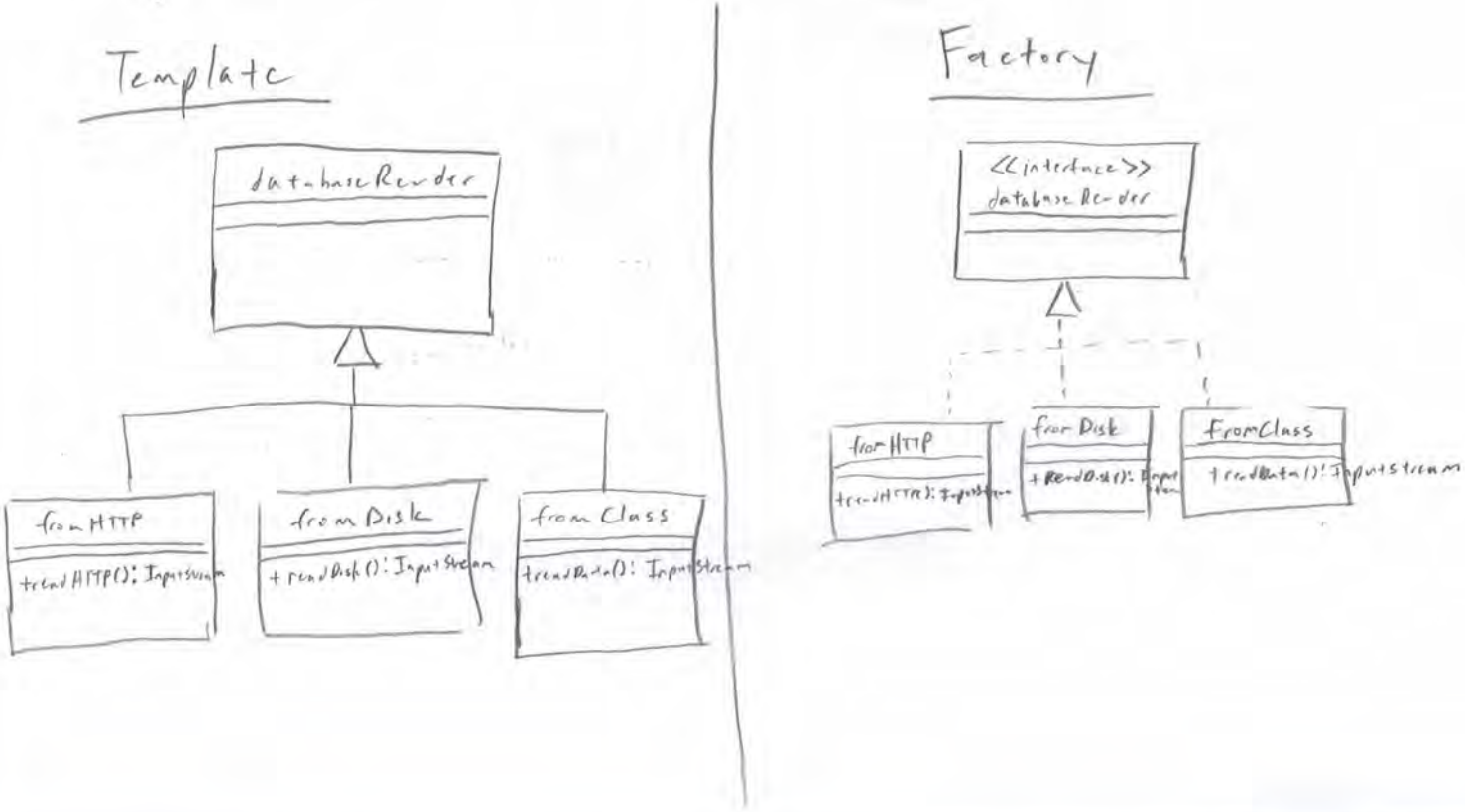
Name:_____

CCID:_____

Template Method, Factory Method and Refactoring: [2 Marks]

Provide the **UML class diagram** and of DatabaseReader and its subclasses after you have refactored the read() method using the **Template Method** Pattern and **Factory Method** Patterns. No sub class code is required, method names in the UML and the read method is good enough.

```
class DatabaseReader {
      ...
      Database read() {
            InputStream in = null;
            if (this.remote) {
                  in = new HttpInputStream( this.filename );
            } else if (this.fromDisk) {
                  in = new FileInputStream( this.filename );
            } else {
                  in = new ByteArrayInputStream(
                        this.data.getBytes("UTF-8")
                  );
            }
            Database dbOut = databaseFromStream( in );
            in.close();
            return dbOut;
      }
}
```



CMPUT 301 Winter 2014 Final;

Name:_____

CCID:_____

Testing: [2 Marks] Write the code for a **mock object class** (MockPowerMeasurable) that will allow testing of line **12** of **PowerMeter** in **testBlownFuse** of **TestPowerMeter.** Write the code for **MockPowerMeasurable.**

```
// Prints 3D Shapes on a 3D printer in plastic
class PowerMeter {
      Wattage measurePower( PowerMeasurable pm ) throws PowerException {
            try {
                  Amperage amps = pm.measureCurrent( this );
                  Voltage volts = pm.measureVoltage( this );
                  return new Wattage( amps, volts );
            } catch (ProtectionFuseException e) {
                  // The fuse that protects the power meter
                  // has been blown and the unit is now incapable
                  // of operation until it is replaced
12:               Manager.getInstance().invokeShutdown("Please replace Fuse", e);
                  throw e;
            }
      }
}
class ProtectionFuseException extends PowerException {}
interface PowerMeasurable {
      Amperage measureCurrent( PowerMeter pm );
      Voltage measureVoltage( PowerMeter pm );
}
class TestPowerMeter extends TestCase {
      void testBlownFuse() {
            PowerMeter pm = PowerMeter();
            MockPowerMeasurable mpm = new MockPowerMeasurable();
            try {
                  Wattage w = pm.measurePower( pm );
                  assert(false, "This was supposed to fail");
            } catch (ProtectFuseException e) {
                  assert(Manager.getInstance.hasShutdown(),"Manager not shutdown");
            }
      }
}
```

class MockPowerMeasurable extends PowerMeasurable () {

ProtectionFuseException e = new ProtectionFuseException();

Manager.getInstance().invokeShutdown("Please replace Fuse", e);

throw e;

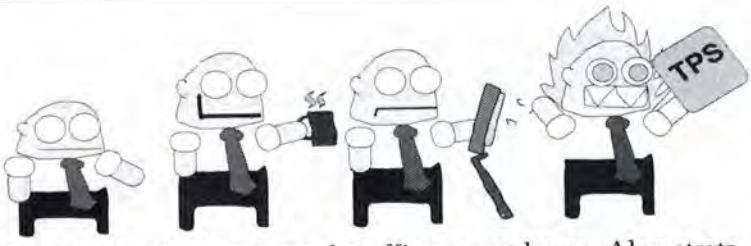
}

CMPUT 301 Winter 2014 Final

Name:_____

CCID:_____

UML State Diagram [3 marks total]31



Your unimaginative boss is making you code a videogame like Super Mario: **Alright Alan**. In **Alright Alan**, **Alan** explores an office environment, **Alan** has 3 tries (lives) to navigate the office to get home. Alan starts off short as *Small Alan*. If an enemy, a co-worker or his boss, manages to grab **Alan**, **Alan** will be forced to stay late and will lose a try (Caught Alan). But Alan can collect power-ups which help him avoid work!

- If **Alright Alan** collects a **TPS-report** he is invincible for 10 seconds and cannot be grabbed by an enemy. After 10 seconds, **Alan** will burn out and return to *Small Alan*. (*Invincible Alan*)
- If **Alright Alan** collects a **coffee**, he grow twice as tall, and if an enemy grabs him, he will revert back to his original short size, but will not lose a try! (*Caffeinated Alan*)
- If **Alright Alan** collects a stapler, **Alan** grows twice as tall AND he can fire staples at his coworkers, temporarily disabling them. If an enemy catches **Alright Alan** with a stapler, **Alright Alan** loses the stapler, and shrinks back to *Small Alan* but will not lose a try. (*Stapler Alan*)

Your job is to **make** a **UML state diagram** that models Alan's **states**: *Small Alan* (default), *Invincible Alan*, *Caffeinated Alan*, *Stapler Alan*, and *Caught Alan* (when grabbed and loses a try). Also in the **UML state diagram** be sure to show the transition between these states. Using this diagram I should be able to see how Alan transitions from Small Alan into Invincible Alan.