CMPUT 301 Fall 2012 Final

Name:____

CCID:____

Object Oriented Analysis: Potential Classes and Methods [3 marks]

Read the following paragraph and pull out potential nouns that may lead to classes and verbs that may lead to relationships and methods according to Object Oriented Analysis.

Gameplay takes place on a 2D battlefield, usually with some obstacles. Each player (from 2 to 6, computer or human) has an army of particles and a flag. The objective of the game is to assimilate all enemy particles. Players place a flag down and all of their particles follow the shortest path around the obstacles (and through enemy particles) to their player's flag. If a particle pushes through an enemy particle without the enemy particle pushing in the opposing direction, the enemy particle will be assimilated to the team of pushing particle. The game ends when one player assimilates all particles.

Simplified version of liquid war taken from Wikipedia (CC-BY-SA 3.0) Copyright 2012 Wikimedia https://en.wikipedia.org/wiki/Liquid_War. LiquidWar is a cool game http://www.ufoot.org/liquidwar/

List the potential Classes (appropriate nouns):

battlefield, obstacles, player, particles, flag,
path, obstacles

List the potential Actions/Methods/Relationships (appropriate verbs):

follow, shortest path to flag
assimilate enemy particles
place a flag
push in the opposing direction

CMPUT 301 Fall 2012 Final

Name:____

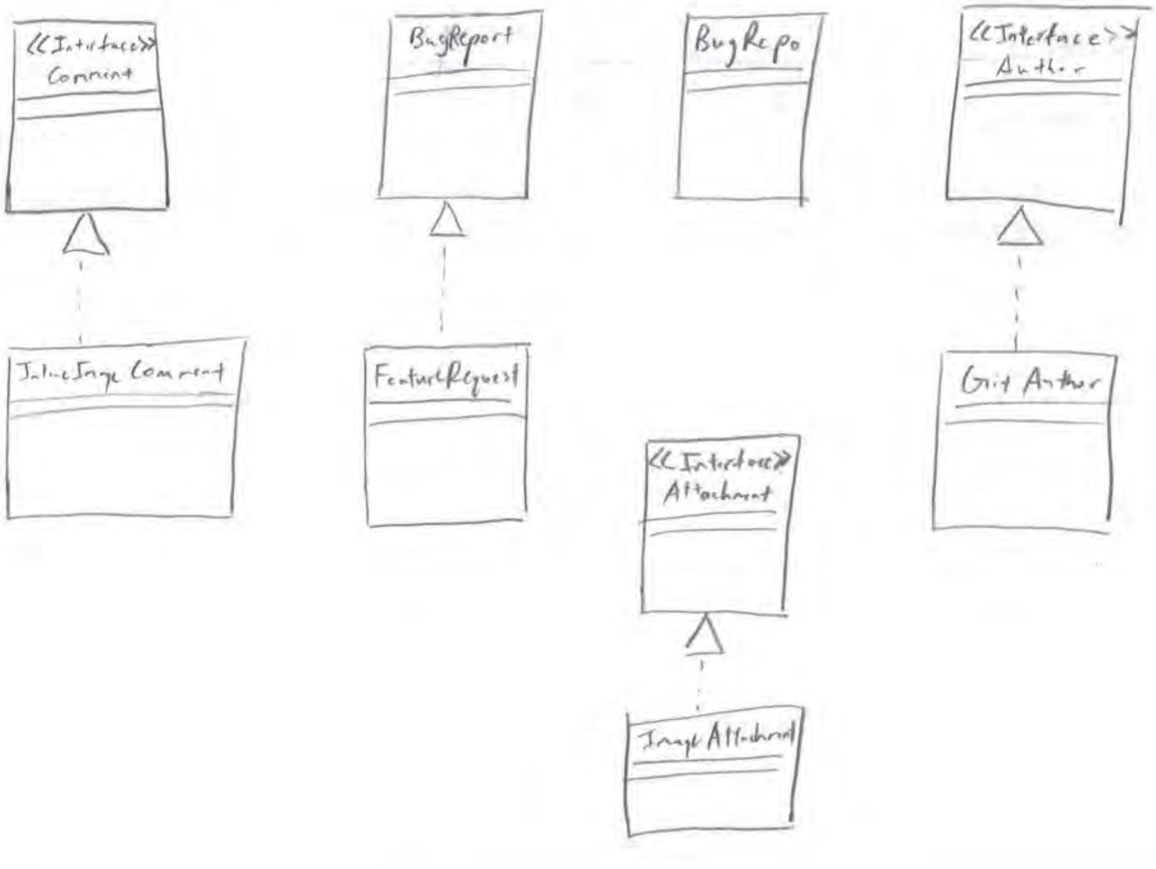
CCID:____

UML: Composition or Aggregation? [3 marks]

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

```
public class BugReport {
   String description;
   List<Comment> comments;
   Author author;
}
public interface Comment {
   Author author();
   String comment();
   List<Attachment> attachments();
}
class InlineImageComment
implements Comment {...}
```

```
public class BugRepo {
   List<BugReport> bugReports;
   ImageAttachment bugRepoLogo;
}
public class FeatureRequest extends
BugReport {...}
public interface Author { ... }
class GitAuthor implements Author
{ ... }
public interface Attachment { ... }
publc class ImageAttachment
implements Attachment{ ... }
```



CMPUT 301 Fall 2012 Final

Name:____

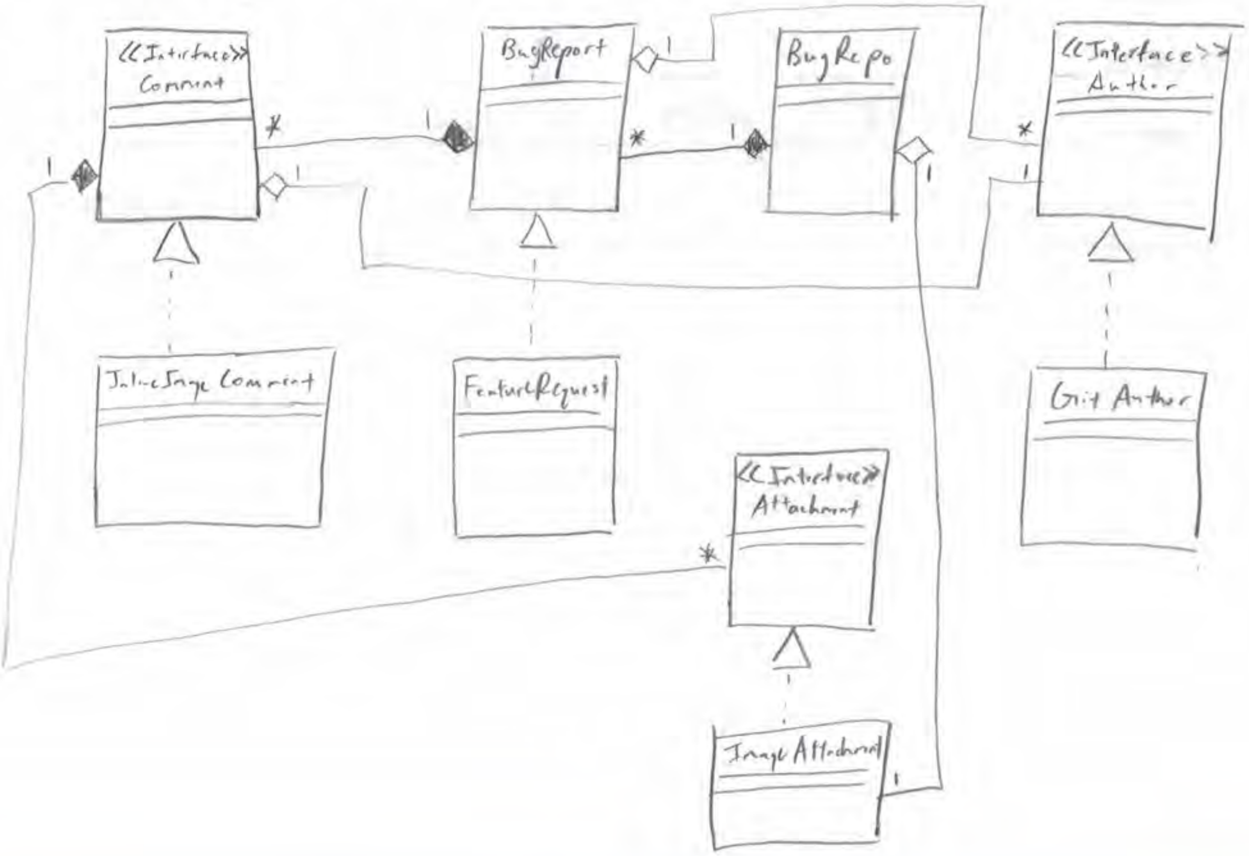
CCID:____

UML: Composition or Aggregation? [3 marks]

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

```
public class BugReport {
   String description;
   List<Comment> comments;
   Author author;
}
public interface Comment {
   Author author();
   String comment();
   List<Attachment> attachments();
}
class InlineImageComment
implements Comment {...}
```

```
public class BugRepo {
   List<BugReport> bugReports;
   ImageAttachment bugRepoLogo;
}
public class FeatureRequest extends
BugReport {...}
public interface Author { ... }
class GitAuthor implements Author
{ ... }
public interface Attachment { ... }
publc class ImageAttachment
implements Attachment{ ... }
```



CMPUT 301 Fall 2012 Final

Name:____

CCID:____

Use Cases and Use Case Diagram [3 marks total]

Background:

Facebook and other Web2.0 sites tend to try to limit users and their freedom. We want to design a Peer-To-Peer social networking platform, where each participant has the freedom to host their own instance and thus be responsible for their own privacy.

Goals/Stories:

As **node admin** I can host my own social network node.
As **node admin** I can administer users on my node.
As **node admin** I can control which external nodes can access my node.
As a **user** of a node I can publish posts to that node.
As a **commenter** I do not need to be a user and I may post comments on public posts.

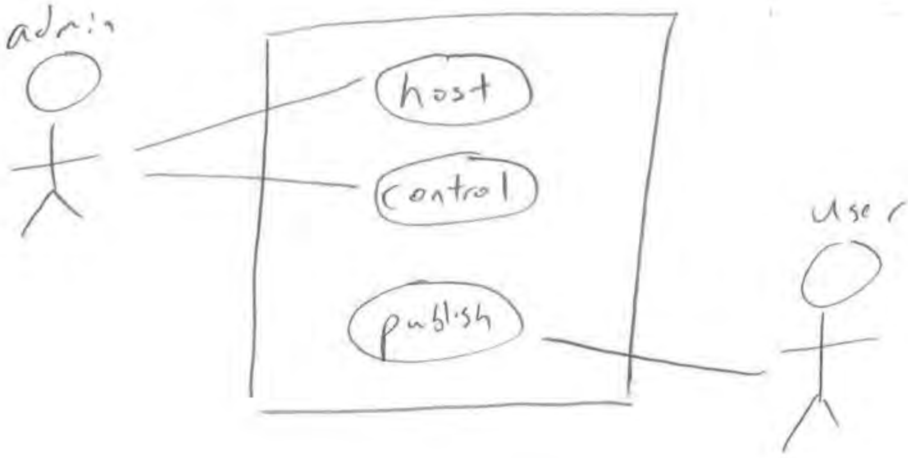
What are **three** primary use cases derived from the background and goals provided? (Only provide a good title for the use case).

Use case 1: host social network node

Use case 2: control who accesses node

Use case 3: publish posts to node

Now draw these uses cases in a **UML use case diagram**, including boundary, **actors**, use case bubbles and relationships between actors and use case.



CMPUT 301 Fall 2012 Final

Name:____

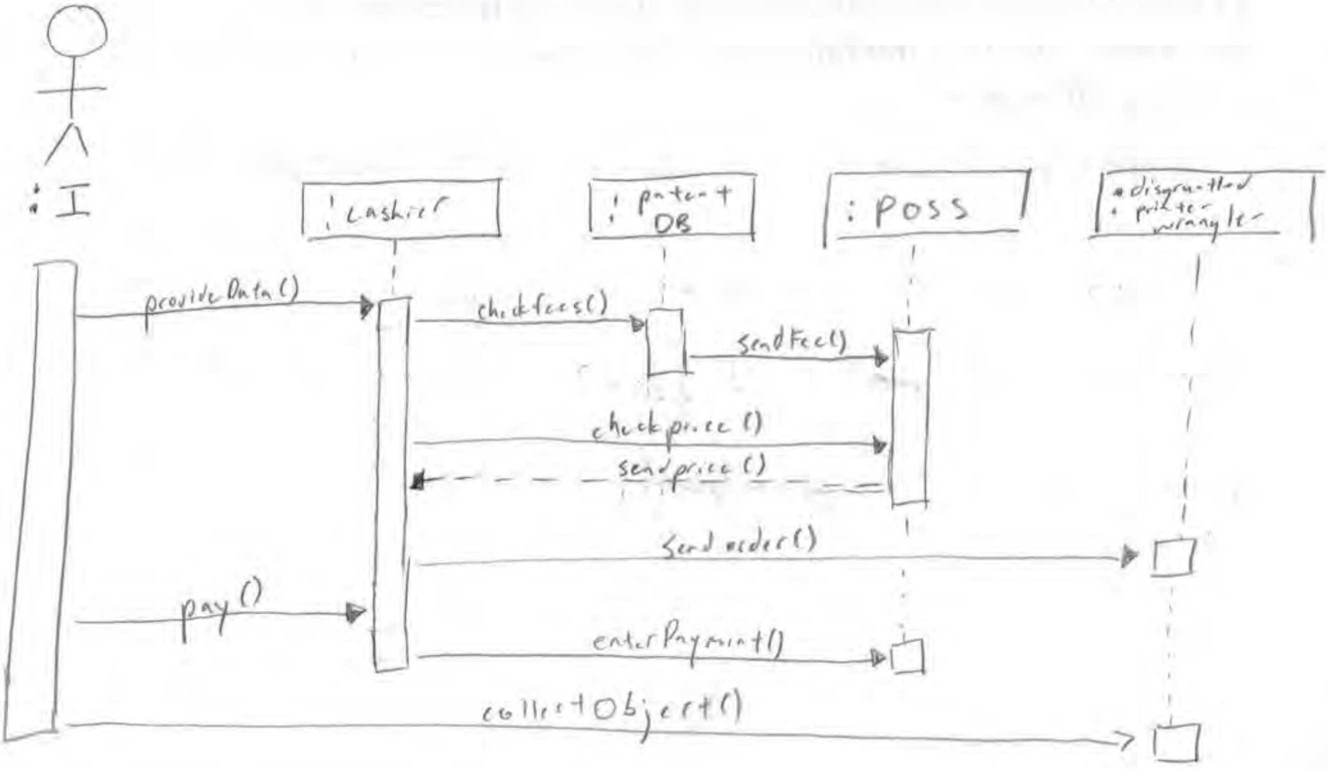
CCID:____

UML Sequence Diagrams: [3 marks]

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

Use Case: 3D Printing

1. **I** approach the **cashier** at the 3D printing kiosk and provide a USB key or a URL to the 3D object I want to print.
2. The **cashier** checks the file and passes it off to the **patent database** in order to check for patent fees associated with my provided 3D object.
3. The **cashier** checks the **point of sale system (POSS)** for a price that includes the patent-fee, the printing fee, and material fees.
4. The **cashier** yells the order to a **disgruntled printer wrangler**, who takes the order and sets up the printer.
5. The **disgruntled printer wrangler** prints my object on the **3D printer**.
6. **I** pay and the **Cashier** accepts payment and enters it into the **POSS.**
7. **I** walk to end of the kiosk and collect my object from the **disgruntled printer wrangler.**



CMPUT 301 Fall 2012 Final

Name:____

CCID:____

Software Processes: [3 marks]

[2 marks] Using Git repositories how would you enable or help track a **staged delivery process** where clients might be using older (but maybe stable versions) of your software?

- using branches from older versions and making sure the new changes work with the old builds,
- if changes don't work it can easily be reverted

[1 mark] Explain the relationship between the **iterative** model of software processes and the **waterfall** model. Focus on how they **related**, but also the primary **difference**.

- they are both software process models
- iterative has many cycles while waterfall only happens once (never revisiting a phase)

CMPUT 301 Fall 2012 Final;

Name:____

CCID:____

Human Error and Usability: [3 Marks]

[1 mark] What is the **name** of the law that describes the speed of choosing from a list of choices?

Hick's Law

[1 mark] Why is the **difference** in time of choosing between 2 and 8 choices greater than the **difference** between 80 and 100 choices?

average choice reaction time is greater

2 & 8 : $\ln 2(8+1) - \ln 2(2+1) = 0.81$

80 & 100 : $\ln 2(100+1) - \ln 2(80+1) = 0.22$

? how can this be done w/o calculator?

[1 mark] **Saccadic Masking**, what is it and how does it affect software?

When you miss something because you moved your eye.

It affects software because the user may become confused as to what is happening.

CMPUT 301 Fall 2012 Final;

Name:____

CCID:____

User Interfaces: [3 Marks]

[1 mark] What is one user interface method we can use that aids usability but also reduces human/operator error?

making sure every choosable action is legal by graying out invalid choices

[1 mark] Why must we be very careful about the colours we use in user interfaces (e.g. What's wrong with red and green)?

they can imply certain meanings
e.g. green means go
red means stop

[1 mark] Give 2 examples (or instances) of interface metaphors.

desktop

menus

CMPUT 301 Fall 2012 Final;

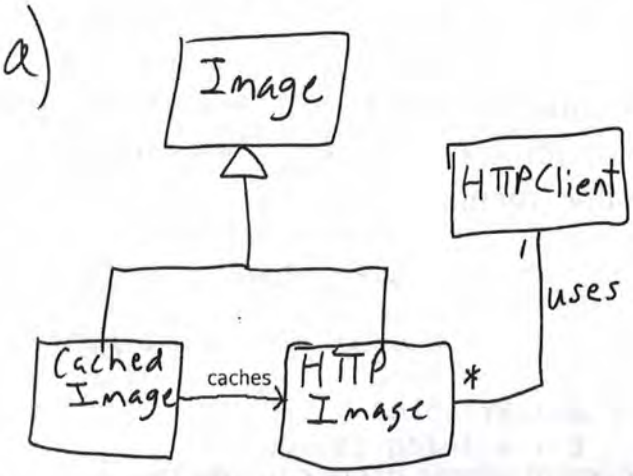
Name:____

CCID:____

Design Patterns: [3 Marks]

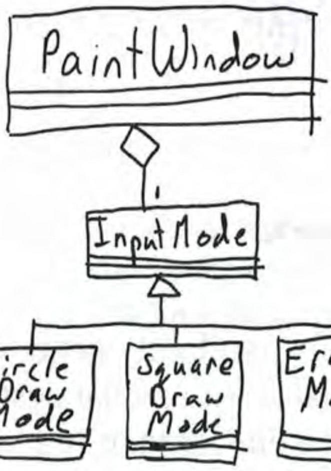
Identify and name each of these design patterns. If you make an assumption, explain it.

a)



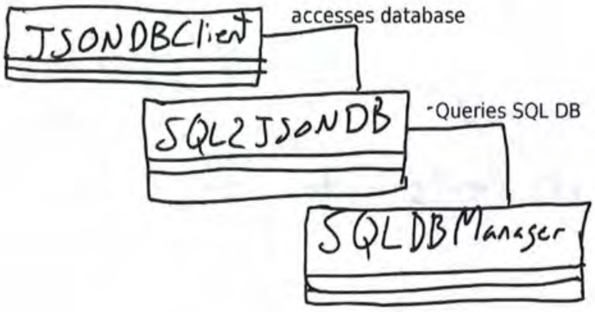
Proxy

b)



Command

c)



adapter

CMPUT 301 Fall 2012 Final;

Name:____

CCID:____

Design Paterns: [3 Marks]

Read the following scenarios, then **name** and **explain** the design pattern most appropriate to address this problem.

A) You want to implement macros or shortcuts that are can be learned from the user using your document manipulation user interface. These macros can be stored and replayed later on other documents.

command because we can store it, rerun it, and pass it around to other documents

B) You are building an event-based system where users can add plugins at run-time. These plugins can agree to handle some events but might only do so conditionally (under certain conditions).

factory method because we can decide which plugins to use

C) You're making a program that procedurally details (randomly generates) an entire universe lazily. You can drill down from galaxies, to solar systems, to planets, to countries, to people, to their blood, to atoms of their red blood cells, and further still.

decorator because you can keep on attaching universe details to the current universe object

CMPUT 301 Fall 2012 Final;

Name:____

CCID:____

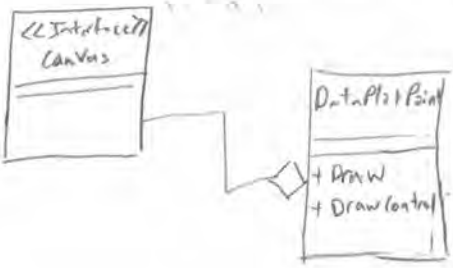
Refactoring: [3 Marks]

```
// We are implementing a statistical package that allows
// manipulatable control points that users can drag
// as well as plotting of other points.
interface Canvas {
    void color( Color c);
    void square( int cx, int cy, int w, int h );
    void circle( int cx, int cy, int r );
    ...
}
class DataPlotPoint {
    Boolean isControlPoint; //is this a control point
    ...
    void drawPoint( Canvas c ) {
        if (isControlPoint) {
            c.color( Color.YELLOW );
            c.square( point.x - 5 , point.y - 5 , 10, 10 );
        } else {
            c.color( Color.BLUE );
            c.circle( point.x , point.y, 5 );
        }
    }
}
```

[3 mark] **List** at least 3 bad smells one finds, and then at least 1 refactoring one could apply to this code snippet and then **draw** the **UML class diagram** of the relevant code after you applied these refactorings. State assumptions.

bad smells: procedural (if/else), drawPoint() does too much, Canvas is a blob class

refactoring: separate drawPoint into two methods



CMPUT 301 Fall 2012 Final;

Name:____

CCID:____

Bonus Testing: [3 Marks] Write a class for a **mock object** that will allow testing of line **11** of **ImagePrinter** in **testIOError** of **TestImagePrinter**

```
// Prints Images on a physical printer
// But unfortunately my printer works really well.
class ImagePrinter {
    Image image;
    Printer printer;
    ImagePrinter(Image image, Printer printer) { ... };
    void printImage() throws PrintFailureException {
        try {
            printer.print( image );
        } catch (PrinterIOError e) {
11:             throw new PrintFailureException(e.toString());
        }
        return true;
    }
    ...
}
// Test Case for ImagePrinter
class TestImagePrinter extends TestCase {
    // Let's ensure we throw the right exception when the IO
    // is interupted
    void testIOError() {
        ImagePrinter ip = new ImagePrinter(new Image(300,300),
                                           new MockPrinter());

        try {
            ip.print();
            assert(False, “Failed to throw Exception”);
        } catch (PrintFailureException e) {
            assert(True, “Should get here”);
        }
    }
}
```

```
Class   MockImagePrinter {
    ImagePrinter imagePrinter = new ImagePrinter(null, new
                                                   MockPrinter());

    return imagePrinter

}
```