CMPUT 301 Winter 2013 Midterm

Name:____

CCID:____

Object Oriented Analysis: Potential Classes and Methods [2 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.

My personal assistant (MyPA) takes notes for me. I dictate to it and MyPA writes down the notes and time-stamps them. I can retrieve old notes. MyPA will ask me every hour during work hours what I have been doing and record a work record if I respond. MyPA will produce summaries and graphs/charts of work records. I can also make TODO notes, and if I mention a location, when I am at that location (like the grocery store) MyPA can remind me of location-relevant TODO notes. An example of a location based TODO would be, “TODO buy milk when at grocery store.”

List the potential Classes [e.g. nouns]:

List the potential Actions/Methods/Relationships [e.g. verbs]:

CMPUT 301 Winter 2013 Midterm

Name:____

CCID:____

UML: **Composition** or **Aggregation**? [3 marks]

Convert this Java code to a **UML class diagram**. This Java code is meant to represent a multi-threaded job control queue, the dispatcher thread for the WorkerPool is required. 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 WorkerPool
{
   Collection<WorkThread> workers;
   Collection<Task> tasks;
   Thread dispatcher;
   public void addTask( Task task );
}
class WorkThread extends Thread {
   public void executeTask(Task task);
}
```

```
public class Thread { ... }
public interface Task {
   public void runTask();
   public Result finalResult();
}
public class CleanUpTask
   implements Task { ... }
public class MirrorWebSiteTask
    mplements Task { ... }
public class Result { ... }
```

CMPUT 301 Winter 2013 Midterm

Name:____

CCID:____

Use Cases and Use Case Diagram [2 marks total]

What are **three** primary use cases of the following situation: [1 mark]

Background:

I want to make a gift giving network, where anyone signing up will be expected to mail a gift to someone (randomly chosen) else after a deadline passes. The people will sign up voluntarily and at the end be assigned people to mail gifts to.

Description:

I want to define a network where gift givers can join (say a group of my friends). I want to be able to invite friends via email. I also want to specify a maximum price for the gift. I will set a deadline for participation. Friends who join will be notified of the random gift reciever assigned to them after a deadline. Once the deadline passes the system will randomly assign gift recievers to gift givers. The system will notify the gift givers of their gift receiver selection. Gift givers will mail their gifts to the gift recievers. Gift recievers can confirm reciept of their gift.

Use case 1:____

Use case 2:____

Use case 3:____

Now complete this **UML use case diagram**, including boundary, actors, use case bubbles and relationships between actors and use case. [1 mark]

CMPUT 301 Winter 2013 Midterm

Name:____

CCID:____

UML Sequence Diagrams: [3 marks]

Convert this use case into a **sequence diagram,** remember to include all the actors, the components, the lifelines and use good names for the methods. You can use the back of hte page if you need space.

Use Case: Buying an embroidered hat

1. **I** initiate a conversation with the store **clerk**, and request to buy an **embrodered hat**.
2. **Clerk** requests the **design I** wish to have on the hat as well as the hat **type**, and hat **colour**.
3. **I** provide the **clerk** with a **USB Keydrive** containing **1 picture file**, and **hat details**.
4. The **clerk** reads the **USB keydrive** and converts the **picture file** into an **embroidery file** (.emb) using the store's BMP->EMB **webservice**.
5. The **clerk** takes the appropriate hat from the inventory and the new **EMB file** and gets the **printer** to embroider the hat with my design.
6. **I** inspect the embroidered **hat**.
7. The **clerk** provides **me** with the total **cost**.
8. **I** provide the **clerk** with the appropriate **payment**.
9. **Clerk** processes my **payment** and returns the hat to **me**.

CMPUT 301 Winter 2013 Midterm

Name:____

CCID:____

Software Engineering: User Interfaces. [**2** marks]

Using principles of usable **user interfaces** and **graphic design**, design (**draw**) a UI that allows us to review the difference between changed lines in two versions of file (diff). Choose and layout appropriate **user interface elements** to address these requirements and constraints as well, use appropriate graphic design elements (organization, flow, etc.). Assume the user is using a touch screen (of any size) and a software touchscreen keyboard (so they can enter text). This UI will allow us to choose which version of the line we want in a merged version of the file. Think of a git-merge.

- Both versions of each line must be visible on the screen (and at least 1 line from each file)
- The following actions must be accessible to a touch screen user:

Go to previous change	Option to Highlight the different tokens
Go to next change	Option to Highlight the same tokens
Jump to an arbitrary change	Choose to keep the line from File 1
View more file 1 context (more lines from file 1)	Choose to keep the line from File 2
View more file 2 context (more lines from file 2)	Cancel changes and exit
Save and exit	

CMPUT 301 Winter 2013 Midterm

Name:____

CCID:____

Software Engineering: Software Development Processes [3 marks]
Keep the responses short. A long response that is not on topic is dangerous.

How does **reflection** or **self-evaluation** relate to **software processes** and **iterations**? [1 marks/3]

Explain the relationship between the **iterative** model of software processes and the **waterfall** model. Focus on how they **related**, but also the primary **difference**. [1 marks/3]

Explain **how** the **Model View Controller** pattern (**MVC**) decouples the **Model** from the **Views** and why this is an advantage [1 mark/3].