**Phillip Hoang 1409768**

-Listing potential classes, actions/methods/relationships (verbs). And **DRAW** UML class diagram based on them.

-Converting Java code to UML class diagrams

-Deriving Use Cases from goals/description, make **UML Use Case diagram** from this including boundary, actors, use case bubbles, and relationships between actor and use case

-Converting Use Case to **Sequence Diagram (UML Sequence Diagram?)**

-Identify Design Patterns based on picture (draw design the design patterns)

-Git  
 -How to track staged delivery process where clients might be using older (stabler) versions

-In **daily scrum meeting** why would you use **git log**?

-Human Error and Usability   
 -Name of the law that describes the **speed of choosing from a list of choices**

-Name of the law that defines **speed of clicking on a target**

-Which target is the **fastest to click**?

-Why does it take longer to click on the other targets?

-Why is the **difference in time of choosing 2 and 8** choices greater than the difference between 80 and 100

-What is **Saccadic Masking** and how does it affect software, how to design against it

-Use **Fitt's law** and **Hick's law** to explain which array of buttons would be faster.

-User Interfaces  
 -What is one UI method that aids usability but also reduces human error?

-Why must we be careful about **colours** we use in UI? *Colour blindness* -How to design light switches that aren't red and green

-Give 2 examples (or instances) of interface metaphors.

-Name of the law that estimates average time to make a simple decision, *n* choices vs *t* time

-Software Process  
 -Explain what a software development process is

-Relationship between **iterative model** and **waterfall model**. What is their primary difference?

-Provide an example of 2 different software development processes and how they differ from eachother

-Give 1 reason why the **Unified Process** is similar to the **waterfall process,** 1 reason why different

-Identify Design Patterns appropriate for (and explain why):  
 -Want to implement macros learned from the user. These macros can be stored and replayed later.

-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

-Making a program that procedurally details a universe lazily. Can go down from galaxies to solar systems to planets to countries to people to cells to atoms etc

-Have an algorithm for recognizing different kinds of minerals from photos. The algorithm needs specialized logic for each different mineral, but general control flow and logic can be shared

-Building gravity sim for planets. 3D view is hard to control and configure so want a 2D view and textview that shows planet state. Also want to be able to add or delete bodies in sim as it is running.

-Making mass photo editor where operations can be repeated across entire director of photos.

-Making city sim. People change over time but their identity does not. How do you model difference in behavior between a child and an old man who were the same person

-Making web interface to eclipse. Want to send requests to eclipse to open, view, modify, etc your projects via a web browser and have it executed in your eclipse IDE. Eclipse will run a webserver to do this.

-Making an OS abstraction layer to port apps between different platforms. You have defined interfaces, but the client code needs to get concrete instances of them. How to build appropriate concrete instances for the clients?

-OO Principles  
 -Explain how the **hide delegate** refactoring applied to the **message chains** bad smell increases or decreases **coupling**

-Explain how coding to the **specification** rather than the **implementation** increases or decreases coupling

-MVC and Observer Pattern  
 -How does observer pattern **decouple** a model from views?

-Refactoring (and Decorator)  
 -Find at least 3 bad smells, and at least 2 refactoring that could be applied to this code snippet. Then DRAW the UML class diagram of the code after refactoring

-Provide UML class diagram for xClass after refactoring xMethod() using **Decorator Pattern**

-Testing  
 -Write a class for a **mock object** that will allow for testing of line x of xClass

-Provide 5 good test cases for a function. (Max of 1 test per equivalence).

Clear:

2012F

2012W

2013F