CMPUT 301 2018 Fall Term Final Exam
TEST VERSION: Zubat

by Abram Hindle (c) 2015-2018
hindle1@ualberta.ca

Name:_____

CCID:_____

Student Number:_____

Question	Mark	Out of
Object Oriented Analysis: Potential Classes and Methods		3
UML: Association, Aggregation, Composition?		3
Template Method Pattern		3
UML Sequence Diagrams		3
Decorator Pattern		3
Mock Objects		3
Behavioural Patterns		3
GrabBag		3
Testing		3
Refactoring		3
TOTAL		30

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

Object Oriented Analysis: Potential Classes and Methods [3 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.

I like to eat cheese. I want to record my cheese eating. I want to record the amount and types of cheese that I ate, and when I ate them, and my reviews of the cheese. Furthermore I want to see the timeline of the cheeses eaten throughout the session. I often live-stream my cheese eating on *Twitch*, a video streaming platform, where I give reviews of cheeses and my viewers give me donations. I need these reviews and donations added to a timeline of events so that I can analyse the timeline so I can optimize my cheese eating strategies to maximize my twitch revenue.

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

UML: Association, Aggregation, Composition? [3 marks]

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

```
interface Keyboard {
    public void clickClack();
}
class OnScreenKeyB implements Keyboard {
    Screen s;
    public void clickClack() { ... }
    ...
}
public interface Screen {
     public void display(UI ui);
}
class OLEDScreen {
    public void display(UI ui) { ... }
    ...
}
```

```
public class Phone {
    Keyboard ky;
    Screen s;
    ...
}
class DiPhone extends Phone {
     DiPhone(Keyboard k, Screen s) { ... }
}
class MyPhone extends Phone {
     MyPhone() {
        ky = new OnScreenKeyB();
        s   =  new OLEDScreen();
     }
     ...
}
```

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

Template Method Pattern: [3 marks]

Refactor those conditionals (hasSlicer, logOps) away using polymorphism. Modify this code to use the **template method pattern**. Only show new or changed or moved code. Show the resulting UML class diagram, be sure to include a “CandyRopeMaker with Slicer” subclass.

```
class CandyRopeMaker {
    boolean hasSlicer = false;
    boolean logOps = false;
    CandyRopeMaker(boolean sliceit, boolean logit) {
       hasSlicer = sliceit;
       logOps = logit;
    }
    CandyRope processCandyRope() {
       if (logOps} { logger.log(“Start Process Candy Rope”); }
       for (int i = 0 ; i < 10; i++) {
           emitCandy();
       }
       if (hasSlicer) { cutRope(); }
       if (logOps} { logger.log(“Finish Process Candy Rope”); }
    }
    void emitCandy() { ... }
    void cutRope() { ... }
}
```

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

UML Sequence Diagrams: [3 marks]

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

I park my car. I look at the parking spot label, it says #A15. I go to the parking payment kiosk. I see a map of parking spots and I see my spot, #A15. I click on my spot #A15 and then the machine asks for my credit card. I insert my credit card and remove it. The machine tells me to insert my credit card upon return. I go out shopping and I come back with a wide assortment of cheeses. I swipe my credit card at the parking payment kiosk. It prints a receipt showing how much I was charged for parking. $3.92? What a rip-off!

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

Decorator Pattern: [3 Marks]

1. **Code** a decorator that can lowercase generateWord() of RandomWordGenerator.
The .toLowerCase() method on strings returns a lowercase version of a string,
for example “A”.toLowerCase().equals(“a”) evaluates to true.

```
public interface WordGenerator {
	public String generateWord();
}
public class RandomWordGenerator implements WordGenerator {
	static final char letters[] = “abcdefgh...zABC..Z”.toCharArray();
	public String generateWord() {
		char[] c = new char[8];
		for (int i = 0; i < 8; i++) {
			c[i] = letters[ (int)(Math.random() * letters.length)];
		}
		return new String(c);
	}
}
```

2. Decorate the instance of RandomWordGenerator (wg) so that the following code print lowercase words:

```
WordGenerator wg = new RandomWordGenerator();
// decorate the instance below
```

```
for (int i = 0; i < 100; i++) {
	System.out.println( wg.generateWord() );
}
```

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

Mock Objects: [3 Marks]

Write the code for a mock object to help reach and test the bolded code block marked “REACH THIS CODE” so that the testcase testBadResponse() in ErrorResponderTest will pass.

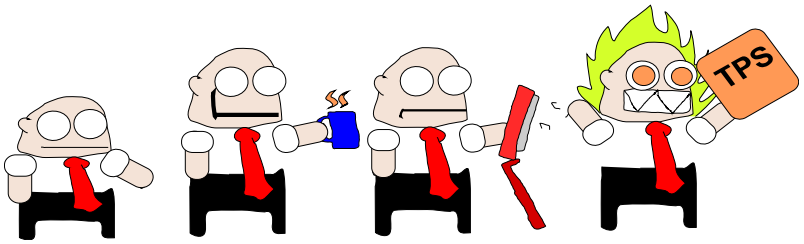
```
interface HTTPRequest {
   public int getCode();
   public String getContent();
}
class ErrorResponder {
   Exception exceptionForCode(HTTPRequest h) {
      int code = h.getCode();
      if (code >= 300 && code < 400) {
         return new RedirectException();
      } else if (code >= 400 && code < 500) {
         return new BadRequestException();
      } else if (code >= 500 && code < 600) {
         doSomeFiveHundredMagic();          // REACH THIS
         return new BadResponseException(); // CODE
      }
   }
   ...
}
class ErrorResponderTest extends TestCase {
   void testBadResponse() {
      HTTPRequest errorRequest =                                    // finish this line
      ErrorResponder e = new ErrorResponder();
      assertEquals(new BadResponseException(),
         e.exceptionForCode(errorRequest));
   }
}
```

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

Behavioural Patterns: [3 Marks]



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

- If **Alright Alan** collects a **TPS-report** he is invicible for 10 seconds and cannot be grabbed by an enemy. After 10 seconds, **Alan** will return to his previous state. (Invincible *Alan*)
- If **Alright Alan** collects a **coffee**, he grows 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 original size but will not lose a try. (*Stapler Alan*)

1. What design pattern is appropriate for modelling Alright Alan's change of behaviour?

2. Draw the **UML class diagram** of a Alright Alan and Alright Alan's behaviour using the appropriate design pattern. Required methods are run, jump, collideWithEnemy, fireStapler.

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

GrabBag: [3 Marks]

1. How is the **cohesion** of an existing object affected when we make it a model object in MVC where the object takes the role of observable in the observer pattern?

2. Abram made a silly mistake and committed the file osxfilenamecorrection.java to the repo. It's supposed to be called OSXFileNameCorrection.java as it contains the class OSXFileNameCorrection. **Using the git command line**, h*ow do you fix Abram's error that exists on the "origin" repo on the "master" branch?* Using **git**, correct the filename and share the correction back to origin's master branch. Assume you have checked out the HEAD of master which contains this mistake.

3. In agile software development **what** is collective code ownership and **Explain** how collective code ownership affects who can change what part of the source code of a project.

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

Testing: [3 Marks]
Write comprehensive testcases for this OurSet class's **intersect** method that models Set Intersection (the items of each set that are shared between both sets) that cover **all** equivalence classes for **OurSet.intersect**. Use jUnit style.

```
class OurSet {
    // Empty Set constructor
    OurSet() { … }
    // Constructor from array
    OurSet(Object[] objectsInSet) { … }
    // Returns an OurSet object that contains the intersection of OurSets "this" and "input".
    OurSet intersect(OurSet input) { ... }
    // Adds an object to the set
    void add(Object o) { ... }
    ...
}
```

CMPUT 301 Fall 2018 Final;

Name:_____

CCID:_____

Refactoring: [3 Marks]
Read the following code:

```
class LandCollection {
    ArrayList<LandPlot> lands;
    int calculateArea(LandPlot l) {
        return l.width * l.height;
    }
    int totalArea() {
        int total = 0;
        for (Land l: lands) {
            total += calculateArea(l);
        }
        return total;
    }
}
```

```
class LandPlot {
    int width;
    int length;
    LandPlot(int width, int length) {
        this.width = width;
        this.length = length;
    }
}
```

1. **Name** a code smell that this code suffers from that you will refactor.

2. **Name** an appropriate refactoring to fix the code smell you named. Then apply that refactoring to the **code**, then **draw** a UML class diagram of the resulting classes.