CMPUT 301 2016 Fall Term Final Exam
TEST VERSION:

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

Name:_____

CCID:_____

Student Number:_____

Question	Mark	Out of
Object Oriented Analysis: Potential Classes and Methods		3
UML: Association, Aggregation, Composition?		3
Use Case		3
UML Sequence Diagrams		3
Decorator Pattern		3
Design Patterns		3
State Pattern + State Diagram		3
Optimization and Observer Pattern		3
Testing		3
Refactoring		3
TOTAL		30

CMPUT 301 Fall 2016 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.

The octodroid has 8 legs, 6 of these legs are for walking and 2 of them are for holding the octo cannons. The octodroid is the end-boss for level 5. The octodroid has 3 behaviours which it repeats in a loop faster and faster until it is destroyed. Once all of the legs have been damaged and destroyed by the player the octodroid will be destroyed. Each leg has 250 hit points, and the octo cannons have 600 hit points each. The octo cannons are invincible until the 6 walking legs have been destroyed. The 3 states of the octodroid are rapid fire, targeted fire, and laser beam sweep. Each state starts lasts 10 seconds long at the start and then are decremented until they reach 2 seconds each.

CMPUT 301 Fall 2016 Final

Name:_____

CCID:_____

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

Convert this Java code to a **UML class diagram**. This Java code was obfuscated**.** 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 IZ {}
public interface IA extends IZ {
    public IA aIA(IA ia);
}
public interface IB extends IZ {
    public IA aIB();
}
public class CC {
   Collection<IB> ibs;
   IA ia;
   IA cIA() { return new CA(); }
}
```

```
public class CA implements IA {
   CA(IA clone);
   public IA aIA(IA ia);
}
public class CBA implements IB, IA {
   public IA aIA(IA ia) {
      return new CA(this);
   }
   public IA aIB();
}
public class QBA {
   private IA[4];
   private IB[5];
   Collection<IZ> interleave();
}
```

CMPUT 301 Fall 2016 Midterm

Name:_____

CCID:_____

Use Case: [3 marks]

Convert this scenario into a single **use case** related to organizing github tasks in the issue tracker. **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: Finding duplicate bugs from Github repositories

I search my github project issues for an issue tagged “task”. I see a “task” about refactoring the DreamController. I write a comment “I claim this”. The task tracker inserts the current time into the issue, and tags the issue with my username, and creates a new branch. I checkout that branch. I work on the issue, finally I have to go pick up my cousin from work so I commit and add the word “checkout” to the commit log message. I push the changes. The task tracker posts a comment on the issue for me , linking to my branch and saying I have released the task and it is available for others to work on.

Use Case Name:

Basic Flow (back page use is OK):

Participating Actors:

Goal:

Trigger

Precondition:

Postcondition:

Exceptions (back page use is OK):

CMPUT 301 Fall 2016 Final

Name:_____

CCID:_____

UML Sequence Diagrams: [3 marks]

Convert this scenario 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.

On the ComputerCar App I request a car to be delivered to my location. The App confirms my request. 5 minutes later I get notified my robo-car has arrived. I get into the car. I ask the car for its request number, it shows it to me and I verify that is it the same as the request on my phone. I tap the GO button in the car after I am ready. The car departs for my destination. Once we arrive at my destination I get out, gather my belongings and tap the DONE button on the car after I close the door.

CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

Decorator Pattern: [3 Marks]

1. Make a decorator that can log “operate” calls to instances of Operation.

```
public class Logger {
   public static void log(String logMessage) { ... } // logs a message to available logger
   ...
}
public interface Operation {
     Result operate();
}
public class BloodTransfusion implements Operation {
     Result operate() {...}
     ...
}
public class HeartSurgery implements Operation {
     Result operate() {...}
     ...
}
public interface Operator {
      // add and execute an Operation
      public void addOperation(Operation operation);
}
```

2. Add your decortator to an instance of HeartSurgey and add it to an Operator instance

```
// Operator operator;
operator.addOperation(


);
```

CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

Design Patterns: [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) The structure of JSON requests that Elasticsearch wants is different than the structure of your Request object. Too much of your system relies on the Request object the way it is.

2) The sole user has preferences about fonts regarding font family and font size. The rest of the system needs this information and you're sick of passing around a user-preferences object.

3) In your game helicopters and drones fly in a similar way but have distinct differences in response and stability. They share a lot of code, but not all code. There are slight differences in how they fly specific to eithe helicopters or drones.

CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

State Pattern and State Diagram: [3 Marks]

We are tracking people, potential employees, from the point they apply to our company to the point they quit, retire, or get fired. People apply to our company. If we choose to hire them they become juniors, then if they get a promotion they become regular employees, and then if they get promoted again they can become seniors. If a senior does something very wrong we can demote them to regular employee. We can fire juniors, regular employees, and seniors, at which point they are considered fired. Any kind of employee can quit at any time, and seniors can retire at anytime. Employees who quit can be rehired as juniors.

1. Model the state of a person using a UML state diagram.
2. Draw the UML class diagram of a person, and its states, that uses the state pattern that implement the states from your UML state diagram.

CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

Proxy Pattern and Optimization: [3 Marks]

We've got a problem. We're making too many network calls. For every call to

```
public interface Blob {
   // the name of the resource,  locally stored
   public String getNetworkURL();
   // get the Data
   public Object getBlob();
}
// using the proxy pattern optimize this method
void displayBlob(String uri) {
   Blob blog = getBlob(uri);
   if (Math.random() > 0.5) { // half of the time
      displayBlob(blob.getData()); // show the contents of the blob, this is network call
   } else {
      displayURL(blob.getNetworkURL()); // show the blob statistics
   }
}
// by implementing
Blob getBlobs(uris) {
```

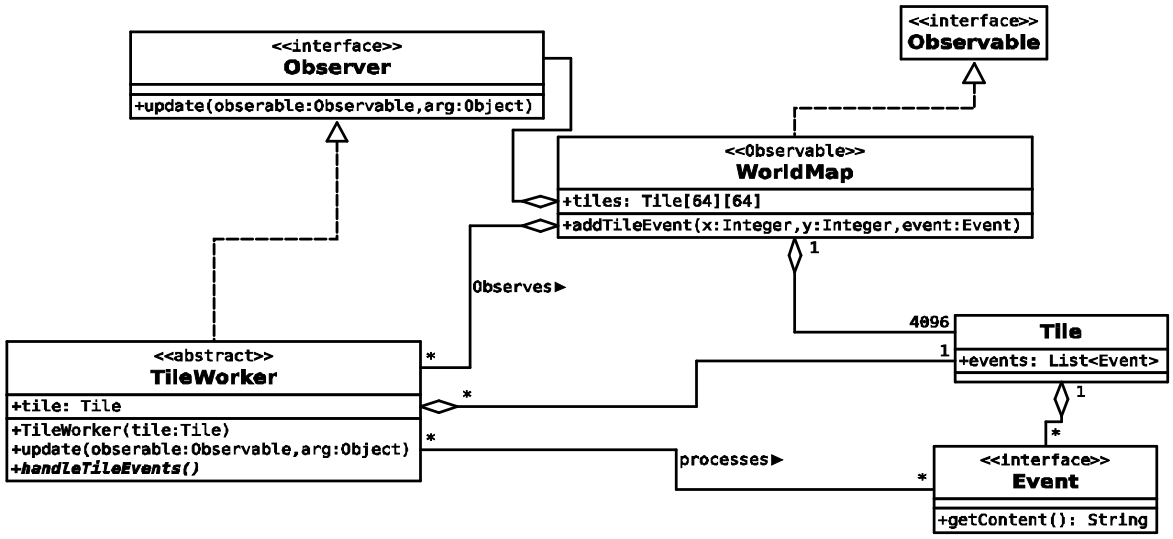
CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

Observer and Optimization: [3 Marks]

We use the observer pattern with WorldMap. We have a map represented by a matrix of tiles. And these tiles accumulate events which we want to process. We have lots of observers, these are TileWorkers, they will consume events accumulated on Tiles. TileWorkers observe the WorldMap. Tiles are shared between TileWorkers and the WorldMap. Unfortunately due to the large number of TileWorkers (1000s), notifyObserver is taking a lot of time. Using the profiler we find that most update calls are hitting irrelevant TileWorkers who have nothing to do with the updated tile. Thus we need to modify and optimize our use of the Observer pattern. **With a UML class diagram show** how you can modify this system such that TileWorkers only get updated for the Tiles they pay attention to.



CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

Testing: [3 Marks]
Write comprehensive testcases for this Stack class that cover **all** equivalence classes.

```
class Stack {
    // returns the object at the top of the stack
    // if no object exists, returns null
    Object pop() { ... }
    // Adds an object to the top of stack
    void push(Object o) { ... }
}
```

CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

Refactoring: [3 Marks]

1. What is the prerequisite that needs to be met before we can refactor?

2. Name and apply an appropriate refactoring for the following, draw a UML class diagram of the resulting code and provide the refactoring name.

```
class Square {
    int width;
    int length;
    Square(int width) {
        this.width = width;
        this.length = width;
    }
    int totalArea() {
        return this.width * this.width;
    }
}
```

```
class Rectangle {
    int width;
    int length;
    Rectangle(int width, int length) {
        this.width = width;
        this.length = length;
    }
    int totalArea() {
        return this.width * this.length;
    }
}
```