CMPUT 301 2016 Fall Term Final Exam
TEST VERSION: Gyarados

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
Factory Method Pattern		3
UML Sequence Diagrams		3
Decorator Pattern		3
Grab Bag		3
State Pattern and State Diagram		3
Patterns and Optimization		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 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: rapid fire, targeted fire, and laser beam sweep. Once all of the legs have been damaged and destroyed by the player the octodroid will be destroyed. Each leg has 300 hit points, and the octo cannons have 600 hit points each. The octo cannons are invincible until the 6 walking legs have been destroyed.

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 ia[4]; // This is owned by a QBA
   private IB ib[5]; // Shared with others
   Collection<IZ> interleave();
}
```

CMPUT 301 Fall 2016 Midterm

Name:_____

CCID:_____

Factory Method Pattern: [3 marks]

Refactor that conditional away using polymorphism. Modify this code to use the factory method pattern. Only show new or changed or moved code. Show the resulting UML class diagram.

```
class BurgerMaker {
    static int DEFAULT=0; // Default BurgerMaker
    static int ANIMALSTYLE=1; // Animal Style BurgerMaker
    int makerType = DEFAULT; // what kind of BurgerMaker
    BurgerMaker(int type) {
       makerType = type;
    }
    Burger prepareBurger() {
       Burger b = new Burger();
       b.add(new Bun());
       b.add(new Lettuce());
       if (makerType == ANIMALSTYLE) {
          b.add(new AnimalStyleCondiments());
          b.add(new AnimalStylePatty());
       } else {
          b.add(new Mustard());
          b.add(new Patty());
       }
       return b;
    }
}
```

CMPUT 301 Fall 2016 Final

Name:_____

CCID:_____

UML Sequence Diagrams: [3 marks]

Convert this self driving car 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.

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. Code a decorator that can log “operate” calls of 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 Surgeon {
      // add and execute an Operation
      public void addOperation(Operation operation);
}
```

2. Add your decortator to an instance of HeartSurgey and add it to the **surgeon** Surgeon instance.

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


);
```

CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

Grab Bag: [3 Marks]

Short answer questions (some of these can be answered with 1 word).

1) What should you do before you optimize any code?

2) What should you have in your codebase before you refactor any code?

3) How does the MV part of MVC affect cohesion?

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. Potential employees (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 any time. 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 implements the states from your UML state diagram.

CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

Patterns and Optimization: [3 Marks]

We've got a problem. We're making too many network calls. Can you reduce the number of calls that hit the network? Consider using a design pattern!

```
// This interface reperesents data blobs that represent static/constant data remotely stored.
// The data they return is different per URL but never changes for 1 URL.
public interface Blob {
   public String getNetworkURL();     // the name of the resource,  locally stored.
   public Object getData();           // get the constant Data for the resource from the network.
}
class BlobDisplayer {
   // using the proxy pattern optimize this method (which you cannot change)
   void maybeDisplayBlob(Blob blob) {
      for (int i = 0 ; i < 10; i++) { // we could call the network from 0 to 10 times!
         if (Math.random() > 0.5) { // half of the time
            displayBlob(blob.getData()); // show the contents of the blob, this is a network call
         }
      }
   }
   void displayBlob(Object o) {...} // displays blob data, do not change.
}
```

Draw a UML Class diagram for your solution. Provide a brief explanation if you need to. **Label** the design patterns that you use. Only provide code if it is needed to explain what you are doing. Do not modify Blob or BlobDisplayer, but subclassing or wrapping or abstracting is possible.

CMPUT 301 Fall 2016 Final;

Name:_____

CCID:_____

Testing: [3 Marks]
Write comprehensive testcases for this Stack class that cover **all** equivalence classes for Stack, Stack.pop and Stack.push . Use JUnit style.

```
class Stack {
    // returns the object at the top of the stack and removes that object.
    // 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. How can continuous integration help refactoring?

2. Name an appropriate refactoring to apply for the following, then draw a UML class diagram of the resulting classes.

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

```
class Rectangle {
    int width;
    int length;
    Rectangle(int width, int length) {
        this.width = width;
        this.length = length;
    }
    int totalArea() {
         return this.width * this.length;
    }
    int perimeter() {
        return 2*width + 2*length;
    }
}
```