Midterm Exam Guide for CSE-40232

The midterm will cover any material from the lectures (meaning that the material is covered by the slides). Please note that if I ask you for Java code I will accept 'close' to real java code. In other words I will look for the main elements of the code rather than absolutely compilable syntax. So a missing semi-colon would be overlooked, whereas incorrect logic would not be overlooked.

The midterm will be on Monday 27th (5.05pm – 6.20pm)
Closed book.

Topics covered:

1. **Design Patterns**:
   For each of the design patterns we've covered so far: Stategy, Observer, Composite, Singleton, State, and Factory Method you should be able to:
   a. Describe the intent of each of the design patterns.
   b. Sketch the UML diagram including major classes, major datastructures, major operations.
   c. Write java code* for various methods in the design patterns.
   d. Understand how such patterns might be implemented in java.
   e. Solve problems by applying these patterns.

2. **SOLID principles**
   Be able to list, describe, and provide examples for each of the SOLID principles (Jin Guo's lecture)

3. **Canonical Form**
   a. What is it?
   b. Why is it important?
   c. Be able to implement Equals(), Hashcode(), and toString().

4. **Design Problem**
   Given a problem – sketch out (and defend) a UML design showing the major classes and major operations you would use to solve it.

5. **Selected Java FX Event Handlers**
   a. Button pressed
   b. On key pressed/typed

6. **Bad Smells**
   a. Given poor quality code – propose a more elegant solution – most likely using one of the design patterns we have studied.
   b. Explain what was wrong with the code.

7. **JavaFX**
   a. Stages, Scenes, and Panes (concepts and code)

8. **JUnit Test Cases**
   a. Given a simple function write a JUnit Test case

See sample exams for style of questions.
Good luck!

A few sample questions:

Q1. Given the following class: (3 points)

```
public class MessageBox {
  int x;
  int y;
  String textMessage;
  Color color;

// Constructor
public MessageBox(){
   // Code here to randomly generate color
}

public void setPosition(Point p){
     x = p.x;
     y = p.y;
}

  Public void setMessage(String message){
     textMessage = message;
  }
```

Override the equals() method so that you can check in a meaningful way that two MessageBox methods are the same.

Q2. Remember the gumball machine? This question requires you to create a state machine for a gate at a tollbooth. A driver can enter money and cause the gate to open. If he/she enters money at the wrong time nothing happens.

a. Draw a state machine diagram for the toll gate. Think about the states needed to operate the gate. Make any reasonable assumptions and feel free to add comments on the side about your assumptions. (4 points)

b. Sketch out a UML Class diagram showing how you might implement a toll gate using the State Design pattern. (3 points)

Q3. Write java code for a simple counter class. Give it two methods for “resetToZero()” and “incrementCounter()”. Make it a **Singleton class**. (5 points)

Q4: Answer the following questions based on the “DoSomething” code below. Assume that methods testCond1, testCond2,….testCond5 define some logic for determining which doSomethingX method to call.

```
 class DoSomething{
    Chooser chooser;
    public DoSomething(Chooser chooser){
        this.chooser = chooser;
        computeScore();
    };
    public int computeScore(){
        if(testCond1(chooser)){
            doSomething1();
        } else if(testCond2(chooser)){
            doSomethingElse2();
        } else if(testCond2(chooser)){
            doSomethingElse3();
        } else if(testCond4(chooser)){
            doSomethingElse4();
        } else if(testCond5(chooser)){
            doSomethingElse5();
        }
    }
 }
```

a. Which OO principle does this code break? Explain your answer(**2 points**)

b. Which design pattern could be used to fix the problem? (**2 points**)

Q5: How would you define “elegant code”? Why is elegance of code important? Put another way, why is it insufficient for a program to function correctly – if the actual code is inelegant? (**3 points**)