

CS4233-A04 Midterm Exam

Name: _____

This exam consists of two parts. If you have read the textbook, attended the lectures, done the homework, and studied the notes you are prepared.

Remember that there is a code of academic honesty at WPI and you are expected to do your own work. Good luck.

Part I: General knowledge (50 points)

This questions in this section are fill-in-the-blank, short answers, and so on.

1. Analysis deals mainly with the **problem** domain, while design deals mainly with the **solution** domain.
2. Objects have three characteristics: state, behavior, and identity. How are each of these represented in Java?
State is represented by variables and computed attributes, behavior by methods, and identity by the object's address.
3. The main purpose of UML is: **communication.**
4. What is the difference between interface realization and subclassing (inheritance)?
When you subclass, you inherit the behavior of all of the methods from the parent. You only have to write code for those methods for which you want to change the behavior. When you realize an interface, you have to write code for all of the methods declared in the interface.
5. What is software architecture?
An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization---these elements and their interfaces, their collaborations, and their composition (The UML Modeling Language User Guide, Addison- Wesley, 1999).
6. The following program counts from 1 to 10 in a separate thread and exits. It is the whole program. What is wrong with it?

```
public class Counter extends Thread
{
    private int counter;
    public Counter( )
    {
        counter = 0;
    }
}
```

CS4233-A04 Midterm Exam

```
public void run()
{
    while (counter < 10) {
        counter++;
        System.out.println("counter = " + counter);
    }
}

public static void main(String[] args)
{
    Counter c = new Counter();
}
}
```

First, the thread is never started. Second, even if it were started, the main loop creates the counter and exits, thereby causing the thread to end (if it were started).

7. In the following code, what might go wrong and how can you correct it?

```
Vector v;
...
public Vector getV()
{
    return v;
}
```

You are returning a vector that can be changed, outside of this class. If you want to control the contents of the vector you should not allow this behavior. You can correct it by returning a deep clone of the vector, thereby insulating the vector, *v*, from any changes from the outside.

8. Explain in a couple of sentences, in general terms, the Lack of Cohesion of Methods (LCOM) metric.
LCOM attempts to measure a class's cohesion by looking at the variables that each method uses and creating equivalence sets. If all of the variables fall into a single set, the class is cohesive. If not, the larger the number of sets, the less cohesive the class.
9. Name two specific advantages of writing tests before you write code?
- You get 100% code coverage.
 - You only write code that you need.
10. The listeners that we have seen in the RemoteMower project are examples of what type of pattern? **The Observer pattern.**
11. What is the difference between a deep clone and a shallow clone?
A deep clone performs a deep clone on all variables that are, themselves, objects. A shallow clone just copies the reference to variables that are objects.

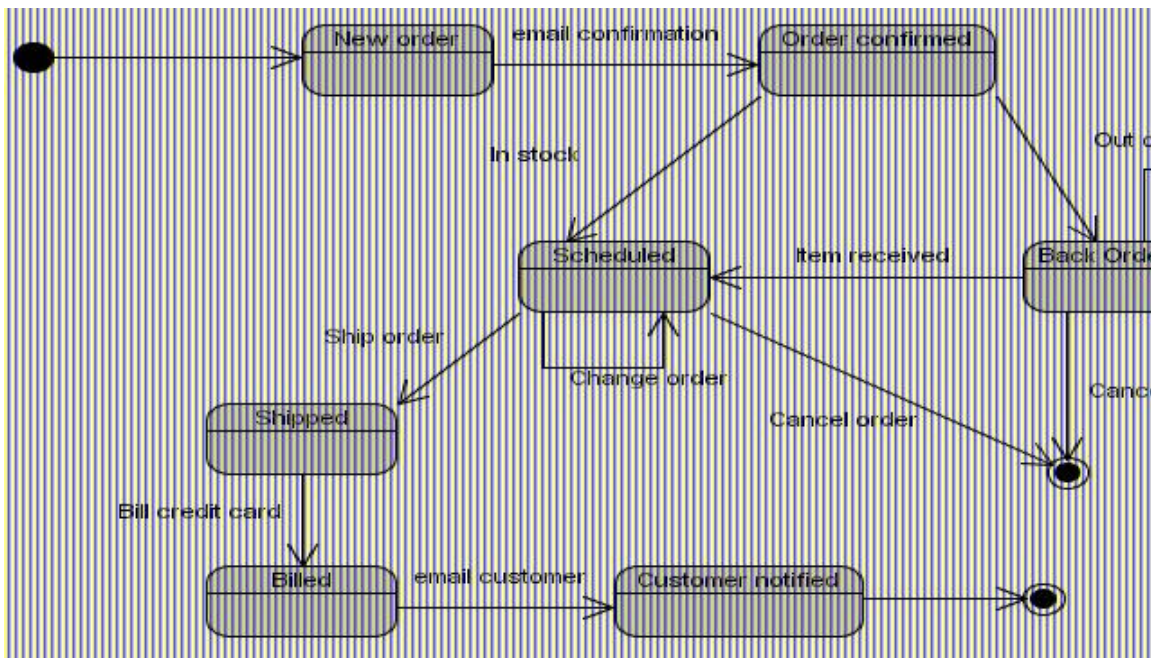
12. Why, when you lower coupling in a class, do you usually lower cohesion as well, and vice-versa?

In order to make a class more cohesive, you might break the class up into multiple classes which can cause other classes to now be associated with the new set of classes rather than one. Also, you can improve cohesion by making a class delegate behavior to other classes. In so doing, the class becomes more highly coupled with the delegate classes. The same argument applies when raising the coupling.

The inverse argument can be made when you lower coupling or cohesion.

Part II (50 points)

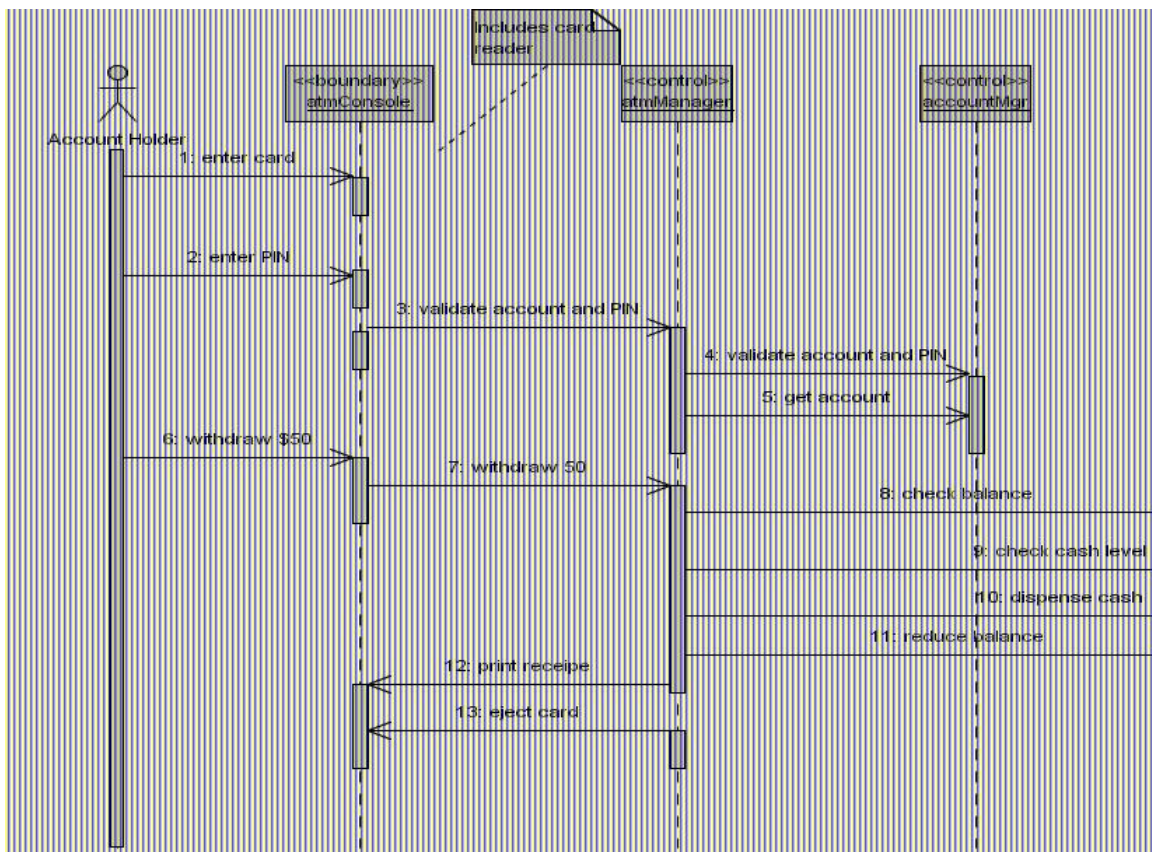
1. Consider processing an order at Amazon.com. When you place an order, Amazon must notify you that your order has been received, schedule the order for shipping, if the item is not available it must be back-ordered, prepare the order for shipping, ship the order, notify you that the order has been shipped (or back-ordered), and bill your credit card. In addition, you can cancel the order, or modify the information on it before it has shipped. Draw a UML state diagram showing the Order class's states and the events that cause the order to change from one state to the next.



CS4233-A04 Midterm Exam

2. Draw a sequence diagram for an analysis model for the following use case scenario. Indicate whether each object is a boundary, control, or entity object. Not every step in the following description is appropriate for the diagram and you will need to identify objects that are not explicitly stated in the scenario. **Draw the diagram neatly on the back of this page.**

The account holder inserts her bank card into the ATM and enters her PIN number. The system verifies that the account is valid and the PIN is the correct one for the card. The account holder selects “Withdraw \$50.” The ATM verifies that there is enough money in the checking account, that there is enough money in the ATM, deducts \$50 from the checking account, and dispenses the money to the account holder. The system prints a receipt and ejects the bank card. The account holder takes the receipt and the card and the use case ends.



CS4233-A04 Midterm Exam

The following class is not very cohesive. Using the principles of the LCOM metric, refactor the code to make a set of cohesive classes. Do the work on the back of this page.

```
public class TestClass
{
    int a, b, c, d, e, f, g, h;

    void method1(int x, int y)
    {
        a = x;
        b = y;
        c = x + y;
    }

    int method2()
    {
        return a * b + e;
    }

    int method3()
    {
        g = d + g;
    }

    void method4(int x)
    {
        h *= x;
    }

    void method5(int x)
    {
        f = x * x;
    }

    int method6()
    {
        return f;
    }

    int method7()
    {
        return g - h;
    }
}
```

Class1: abce, method1, method2

Class2: dgh, method3, method4, method7

Class3: f, method5, method6