CS401 MPP Midterm - *Solutions*

Name:_____ StudentId:_____

Part I, 1-7 (21)	Part II, 1-7 (35)	Part III, 1 (10)	Part III, 2 (10)	Part IV SCI (3)

Part I: Multiple Choice (3 Points Each)

1. In the following code, which of the following is correct regarding the relationship between Employer and Gardener? Circle one letter.

   (A) There is a dependency from Employer to Gardener

   B. There is a one-way association from Employer to Gardener

   C. There is a two-way association between Employer and Gardener

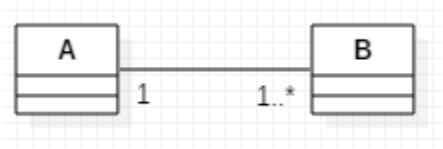
   D. Not possible to determine from the code shown.

```
public class Employer {
    public void employ() {
        Gardener gardener = new Gardener();
        gardener.garden();
    }
}
```

```
public class Gardener {
    public void garden() {
        //do gardening
    }
}
```

Grading:	
B,C,D	3 pts
A,B,C, D,E	0 pts
Answers include any two of the correct answers	2 pts
Answers include any one of the correct answers	1 pt

2. Consider the following class diagram:



Which of the following statements is (are) correct? Circle all that are correct.

A. Each instance of the class B contains a list of instances of A.

B. Each instance of the class A contains a list of instances of B.

C. A is a property of B.

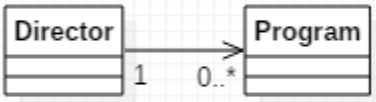
D. If an instance of A has been created, at least one instance of B has also been created.

E. After an instance of A has finished creating instances of B (either inside its constructor or inside some other constructing method), the instances become null automatically after A's constructor or A's constructing method returns.

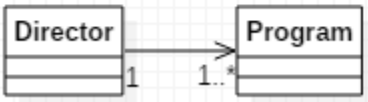
3. Which of the following UML diagrams correctly models the relationship between Director and Program? The code for Director and Program is shown below.

A.

B.



C.




```java
public class Program {
    private String programId;
    public Program(String programId) {
        this.programId = programId;
    }
}
```

```java
public class Director {
    private String name;
    private List<Program> programs = new ArrayList<>();
    public Director(String name, String programId) {
        this.name = name;
        addProgram(new Program(programId));
    }
    public void addProgram(Program p) {
        programs.add(p);
    }
}
```

4. When the following code is executed, it prints the areas of each of the geometric figures that have been initialized in the main method.

```
public class Circle {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public double computeArea() {
        return Math.PI * radius * radius;
    }
}

public class Rectangle {
    private double x, y;
    public Rectangle(double side1, double side2) {
        x = side1;
        y = side2;
    }
    public double computeArea() {
        return x * y;
    }
}

public class Main {
    public static void main(String[] args) {
        List<Object> list = new ArrayList<>();
        list.add(new Rectangle(2,5));
        list.add(new Circle(2.5));
        list.add(new Circle(1.8));
        list.add(new Rectangle(7,2));
        Main m = new Main();
        m.computeAllAreas(list);
    }
    public void computeAllAreas(List<Object> objects) {
        for(Object ob: objects) {
            if(ob instanceof Rectangle) {
                Rectangle r = (Rectangle)ob;
                System.out.println(r.computeArea());
            } else if(ob instanceof Circle) {
                Circle c = (Circle)ob;
                System.out.println(c.computeArea());
            }
        }
    }
}
```

Although the code gives correct outputs, it violates an OO principle. Which of the following OO principles is violated by the structure of this code? Circle only one letter.

A. Liskov Substitution Principle

B. Ripple Effect

C. Program to the Interface

(D.) Open-Closed Principle

5. Determine which of the following classes is NOT immutable. Circle the letters for all that apply.

Grading:	
A, B A,B,C	3 pts
A,C or B, C or A only or B only or C only	1.5 pts

Correct Answer: A, B (and C).

Explanation:
A - instance variables not final, class not final
B - getHireDate returns a mutable object
C – (accidentally correct) deptCode not final

[Note on C: Although a list is returned by getEmployeeIds, it is returned as an unmodifiable list. Typo – forgot to put "final" in deptCode, so C can be a correct answer too]

A.

```
public class Account {
    private double balance;
    private String acctNumber;
    public Account(double bal, String num) {
        balance = bal;
        acctNumber = num;
    }
    public double getBalance() {
        return balance;
    }
    public String getAcctNumber() {
        return acctNumber;
    }
}
```

B.

```
final public class Employee {
    final private String name;
    final private Calendar hireDate;
    public Employee(String name, int year, int month, int day) {
        this.name = name;
        this.hireDate = new GregorianCalendar(year, month, day);
    }
    public String getName() {
        return name;
    }
    public Calendar getHireDate() {
        return hireDate;
    }
}
```

C.

```
final public class Department {
    final private List<String> employeeIds;
    final String departmentCode;
    public Department(List<String> empIds, String code) {
        employeeIds = empIds;
        departmentCode = code;
    }
    public List<String> getEmployeeIds() {
        return Collections.unmodifiableList(employeeIds);
    }
    public String getDepartmentCode() {
        return departmentCode;
    }
}
```

Grading:	
D	3 pts
A,B or B, C or A, C or	2 pts
A only or B only or C only	1.5 pts
E	0 pts

6. The difficulties with the `ExtendedHashSet` example, discussed in class, exemplify the following (circle *the best answer*). (Code shown below.)

A. The Ripple Effect

B. The incompatibility between the OO principles of Encapsulation and Inheritance

C. A subclass may break because of a change in its superclass

D. All of the above.

E. None of the above.

```
blic class ExtendedHashSet<T> extends HashSet<T> {
  //The number of attempted element insertions since its creation --
  //this value will not be modified when elements are removed
  private int addCount = 0;

  public ExtendedHashSet() {}

  @Override
  public boolean add(T a) {
      addCount++;
      return super.add(a);
  }
  @Override
  public boolean addAll(Collection<? extends T> c) {
      //addCount += c.size();
      return super.addAll(c);
  }
  public int getAddCount() {
      return addCount;
  }
}
```

7. What happens when the following code is compiled/run? Select only *one*.

```
public class Main {
    public static void main(String[] args) {
        MySuperSuper mss = new MyClass();
        mss.myMethod();
    }
}
public class MyClass extends MySuper {
    public void myMethod() {
        super.myMethod();
        System.out.print("Goodbye");
    }
}
public class MySuper extends MySuperSuper {
    public void myMethod() {
        System.out.println("Hello ");
        super.myMethod();
    }
}
public class MySuperSuper {
    public void myMethod() {
        System.out.println("Super super");
    }
}
```

A. Compiler error

B. Runtime exception is thrown

C. The following is printed to the console:

```
Hello
Super super
Goodbye
```

D. The following is printed to the console:

```
Super super
```

E. The following is printed to the console:

```
Super super
Hello
Goodbye
```

Part II: Short Answer (5 Points Each)

Grading:

Take off one point for each of the following elements that are missing:

In MainStage:
- controller instance vble
- create new Controller in MainStage constructor
- set instance of MainStage in new Controller instance

In Controller:
- Controller accepts instance of MainStage (either in constructor or by a setter method)

1. Portions of three of the classes in a particular Java application whose UI is coded using JavaFX are shown below. The designer is planning to implement a 1:1 two-way association between the `MainStage` class and the `Controller` class. Write code that will guarantee such a relationship between these classes when the `MainStage` class is instantiated. Write your code directly in the classes provided below. (The `Controller` class appears on the following page.)

```
public class Main extends Application {

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        new MainStage(primaryStage);
    }

}
```

```
public class MainStage {
    private Stage primaryStage;
    private Controller controller;

    public MainStage(Stage primaryStage) {
        this.primaryStage = primaryStage;
        controller = new Controller(this);

    }
}
```

```
public class Controller {
    private MainStage mainStage;
    Controller(MainStage ms) {
        mainStage = ms;
    }
}
```

2. A rectangle can be specified by specifying two sides, but it can also be specified by specifying one side and a diagonal.

   A. The following code attempts to implement a Rectangle class and provide support for the two ways of constructing a Rectangle. The code does not compile. What is the compiler error? (Write your answer below.)

```
public class Rectangle {
    double side1, side2, diagonal;
    public Rectangle(double s1, double s2) {
        this.side1 = side1;
        this.side2 = side2;
        diagonal = Math.sqrt(side1 * side1 + side2 * side2);
    }
    public Rectangle(double s1, double diagonal) {
        this.side1 = side1;
        this.diagonal = diagonal;
        side2 = Math.sqrt(diagonal * diagonal - side1 * side1);
    }
    public double computeArea() {
        return side1 * side2;
    }
}
```

Your Explanation:

Solution: The Java compiler does not allow two constructors in a class that have the same signature.

Grading:

Part A is worth 1 point. No partial credit should be awarded – the student must have the answer shown below to get credit

Part B is worth 4 points. Take off points if any of the following elements are missing:
- no-argument private constructor
- 2 public static factory methods that create instances of Rectangle in the different ways (using 2 sides or a side and a diagonal); these methods must each accept two arguments of type double
- each factory method must create a new instance using the contrstructor and then populate with input arguments, then return the instance to caller

B. In the space provided below, rewrite the code for Rectangle (from Part A) so that it supports the two ways of constructing a rectangle. Use a technique described in the course.

```
public class Rectangle {
    double side1, side2, diagonal;
    private Rectangle() {}
    public static Rectangle createRectangleBySides(double s1, double s2) {
        Rectangle r = new Rectangle();
        r.side1 = s1;
        r.side2 = s2;
        r.diagonal = Math.sqrt(r.side1 * r.side1 + r.side2 * r.side2);
        return r;
    }
    public static Rectangle createRectangleBySideDiag(double s1, double diagonal) {
        Rectangle r = new Rectangle();
        r.side1 = s1;
        r.diagonal = diagonal;
        r.diagonal = Math.sqrt(r.diagonal * r.diagonal - r.side1 * r.side1);
        return r;
    }
    public double computeArea() {
        return side1 * side2;
    }
}
```

3. Name two ways of guaranteeing that no user of your class MyClass can create a subclass of MyClass.

Grading:

Give 2.5 points for each of the two parts of the solution

Solution:

a. Declare MyClass to be final

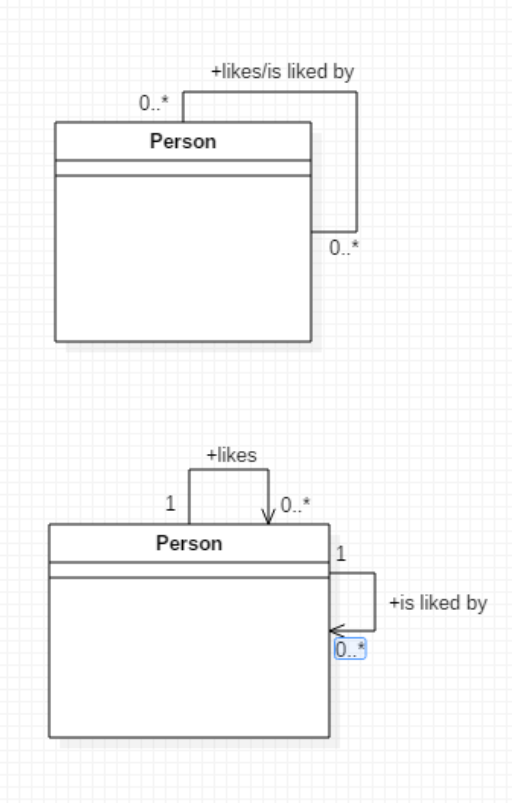
b. **Declare the MyClass constructor to be private and provide a factory method to provide instances of the class.**

4. A Person may *like* other Persons, and may also be *liked by* other Persons. Draw a class diagram showing these relationships. Use one or more associations; each should have a name and appropriate multiplicities. You must indicate clearly whether associations are 1-way or 2-way.

Grading: Look for the student solution that matches most closely one of the correct solutions shown here

Take of 1 point for each element that is not correct; look for:
- association name
- proper multiplicities at each end of an association

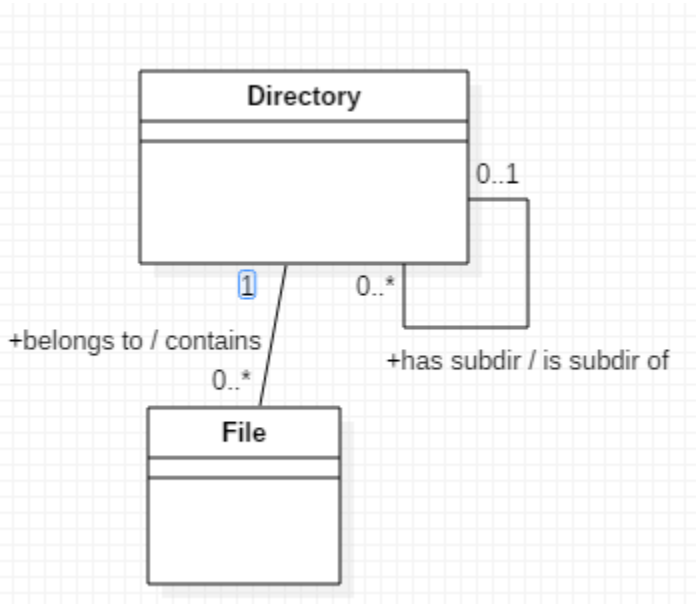
Solution: Both of the following are correct:



5. In a Windows file system, a directory may contain files and other directories. Represent these concepts with classes Directory and File in a class diagram and show association relationships. You do not need to specify attributes or operations in these classes. Associations should be provided with multiplicities and names. NOTE: If A is a subdirectory of B and B is a subdirectory of C, it is not true that A is a subdirectory of C.

Grading: The solution should either look like the diagram shown here or either of the two-way associations can be replaced with 2 one-way associations

Take of 1 point for each element that is not correct; look for:
- association name, for each direction
- proper multiplicities at each end of each association



About the solution: Associations are two-way:

a. **A directory has 0 or 1 parent directory and may have many subdirectories**

b. **A file belongs to exactly one directory; a directory has zero or more files in it.**

6. Explain the difference between *analysis* and *design* in the software development lifecycle.

Grading: Give 5 points if they have made a clear distinction between *what* (analysis) and *how* (design)

If they have not made this distinction, then if there is a good discussion explaining that requirements gathering is done in analysis and building a solution is done in design, give 4 points (or less if discussion is unclear)

Solution: Analysis is concerned with understanding and modeling requirements. It is concerned with determining *what* is needed.

Design is concerned with building a software solution in accordance with requirements. It is concerned with *how* to construct the software objects so they fit together well, in accordance with OO design principles.

7. What is the Evolving API Problem?

Grading: The solution given by the student should be very close to the answer given here. Give 5 points if it is very close, 4 points if good but not quite the same, etc.

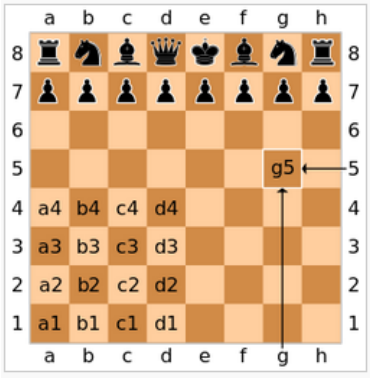
Solution: The Evolving API Problem is the problem that if an interface in a system is expanded to include new operations, then existing classes that have already implemented the interface will break because the new interface operations will not have implementations.

Part III: Design (10 Points Each)

The game of chess in the West is a two-person game involving a checkered 8 x 8 chessboard containing 64 squares, with alternating colors black and white, and a set of 32 pieces – 16 black pieces for one player, 16 white pieces for the other player. The starting position of a game of chess is shown below.

The game proceeds as follows: The white player (called White) moves first by making a legal move with one of his white pieces. The black player (called Black) then makes a legal move with one of his black pieces. Each piece is allowed to move in certain ways, specified by the rules of chess. According to the rules, in certain circumstances, a player may also capture an opponent's piece. Players continue to take turns making moves until a configuration of the game arises that indicates either a victory for one of the players or a draw (neither player wins).

You have been asked to create a design for an online chess game. The game will display a board with pieces, which will look similar to the image above. There will be two online players, and each player will be represented by a Player object in the game. A Player object stores the user's name and whether he is White or Black. It also stores the Player's record of wins, losses, and draws (this record is called the *won-lost record*). The squares on the board are specified using the following grid:



Each piece in the game is represented by a Piece object, which stores its position (using the naming system described above), and which also has a *draw* operation that allows it to draw itself on the displayed Board at its specified position.

During the game, the system maintains a Grid (which is similar to the image shown above), that keeps track of the arrangement and locations on the board of all the pieces in the game.

Each online player uses the naming system described above to submit his/her next move. For example, if WHITE has a piece at position **g5** and he wishes to move it forward one square, he would submit to the system the message "**g5 to g6**."

When a player submits a play to the system (like "**g5 to g6**"), the system will then pass the current Grid and the requested move to a Verifier to check that the move is legal. If the Verifier accepts the move, the system will update the Grid with the new position, and it will locate the Piece to be moved by reading the first position specified by the user (in this example, position **g5**) and locating this position on the Grid. It will then update Piece's position in accordance with the user's specification (in this example, the Piece's position would be changed to **g6**). The system will then pass the updated Grid to the Board and ask the Board to redraw itself. As part of the process of re-drawing itself, the Board will ask the Pieces to re-draw themselves. (If a Piece is captured, the captured Piece will be told to disappear.)

After the system has completed the move, it passes the Grid to the Verifier, which will determine if the game is over, and declare a winner (if there is one). If the game is declared finished, the system updates both Players' won-lost records. A "Game Over" message is then displayed.

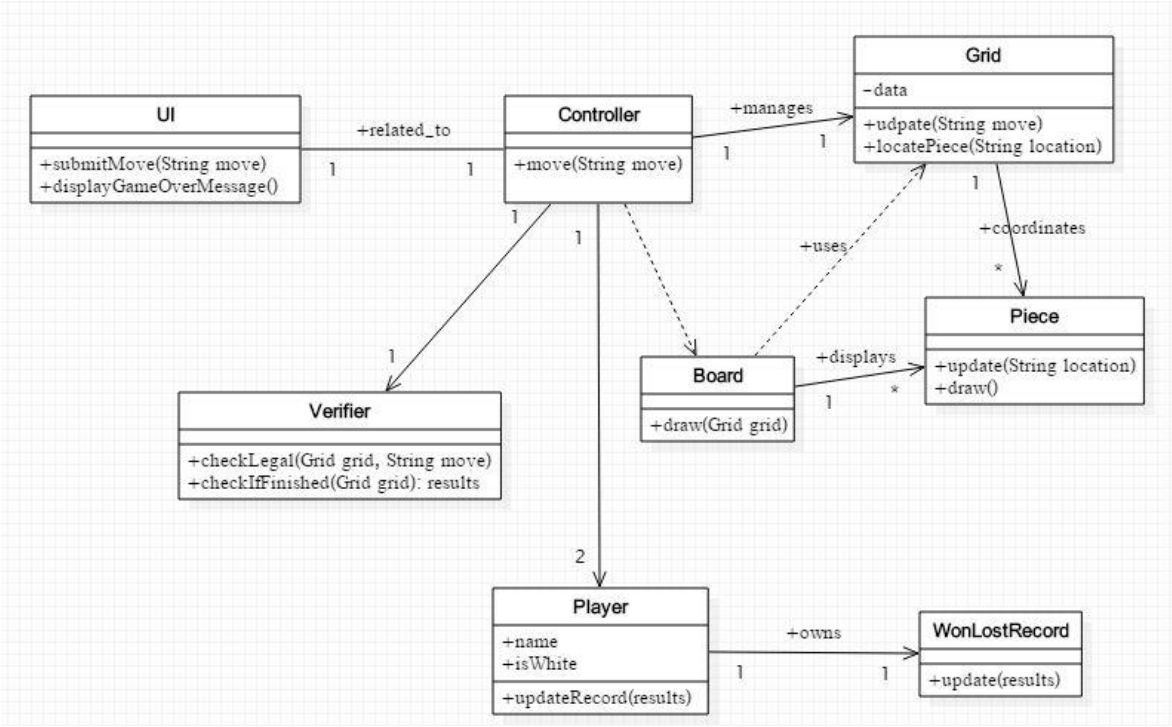
The two parts of this problem are shown on the following pages.

<u>Notes about the problem</u>:

A. There is no need to model a data access layer or a rules engine in this problem (so don't do it!).

B. Your sequence diagram should show an actor, a UI class, and a Controller class. Include the UI and Controller classes in your class diagram also.

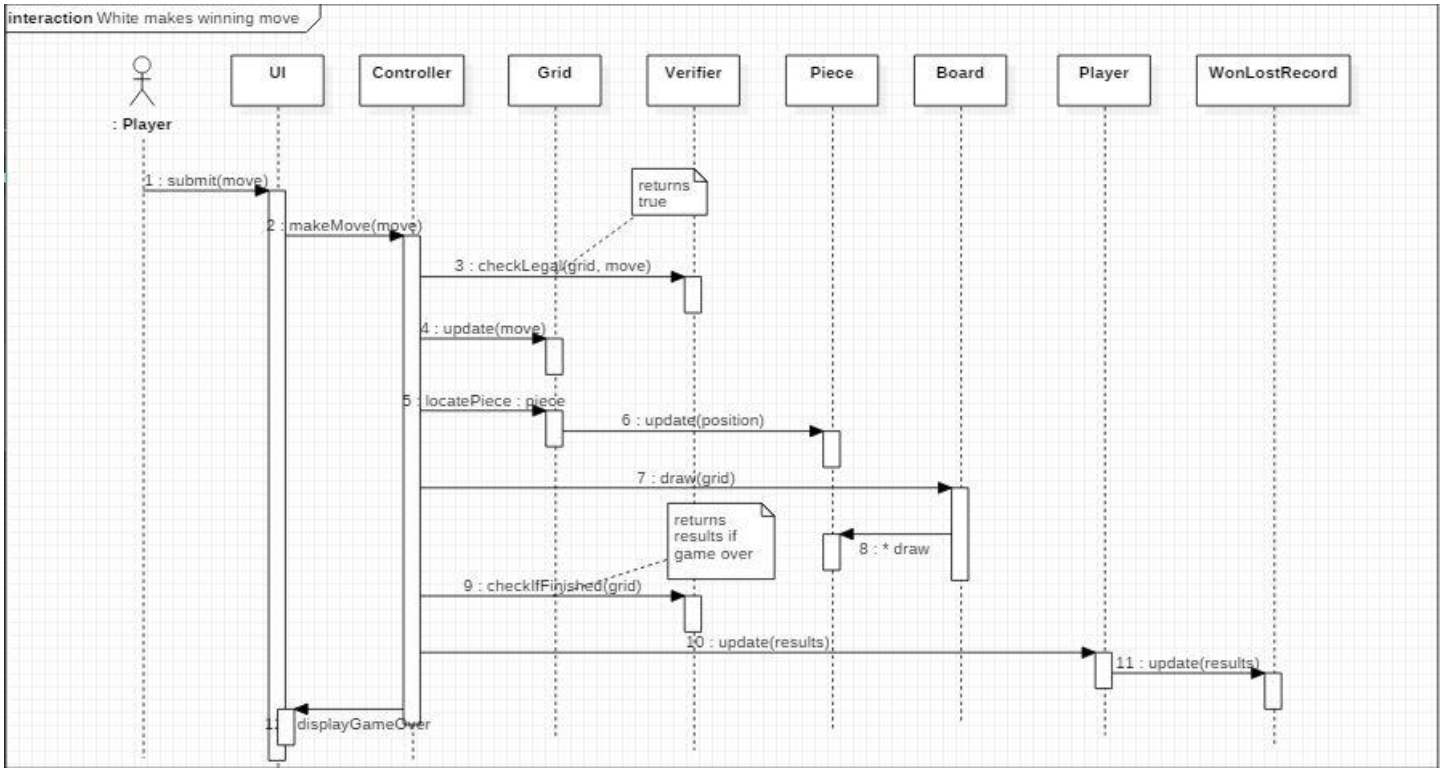
1. Create a class diagram for this system. Show attributes, operations, and associations (include multiplicities; also include names for associations *if* it is useful to do so).



Grading: The problem is worth 10 points. Check the following (and take off points accordingly)

1. All classes shown here are in the student solution (possibly with different names)
2. Most of the attributes are present
3. Check multiplicities
4. Operations should be similar to those shown here
5. No penalty for using association instead of dependency

2. Create a sequence diagram for the following scenario: White makes a move that results in a victory for White. Assume that White's move is legal and that the Verifier determines that White is the winner after this move is made. Be sure to show activation bars and message numbering. Show parameters in your messages if applicable.



Grading: The problem is worth 10 points. Check the following (and take off points accordingly)

1. Check activation bars
2. Check parameters for operations are present
3. Check message numbering
4. Check logic of diagram
5. Check all classes are shown
6. Check that actor, UI, and Controller are present

Part IV: SCI (3 Points)

Write one or two paragraphs relating a point from the course to a principle from SCI.