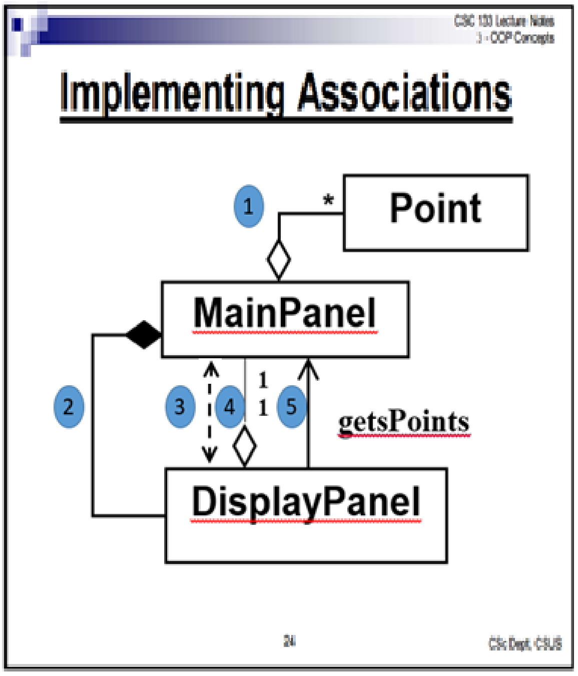
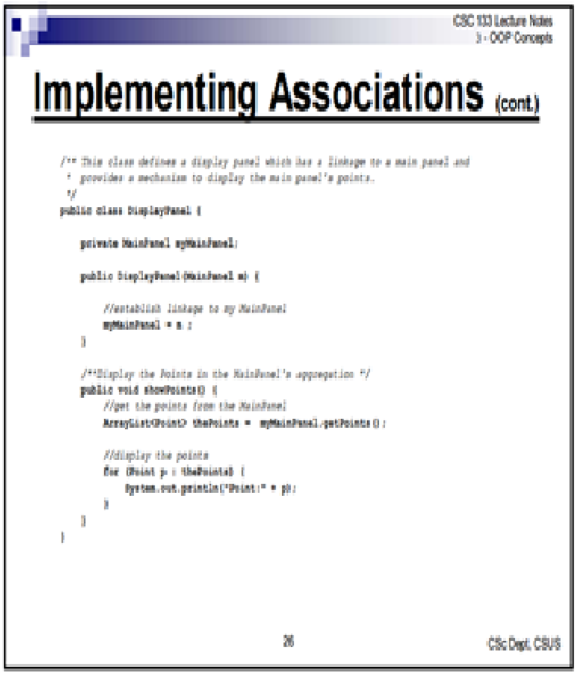
CSC-133 (Spring 2019)
Attendance Quiz 2 – OOP Concepts and UML Class Diagram

Student Name:______________________

Question 1: Two different programming teams have implemented a class named **Rectangle**. One team provided accessors to get and set the location (origin), width, and height of a rectangle, while the other team chose to make the origin, width, and height fields public so that they can simply be directly accessed (read and/or changed). The second team argues that if you have accessors which allow you to both get and set all the values in the rectangle, there is no difference in having the fields public. Explain why the second team does not know what they are talking about. Be specific; give an example of how their approach can produce a software system that fails. **(10 points).**

Question 2: UML is not just about pretty pictures. If used correctly, UML precisely conveys how code should be implemented from diagrams. If precisely interpreted, the implemented code will correctly reflect the intent of the designer. Please review the following slides. In the class diagram in slide 24, given the labels 1, 2, 3, 4, and 5 please fill out the required information in the table beneath. **BE SURE TO JUSTIFY YOUR ANSWER. (10 points).**

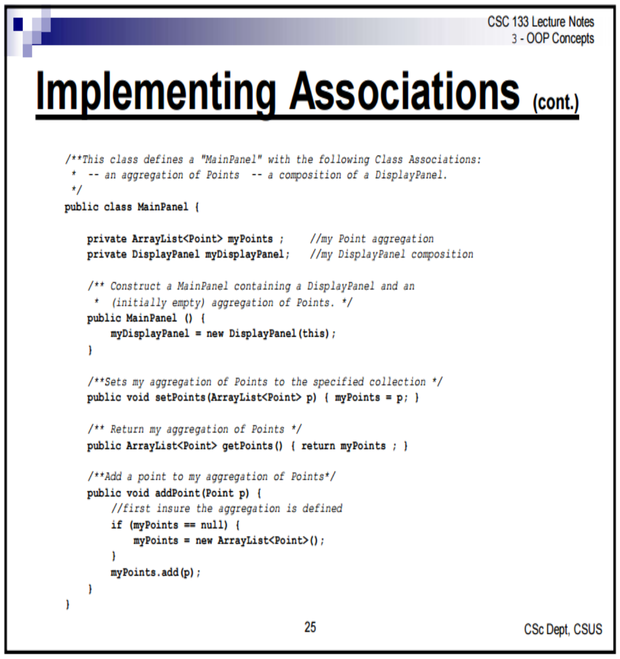



CSC 133 Lecture Notes
3 - OOP Concepts

Implementing Associations (cont.)

```
/** This class defines a display panel which has a linkage to a main panel and
 *  provides a mechanism to display the main panel's points.
 */
public class DisplayPanel {

    private MainPanel myMainPanel;

    public DisplayPanel(MainPanel m) {

        //establish linkage to my MainPanel
        myMainPanel = m ;
    }

    /**Display the Points in the MainPanel's aggregation */
    public void showPoints() {
        //get the points from the MainPanel
        ArrayList<Point> thePoints =  myMainPanel.getPoints() ;

        //display the points
        for (Point p : thePoints) {
            System.out.println("Point:" + p);
        }
    }
}
```

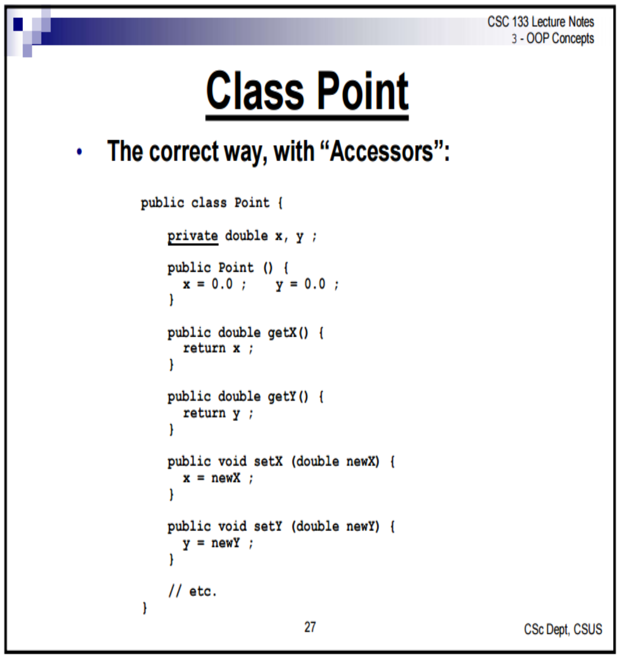
26 CSc Dept, CSUS


CSC 133 Lecture Notes
3 - OOP Concepts

Implementing Associations (cont.)

```
/**This class defines a "MainPanel" with the following Class Associations:
 *  -- an aggregation of Points  -- a composition of a DisplayPanel.
 */
public class MainPanel {

    private ArrayList<Point> myPoints ;    //my Point aggregation
    private DisplayPanel myDisplayPanel;   //my DisplayPanel composition

    /** Construct a MainPanel containing a DisplayPanel and an
     *  (initially empty) aggregation of Points. */
    public MainPanel () {
        myDisplayPanel = new DisplayPanel(this);
    }

    /**Sets my aggregation of Points to the specified collection */
    public void setPoints(ArrayList<Point> p) { myPoints = p; }

    /** Return my aggregation of Points */
    public ArrayList<Point> getPoints() { return myPoints ; }

    /**Add a point to my aggregation of Points*/
    public void addPoint(Point p) {
        //first insure the aggregation is defined
        if (myPoints == null) {
            myPoints = new ArrayList<Point>();
        }
        myPoints.add(p);
    }
}
```

25 CSc Dept, CSUS


CSC 133 Lecture Notes
3 - OOP Concepts

Class Point

- The correct way, with "Accessors":

```
public class Point {

    private double x, y ;

    public Point () {
      x = 0.0 ;    y = 0.0 ;
    }

    public double getX() {
      return x ;
    }

    public double getY() {
      return y ;
    }

    public void setX (double newX) {
      x = newX ;
    }

    public void setY (double newY) {
      y = newY ;
    }

    // etc.
}
```

27 CSc Dept, CSUS

Label Number	Name the association	Justifications (Please use the “Recap 1” Table (Slide # 33) for reference)	Specify Java Class(es) Name and Line of codes
1			
2			
3			
4			
5			