

CSC-133 (Fall 2018)

Attendance Quiz 1 – OOP Concepts and UML Class Diagram

Student Name: Warren Quattrocchi

Question 1: Define encapsulation, abstraction. Explain how these two concepts relate to object-oriented programming (OOP) **(10 points)**.

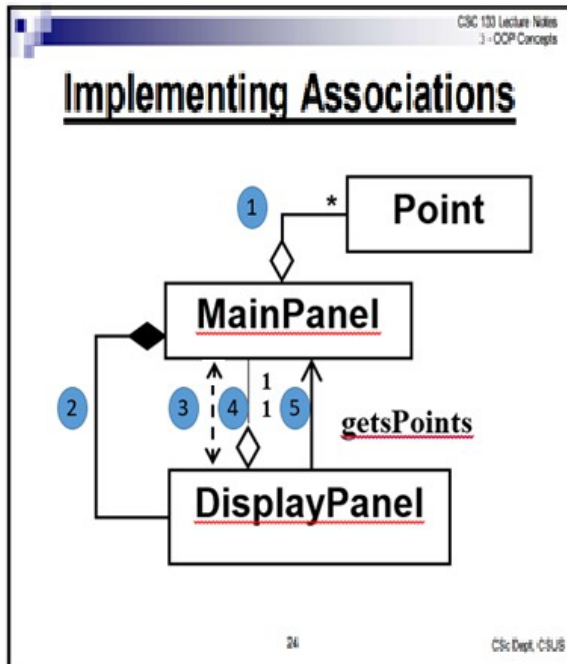
Encapsulation is a way to keep relevant data and methods together thorough “bundling” and “information hiding”

- Encapsulation is important in object-oriented programming as it helps keep related objects together and from unrelated objects

Abstraction is the process of removing characteristics from something in order to reduce it to a set of essential characteristics, and identify these objects by essential characteristics.

- Abstraction is important in object-oriented programming as it can increase the amount of re-usable code and greatly simplify systems.

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 mp) {

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

    /**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 for reference)	Specify Java Class(es) Name and Line of codes
①	Aggregation	A MainPanel has Points Points can exist without MainPanel	In class MainPanel: private ArrayList<Point> myPoints;
②	Composition	A MainPanel is made up of a DisplayPanel The DisplayPanel cannot exist without MainPanel	In class MainPanel: private DisplayPanel myDisplayPanel;
③	Dependency	MainPanel contains parameters of type DisplayPanel DisplayPanel contains parameters of type MainPanel	In class MainPanel: public MainPanel(){ myDisplayPanel = new DisplayPanel(this); } In class DisplayPanel: public DisplayPanel(MainPanel m) { myMainPanel = m; }
④	Aggregation	A DisplayPanel has a MainPanel	In class DisplayPanel: private MainPanel myMainPanel;
5	Association	DisplayPanel sends a message to MainPanel	In class DisplayPanel: ArrayList<Point> thePoints = myMainPanel.getPoints();