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).

Answer:

The second team is wrong in saying that a public variable is the same as accessing a private variable through setter/getter methods. When using a setter method, you can have input validation to ensure that only “correct” data is put into the variables. In addition, when incorrect data is inputted into a variable using a setter method, the method can prevent it and inform the user to try again. It is also difficult to track down where a variable was changed if setter methods are not used. Furthermore, with public variables, it is difficult to ensure that it is not being used outside of where you want it to be used. For example, one member of a team can change a public variable in a completely different part of the code, and the rest of the team would not know. Private variables keep the usage of the variable inside of the class.

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).

Answer:

Label Number	Name the association	Justifications	Specify Java Class(es) Name and Lines of Code
1	Aggregation	MainPanel is made up of instances of Point objects. In this case, MainPanel is the whole and Point is the part.	private ArrayList<Point> myPoints
2	Composition	MainPanel is made up of DisplayPanel and if MainPanel is destroyed, then DisplayPanel is destroyed.	Private DisplayPanel myDisplayPanel

3	Dependency	MainPanel and DisplayPanel are dependencies of each other because they each contain an instance of each other.	Public MainPanel() { myDisplayPanel = new DisplayPanel(this);
4	Aggregation	DisplayPanel is made up of instances of MainPanels	Public MainPanel() { myDisplayPanel = new DisplayPanel(this);
5	Association	DisplayPanel calls methods from MainPanels	myMainPanel.getpoints()