08.03 Virtual Lecture Notes (Part 2)

A class diagram is a graphic organizer that visually depicts the key properties of a class and the relationship with other classes. A generic example is shown below.

```
Class
------------
Variables
-  private
+ public

------------
Methods
- private
+ public
# protected
```

A class diagram is a simple way to graphically depict the components of a class. It consists of a rectangle with three compartments separated by two horizontal lines.

The top compartment indicates the name of the class, attributes are listed in the middle compartment, and the class's methods are shown in the lower compartment.

Attributes and methods may be preceded by a symbol to indicate their scope: public (+), private (-), or protected (#). Components may be annotated and grouped into categories or simply listed.

At a glance, this diagram reveals two important aspects of a class: attributes and behaviors.

Where does UML fit into the programming process? Pseudocode helps to abstract a general solution, but is short on details. Once you have an idea of what you want to do, a class diagram helps identify and name specific attributes and behaviors: the variables and the methods. Flowcharting is more specific, but should be limited to small segments of code if you need to trace the flow of control through decisions or loops. Never try to flowchart an entire object-oriented program!

The following class diagram corresponds to the last demo program in this lesson. What does it tell you?

```
                 ShapesV3
--------------------------------------------
           << In main method >>
+ int side1
+ int side2
+ double triArea
+ double hypoteneuse

--------------------------------------------
            << Constructor >>
+ ShapesV3()

              << Methods >>
+ double calcTriArea(int s1, int s2)
+ double calcHypoteneuse(int s1, int s2)
```

This class diagram tells you the following:

1. The name of the class is **`ShapesV3`**.
2. There are four public variables in the **`main()`** method; their names and types are listed.
3. There is one default constructor, named **`ShapesV3()`**.
4. There are two public methods; their names, types, and parameter lists are provided.

Start making class diagrams *before* you start coding, you will be more organized and less confused. Armed with the class diagram shown above, it would not be difficult to write the ShapesV3 class.

```
public class ShapesV3
{
  //default constructor
  ShapesV3()
  {
  }

  //calculate area of a triangle
  public double calcTriArea(int s1, int s2)
  {
      return s1 * s2 * .5;
  }

  //calculate the hypoteneuse of a right triangle
  public double calcHypoteneuse(int s1, int s2)
  {
      return Math.sqrt(Math.pow(s1, 2) + Math.pow(s2, 2));
  }

  //main method
  public static void main(String[] args)
  {
     //declaration of variables
     int side1, side2;
     double triArea, hypoteneuse;

     //initialization of variables
     side1 = 10; side2 = 5;
     triArea = 0; hypoteneuse = 0;

     ShapesV3 shapes = new ShapesV3();

     //call methods
     triArea = shapes.calcTriArea(side1, side2);
     hypoteneuse = shapes.calcHypoteneuse(side1, side2);

     //print results
     System.out.printf(" Triangle Area = %8.2f%n", triArea);
     System.out.printf("   Hypoteneuse = %8.2f%n", hypoteneuse);
  }
}
```