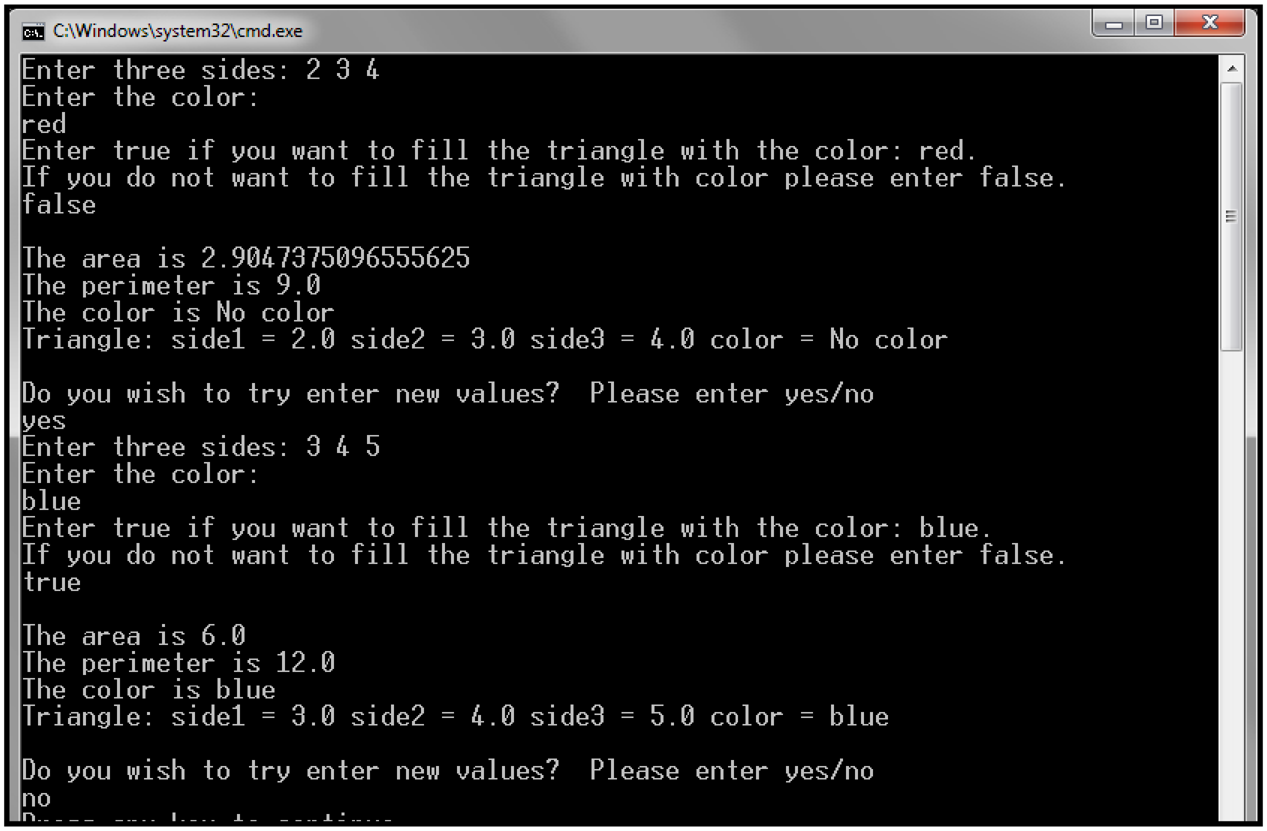
CIT 249
Chapter 11
Lab 1

In this lab we will cover #1 under Programming Exercises on page 442. When the program is run it will display as:



C:\Windows\system32\cmd.exe

```
Enter three sides: 2 3 4
Enter the color:
red
Enter true if you want to fill the triangle with the color: red.
If you do not want to fill the triangle with color please enter false.
false

The area is 2.9047375096555625
The perimeter is 9.0
The color is No color
Triangle: side1 = 2.0 side2 = 3.0 side3 = 4.0 color = No color

Do you wish to try enter new values?  Please enter yes/no
yes
Enter three sides: 3 4 5
Enter the color:
blue
Enter true if you want to fill the triangle with the color: blue.
If you do not want to fill the triangle with color please enter false.
true

The area is 6.0
The perimeter is 12.0
The color is blue
Triangle: side1 = 3.0 side2 = 4.0 side3 = 5.0 color = blue

Do you wish to try enter new values?  Please enter yes/no
no
Press any key to continue . . .
```

If the user types in "yes", the code will be repeated.

Let's get started.

GeometricObject class

This class will hold all attributes for the triangle. The Triangle class will extend this class. The GeometricObject class is an abstract class. This means that you cannot create an instance of the class. It will also hold some methods that are abstract as well. Abstract methods do not have a body. Any class that extends an abstract class MUST contain methods with the same name as the abstract methods. They provide the body of the abstract methods, although the abstract class will process the information.

1. Open a new document window in either TextPad or another IDE and name the file GeometricObject.java.
2. First we want to include the purpose of the class by typing:

   ```
   // This class will contain basic attributes for all shapes and is not limited to just Triangles
   ```

   - All programs should include the purpose of the class. Documentation is important in larger programs.

3. We type the class header and opening brace:

   ```
   public abstract class GeometricObject
   {
   ```

   - Abstract classes must have the word abstract in the class header.

4. Declare our variables, including a comment before the section.

   ```
   //declare variables
   private String color ;
   private boolean filled;
   ```

5. We create the default constructor method. This method will set the default values of the declared variables. Type:

   ```
   /**Default construct*/
   protected GeometricObject()
   {
       color = "white";
       filled = true;
   }
   ```

6. Overload the constructor method by typing:

   ```
   /**Construct a geometric object*/
   protected GeometricObject(String color, boolean filled)
   {
           this.color = color;
           this.filled = filled;
   ```

```
}
```

7. We create the accessor and mutator method for the color variable. Type:

```
/**Getter method for color*/
public String getColor()
{
	return color;
}

/**Setter method for color*/
public void setColor(String color)
{
	this.color = color;
}
```

8. We create an accessor and mutator method for the fill. In this case though the accessor method does not start with the word get. Since this method returns a boolean we name the method so that it best explains the purpose. Type:

```
/**Getter method for filled. Since filled is boolean,
so, the get method name is isFilled*/
public boolean isFilled()
{
	return filled;
}

/**Setter method for filled*/
public void setFilled(boolean filled)
{
	this.filled = filled;
}
```

   - The keyword "this" is used because the name of the variable passed to the method is exactly the same as the variable declared in this class. The keyword "this" refers to the variable declared within the class.

9. Our final two methods are abstract methods and will not have a body. Abstract methods simply have the method header and ends with a semicolon. Type:

```
/**Abstract method findArea*/
public abstract double getArea();

/**Abstract method getPerimeter*/
public abstract double getPerimeter();
```

- Since these are abstract, any class that extends this class must have methods of the same name, although the methods in the class will not be abstract.

10. Close the class.
11. Compile the class and fix any errors if necessary.

The Triangle class

The Triangle class will extend the GeometricObject class. Because of this any abstract methods in the GeometricObject class must also be present within this class.

1. Open a new document and save the file as Triangle.java. We start off by typing the class header of:

```
public class Triangle extends GeometricObject
{
```

2. We declare our variables:

```
//declare variables for the sides of the triangle and its color
private double side1, side2, side3;
private String color;
```

3. We create the default constructor method, setting default values for the variables. The color we set to having no color:

```
/** Default Constructor */
public Triangle()
{
    side1= 1.0;
    side2 = 1.0;
    side3 = 1.0;
    color = "No color";
}
```

4. We overload the constructor method:

```
/** Constructor */
public Triangle(double side1, double side2, double side3, String color) {
    this.side1 = side1;
    this.side2 = side2;
    this.side3 = side3;
    this.color = color;
}
```

5. Next we create all of our accessor methods for each variable. Type:

```
public double getSide1() {
        return side1;
}

public double getSide2() {
        return side2;
}

public double getSide3() {
        return side3;
}

@Override /** Override method findArea in GeometricObject */
public double getArea() {
        double s = (side1 + side2 + side3) / 2;
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
}

public String getColor() {
         return color;
}
```

6. We implement the abstract method of getArea(). Type:

```
/** Implement the abstract method findArea in GeometricObject */
public double getArea()
```

```
{
        double s = (side1 + side2 + side3) / 2;
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
}
```

- This method is used to find the area of the GeometricObject.

7. We implement the abstract method of getPerimeter() to find the circumference of the triangle Type:

```
/** Implement the abstract method findCircumference in
* GeometricObject
**/
public double getPerimeter()
{
        return side1 + side2 + side3;
}
```

8. Finally we override the toString() method in the GeometricObject class. You might say that there is no such method. This method is automatically available for use and is made to be overridden. If the method is not present in the class the memory location of the object to be printed is displayed rather than the attribute values of the object. Type:

```
/** Override the toString method */
public String toString()
{
        // Implement it to return the three sides and the color
        return "Triangle: side1 = " + side1 + " side2 = " + side2 + " side3 = " + side3 + "
       color = " + color;
}
```

9. Close the class.
10. Compile the program and fix any errors if necessary.

Chapter11Lab1 class

This is the main class. It will contain the necessary code to run the program. We will create another class named Triangle after this class. As long as we do not include and access modifier such as "public" at the beginning of the class, this is acceptable. This is often done when one class is the only class to access it. If we wanted t reuse it for numerous classes we would create it in its own separate file.

1. Open a new document window and save the file as Chapter11Lab1.java.
2. First we create comments telling the purpose of the class. Type:

```
/*  This program will request the size of the sides of the triangle and determine the area
and perimeter.
It will also ask if the user wants a fill color.  If no fill is designated it will display that
the triangle will not be solid.
*/
```

3. Type the code that will import the Scanner class, located in the java.util package.
4. Type the class header and opening brace.
5. First we create our variables. Some will receive values and others will not. Type:

```
//declare the variable, Scanner object and set repeat to yes
Scanner input = new Scanner(System.in);
double[] sides = new double[3];
String color;
boolean filled = false;
   String repeat = "yes";
```

- The Scanner object is for user input.
- An array that will hold the values of the 3 sides of the triangle
- The first String variable is for the color.
- The boolean is set to whether the triangle will be filled with the entered color.
- The last String is for repeating the questions for a triangle of a different size and color.

6. Within a do/while loop we will request input from the user. Type:

```
//create a do/while loop that will run as long as repeat equals yes
do
{
          System.out.print("Enter three sides: ");
           for(int i=0; i< sides.length; i++)
           {
                    sides[i] = input.nextDouble();
           }

           System.out.println("Enter the color: ");
```

```
color = input.next();

Triangle triangle = new Triangle(sides[0], sides[1], sides[2], color);


System.out.println("Enter true if you want to fill the triangle with the color: " +
color + ".  \nIf you do not want to fill the triangle with color please enter
false.");

filled = input.nextBoolean();

if(filled == false)
{
        color = "No color";
        triangle.setColor(color);
}

triangle.setFilled(filled);
```

- A for loop is used to apply the user input to the values of the array.
- We request the user to enter a color.
- We ask whether the user wants the triangle to be filled with this color, and to enter either true or false.
- We use an if statement to check to see whether the triangle is to be filled with this color and we pass the boolean value to the setFilled() method in the Triangle class.

7. We display the area and perimeter of the triangle. Type:

```
System.out.println();
System.out.println("The area is " + triangle.getArea());
System.out.println("The perimeter is " + triangle.getPerimeter());
System.out.println("The color is " + triangle.getColor());
System.out.println(triangle);
System.out.println();

System.out.println("Do you wish to try enter new values?  Please enter yes/no");
repeat = input.next();

 }while(repeat.equalsIgnoreCase("yes"));
```

```
}
}
```

8. Compile the class and fix any errors.
11. Run and test the program.
12. Compress all files (both .java and .class) into a single zip or rar file and submit to the drop box.