Problem 13.1

```
public class TriangleDriver {

	public static void main (String[] args) {
		Triangle triangle = new Triangle(1, 1.5, 1);
		triangle.setColor("Yellow");
		triangle.ifFilled(true);
		System.out.println(triangle);
		System.out.println("The area is "+ triangle.getArea());
		System.out.println("The perimeter is "+ triangle.getPerimeter());
		System.out.println(triangle);
	}

}

public abstract class GeometricObject1 {
	private String color = "white";
	private boolean filled;

	protected GeometricObject1() {
	}

	protected GeometricObject1 (String color, boolean filled) {
		this.color = color;
		this.filled = filled;
	}

	public String getColor() {
		return color;
	}
	public void setColor(String color) {
		this.color = color;
	}
	public boolean isFilled() {
		return filled;
	}
	public void ifFilled(boolean filled) {
		this.filled = filled;
	}
```

```
	public abstract double getArea();
	public abstract double getPerimeter();

}
```

Problem 13.3

```
import java.util.ArrayList;
import java.util.Scanner;

public class problem13_3 {

	public static void main (String[] args) {
		ArrayList<Number> list = new ArrayList<Number>();
		Scanner input = new Scanner(System.in);

		System.out.print("Enter 5 numbers: ");
		int i=0;
		while(i<5) {
			list.add(input.nextInt());
			i++;
		}
		sort(list);

		System.out.print("List after sorting: ");
		for(int j=0; j<list.size(); j++)
			System.out.print(list.get(j)+" ");
	}
	public static void sort(ArrayList<Number> list) {
		Number temp;
		for(int i=0; i<list.size(); i++) {
			for(int j=i+1; j<list.size()-1; j++) {
			if(list.get(i).intValue() >list.get(j).intValue() ) {
				temp = list.get(i);
				list.set(i, list.get(j));
				list.set(j, temp);
			}
		}
	 }
```

```
    }
}
```

Problem 13.5

```
public class GeometricObjectDemo {

    public static void main(String[] args) {

        GeometricObject circle1 = new Circle(5);
        GeometricObject circle2 = new Circle(6);

        GeometricObject rectangle1 = new Rectangle(2, 8);
        GeometricObject rectangle2 = new Rectangle(3.5, 4);

        GeometricObject maxCircle = (GeometricObject)
        GeometricObject.max(circle1, circle2);

        GeometricObject maxRectangle = (GeometricObject)
        GeometricObject.max(rectangle1, rectangle2);

        System.out.println("Circle 1: ");
        printGeometricObject(circle1);

        System.out.println("Circle 2: ");
        printGeometricObject(circle2);

        System.out.println("Largest Circle:");
        printGeometricObject(maxCircle);

        System.out.println("Rectangle 1: ");
        printGeometricObject(rectangle1);

        System.out.println("Rectangle 2: ");
        printGeometricObject(rectangle2);

        System.out.println("Largest Rectangle: ");
        printGeometricObject(maxRectangle);
```

```
	}
	public static void printGeometricObject(GeometricObject obj) {
		System.out.println("Area: "+obj.getArea());
		System.out.println("Perimeter: "+obj.getPerimeter());
		System.out.println();
	}

}


public abstract class GeometricObject implements Comparable {

	public int compareTo(Object obj) {
		if(getArea() > ((GeometricObject)obj).getArea()) {
			return 1;
		}
		else if(getArea() < ((GeometricObject)obj).getArea()) {
			return-1;
		}
		else
			return 0;
	}
	public abstract double getArea();
	public abstract double getPerimeter();

	public static Comparable max(Comparable obj1, Comparable obj2) {
		if(obj1.compareTo(obj2) >= 0) {
			return obj1;
		}
		else {
			return obj2;
		}
	}
}


class Circle extends GeometricObject{
	public double radius;
	public Circle(double radius) {
```

```
            this.radius = radius;
        }
        public double getPerimeter() {
            return  Math.PI * 2 * radius;
        }
        public double getArea() {
            return Math.PI * (radius * radius);
        }
}


class Rectangle extends GeometricObject {
        double length;
        double width;

        Rectangle(double length, double width)
        {
           this.length = length;
           this.width = width;
        }
        public double getArea()
        {
           return length * width;
        }
        public double getPerimeter()
        {
           return 2 * (length + width);
        }
    }
```

Problem 13.7

```
interface Colorable {

    public void howToColor ();

}
  class GeometricObject4 {
    public GeometricObject4 () {
```

```
	}

}
   class Square extends GeometricObject4 implements Colorable {
	public Square () {

	}
	public void howToColor() {
		System.out.println("Colorable all four sides");
	}
   }


public class ColorableTest{
	public static void main(String[] args) {
		GeometricObject4[] obj = new GeometricObject4[5];

		obj[0] = new GeometricObject4();
		obj[1] = new Square();
		obj[2] = new Square();
		obj[3] = new GeometricObject4();
		obj[4] = new Square();

		for (int i=0; i<obj.length;i++) {
			GeometricObject4 object = obj[i];
			System.out.print("Object["+i+ "] ");
			if (object instanceof Colorable) {
				Colorable colorable = (Colorable) object;
				colorable.howToColor();
			} else {
				System.out.println("Not colorable");
			}
		}

	}

}
```

Problem 13.9

```
abstract class GeometricObject5 {
	private String color = "white";
	private boolean filled;
	private java.util.Date dateCreated;

	protected GeometricObject5() {
		dateCreated = new java.util.Date();
	}
	protected GeometricObject5(String color, boolean filled) {
		dateCreated = new java.util.Date();
		this.color = color;
		this.filled = filled;
	}
	public String getColor() {
		return color;
	}
	public void setColor(String color) {
		this.color = color;
	}
	public boolean isFilled(boolean filled) {
		return filled;
	}
	public void setFilled(boolean filled) {
		this.filled = filled;
	}
	public java.util.Date getDateCreated() {
		return dateCreated;
	}
	public String toString() {
		return "created on "+dateCreated+"\ncolor: "+color+" and filled: "+filled;
	}
	public abstract double getArea();
	public abstract double getPerimeter();
}


class Circle2 extends GeometricObject5{
```

```
	public double radius;
	public Circle2(double radius) {
		this.radius = radius;
	}
	public double getPerimeter() {
		return Math.PI * radius * radius;
	}
	public double getArea() {
		return 2 * Math.PI * radius;
	}
	public boolean equals(Object obj) {
		if (this.radius == ((Circle2)obj).radius)
			return true;
		return false;
	}
}

class ComparableCircle extends Circle2 implements Comparable {
	public ComparableCircle(double rad){
		super(rad);
	}
	public int compareTo (Object o) {
		if(getArea() < ((ComparableCircle)o).getArea())
			return 1;
		else if(getArea() < ((ComparableCircle)o).getArea())
			return -1;
		else
			return 0;
	}
}



class TestEqualCircle {
	public static void main(String[] args) {
		Circle2 cc = new Circle2 (6);
		Circle2 cc1 = new Circle2 (6);
boolean res = cc.equals(cc1);
System.out.println("Result: "+res);
```

```
        }
}
```

Problem 13.11

```
class Octagon extends GeometricObject6 implements Comparable, Cloneable {
        private double side;

        public Octagon () {
        }
        public Octagon (double side) {
                this.side = side;
        }
        public void setSide(double side) {
                this.side = side;
        }
        public double getSide() {
                return side;
        }
        public double getArea() {
                return (2+(4/Math.sqrt(2)))*side*side;
        }
        public double getPerimeter() {
                return 8*side;
        }
        public int compareTo(Object obj) {
                if(getArea() == ((Octagon)obj).getArea())
                        return 0;
                else if (getArea() > ((Octagon)obj).getArea())
                        return 1;
                else
                        return -1;
        }
        public Object clone() throws CloneNotSupportedException {
                return super.clone();
        }

}
```

```
abstract class GeometricObject6 {
	private String color = "White";
	private boolean filled;
	private java.util.Date dateCreated;

	protected GeometricObject6() {
		dateCreated = new java.util.Date();
	}
	protected GeometricObject6(String color, boolean filled) {
		dateCreated = new java.util.Date();
		this.color = color;
		this.filled = filled;
	}
	public String getColor() {
		return color;
	}
	public void setFilled(boolean filled) {
		this.filled =filled;
	}
	public java.util.Date getDateCreated() {
		return dateCreated;
	}
	public String toString() {
		return "created on "+dateCreated+"\ncolor: "
				+color+" and filled: "+filled;
	}
	public abstract double getArea();
	public abstract double getPerimeter();
}

public class OctagonTest {

	public static void main(String[] args) throws CloneNotSupportedException{
		// TODO Auto-generated method stub
		Octagon test1 = new Octagon(5.0);
		Octagon test2 = (Octagon)test1.clone();

		System.out.println("The Area of first object="+test1.getArea());
		System.out.println("The Perimeter of first object="+test1.getPerimeter());
```

```
            System.out.println("The Area of the clone object="+test2.getArea());
            System.out.println("The Perimeter of the clone object="+test2.getPerimeter());
            System.out.println("\nCompare the two objects:");

            switch (test1.compareTo(test2)) {
            case 1:
                  System.out.println("The first Object is larger than the clone object");
                  break;
            case 0:
                  System.out.println("Those objects are equal");
                  break;
            case(-1):
                  System.out.println("The first Object is Smaller than the clone Object");
            break;
            }

      }

}
```