```
public class Student
{
    //data members
    public final static int NUM_TESTS = 3;
    private String myName;
    private int[] myTests;
    private String myGrade;

    //constructors
    public Student()
    {
        myName = "";
        myTests = new int[NUM_TESTS];
        myGrade = "";
    }

    public Student(String name, int[] tests, String grade)
    {
        myName = name;
        myTests = tests;
        myGrade = grade;
    }

    public String getName()
    { return myName; }

    public String getGrade()
    { return myGrade; }

    public void setGrade(String newGrade)
    { myGrade = newGrade; }

    public void computeGrade()
    {
        if (myName.equals(""))
            myGrade = "No grade";
        else if (getTestAverage() >= 65)
            myGrade = "Pass";
        else
            myGrade = "Fail";
    }

    public double getTestAverage()
    {
        double total = 0;
        for (int score : myTests)
            total += score;
        return total/NUM_TESTS;
    }
}
```

```
public class UnderGrad extends Student
{
    public UnderGrad()   //default constructor
    { super(); }

    //constructor
    public UnderGrad(String name, int[] tests, String grade)
    { super(name, tests, grade); }

    public void computeGrade()
    {
        if (getTestAverage() >= 70)
            setGrade("Pass");
        else
            setGrade("Fail");
    }
}

public class GradStudent extends Student
{
    private int myGradID;

    public GradStudent()      //default constructor
    {
        super();
        myGradID = 0;
    }

    //constructor
    public GradStudent(String name, int[] tests, String grade,
            int gradID)
    {
        super(name, tests, grade);
        myGradID = gradID;
    }

    public int getID()
    { return myGradID; }

    public void computeGrade()
    {
        //invokes computeGrade in Student superclass
        super.computeGrade();
        if (getTestAverage() >= 90)
            setGrade("Pass with distinction");
    }
}
```

a)

```
Student s = new Student("Brian Lorenzen", new int[] {90,94,99},
        "none");
Student u = new UnderGrad("Tim Broder", new int[] {90,90,100},
        "none");
Student g = new GradStudent("Kevin Cristella",
        new int[] {85,70,90}, "none", 1234);
```

b)

```
Student s = null;
Student u = new UnderGrad("Tim Broder", new int[] {90,90,100},
        "none");
Student g = new GradStudent("Kevin Cristella",
        new int[] {85,70,90}, "none", 1234);
System.out.print("Enter student status: ");
System.out.println("Grad (G), Undergrad (U), Neither (N)");
String str = IO.readString();        //read user input
if (str.equals("G"))
    s = g;
else if (str.equals("U"))
    s = u;
else
    s = new Student();
s.computeGrade();
```

c)

```
public class StudentTest
{
    public static void computeAllGrades(Student[] studentList)
    {
        for (Student s : studentList)
            if (s != null)
                s.computeGrade();
    }

    public static void main(String[] args)
    {
        Student[] stu = new Student[5];
        stu[0] = new Student("Brian Lorenzen",
                        new int[] {90,94,99}, "none");
        stu[1] = new UnderGrad("Tim Broder",
                        new int[] {90,90,100}, "none");
        stu[2] = new GradStudent("Kevin Cristella",
                        new int[] {85,70,90}, "none", 1234);
        computeAllGrades(stu);
    }
}
```

```
public abstract class Shape
{
    private String myName;

    //constructor
    public Shape(String name)
    { myName = name; }

    public String getName()
    { return myName; }

    public abstract double area();
    public abstract double perimeter();

    public double semiPerimeter()
    { return perimeter() / 2; }
}

public class Circle extends Shape
{
    private double myRadius;

    //constructor
    public Circle(double radius, String name)
    {
        super(name);
        myRadius = radius;
    }

    public double perimeter()
    { return 2 * Math.PI * myRadius; }

    public double area()
    { return Math.PI * myRadius * myRadius; }
}

public class Square extends Shape
{
    private double mySide;

    //constructor
    public Square(double side, String name)
    {
        super(name);
        mySide = side;
    }

    public double perimeter()
    { return 4 * mySide; }

    public double area()
    { return mySide * mySide; }
}
```

```
Shape circ = new Circle(10, "circle");
Shape sq = new Square(9.4, "square");
Shape s = null;
System.out.println("Which shape?");
String str = IO.readString();          //read user input
if (str.equals("circle"))
    s = circ;
else
    s = sq;
System.out.println("Area of " + s.getName() + " is "
        + s.area());
```



```
public abstract class Shape implements Comparable
{
    private String myName;

    //constructor
    public Shape(String name)
    { myName = name; }

    public String getName()
    { return myName; }

    public abstract double area();
    public abstract double perimeter();

    public double semiPerimeter()
    { return perimeter() / 2; }

    public int compareTo(Object obj)
    {
        final double EPSILON = 1.0e-15;    //slightly bigger than
                                           //machine precision
        Shape rhs = (Shape) obj;
        double diff = area() - rhs.area();
        if (Math.abs(diff) <= EPSILON * Math.abs(area()))
            return 0;  //area of this shape equals area of obj
        else if (diff < 0)
            return -1; //area of this shape less than area of obj
        else
            return 1;  //area of this shape greater than area of obj
    }
}
```

Here is a program that finds the larger of two Comparable objects.

```
public class FindMaxTest
{
    /* Return the larger of two objects a and b. */
    public static Comparable max(Comparable a, Comparable b)
    {
        if (a.compareTo(b) > 0) //if a > b ...
            return a;
        else
            return b;
    }

    /* Test max on two Shape objects. */
    public static void main(String[] args)
    {
        Shape s1 = new Circle(3.0, "circle");
        Shape s2 = new Square(4.5, "square");
        System.out.println("Area of " + s1.getName() + " is " +
                s1.area());
        System.out.println("Area of " + s2.getName() + " is " +
                s2.area());
        Shape s3 = (Shape) max(s1, s2);
        System.out.println("The larger shape is the " +
                s3.getName());
    }
}
```

Here is the output:

```
Area of circle is 28.27
Area of square is 20.25
The larger shape is the circle
```

```
import java.util.*;

public class PointTest {

        public static void main(String[] args) {

                Point p1 = new Point(1,1);
            Point p2 = new Point(5,5);

            if(p1.compareTo(p2) > 0) {
                        System.out.println("P1 is farther from the orign--is
                        larger");
                }else if(p1.compareTo(p2) < 0) {
                        System.out.println("P2 is farther from the orign--is
                        larger");
                }else {
                        System.out.println("P1 and P2 are the same distance
                        from the origin--equal in size");
                }
         }
}

class Point implements Comparable {

        private int x;
        private int y;

        public Point(int xVal, int yVal) {
                x = xVal;
                y = yVal;
        }

        public Point() {

                x = 0;
                y = 0;
        }

        // A point is larger if its distance is farther from the origin.
        public int compareTo(Object other) {
                Point aPoint = (Point) other;
                if(distanceFromOrigin() > aPoint.distanceFromOrigin()) {
                //implicit parameter is larger
                    return 1;
                }else if (Math.abs(distanceFromOrigin() -
                aPoint.distanceFromOrigin()) < .0001) {  //both ponts are
                equal in distance
                    return 0;
                }
        return -1; //explicit parameter is larger
        }

        public double distanceFromOrigin() {
            return Math.sqrt(x * x + y * y);
        }


        public String toString() {
            return "(" + x + ", " + y + ")";
        }
}
```