Exercise 6:

Summary: Implement a software framework that allows you to set and compute some properties of triangles and rectangles. More specifically, you should be able to set the points of these shapes, to compute their unit normals and their area. **Exercise** is fully implemented.

Task 1: Implement the class Point and the interface AlgebraSupport (2 points)

The class **Point** has the class diagram shown below.

The instance variables are the point coordinates and are initialized in the constructor.

Their values are returned and set by the instance methods. The constructor uses them to set the coordinates.

toString() returns ”Point = (” followed by the values of x, y, z separated by a comma and followed by the closing parenthesis ”)”.

Point
−x, y, z : double
+Point(double, double, double) +getX() : double +getY() : double +getZ() : double +setX(double) : void +setY(double) : void +setZ(double) : void +toString() : String

Create the interface **AlgebraSupport**. It declares two instance methods that all triangles and rectangles should have available. A method *comparePoints(Point p1, Point p2)* that compares two points. It returns *true* if *p1* and *p2* have identical coordinates and *false* otherwise.

It also declares the method *setNormal(double n1, double n2, double n3)* with the return type void. It uses the 3 components in the argument list to set the normal and the area of a rectangle or triangle. It should set the area of the rectangle to the length of $(n1, n2, n3)$ and it should set the area of the triangle to 0.5 times the length. It should also compute the unit normal from these values. The latter sets the values of nX, nY, nZ discussed below.

Task 2: Implement the shapes (6 points)

Develop the superclass **Shape**, which implements **AlgebraSupport**:

Shape
−pointCounter : int #thePoints : Double[] #area : double #nX, nY, nZ : double
+Shape() +getPoint() : Point +setPoint(Point) : String +toString() : String

pointCounter is used by *getPoint()* and initialized to 0.

thePoints is a static array for the corner points of a triangle or rectangle.

area corresponds to the area of the shape.

nX, nY, nZ are the components of the unit normal vector.

getPoint() returns a point of the shape. It returns the point at the slot *pointCounter* of *thePoints* and increases *pointCounter* by 1. If the value of *pointCounter* exceeds the length of *thePoints* it should set *pointCounter* to 0.

setPoint(arg) compares *arg* to the points in *thePoints*. If *arg* is different from the ones of *thePoints* and if there is a free slot available in *thePoints* then it should add it to the array and return "Added". Otherwise it should return "not added".

The method *toString()* should return the string "has the area *area* and normal *(nX, nY, nZ)*" where the values of *area, nX, nY, nZ* are expressed using 4 digits, two of which come after the radix point.

The two subclasses **Rectangle** and **Triangle** have the class diagrams

Rectangle
+Rectangle() +toString() : String

Triangle
+Triangle() +toString() : String

The constructor in each class initializes *thePoints* with the appropriate length.

The methods *toString()* return "The rectangle " or "The triangle " followed by the return value of the method *toString()* of the superclass.

One of the two methods declared in the interface should be implemented in **Shape** and the other in the subclasses. Remember that the area of a rectangle is given by the length of the cross product of its base vectors while the area of a triangle is given by half that value.

Task 2: Implement Algebra (2 points)

The class is the superclass of **Exercise** and has the class diagram

Algebra
+computeNormal(Shape) : void +createPoint(double, double,double) : Point

The method *createPoint(arg, arg, arg)* creates an instance of **Point** using the three values in the argument list.

computeNormal(arg) computes the edge vectors with two consecutive points (point 0 and point 1 is one edge and point 1 and point 2 the second) and computes the cross product of both. The length of the vector is related to the area of the shape and the unit vector sets the normal.

Run **Exercise** to see if everything works.