AP Computer Science
Introductory Project

Quadrilaterals (55 points)

The purpose of this project is to determine the type of a quadrilateral based upon its points on the coordinate plane. Your project must include a driver class (`QuadrilateralDriver`), but must also work with my driver. User input and output must be handled in the driver class, and you are permitted to use any method of doing so here (i.e., `Scanner`, `JOptionPane`, full GUI).

Your project must include the following classes, and must include (at least) the methods and fields listed.

Class `Point`

Fields:
```
double x, double y
```

Methods:
accessor and mutator methods for all fields
constructor that accepts two `double` parameters
```
double distance (Point p)
boolean equals (Point p)
Point midpoint (Point p)
double slope (Point p)
String toString()
static boolean approx(double a, double b)
```

Class `Quadrilateral`

Fields:
array of 4 `Point` objects to store the coordinates of the vertices

Methods:
accessor and mutator methods for all fields
constructor that accepts 4 `Point` objects
classification methods for each quadrilateral (trapezoid, parallelogram, rectangle, rhombus, square)
i.e., `boolean isTrapezoid()`
```
String toString()
static boolean approx(double a, double b)
```

Your project must follow all standard java conventions (naming, style, indentation, etc.). All data must be fully encapsulated, and all methods must have appropriate visibility modifiers. Your code MUST BE COMMENTED.

Bonus Methods (if you finish early):
Add some or all of the following transformation methods to the `Point` class. They should not modify the original `Point`, but rather they should return a new `Point` object which represents the result of applying the given transformation to the given `Point`. They get harder as you go down the list.

```
Point translate(double deltaX, double deltaY)
Point dilate(double factor)   //dilates by given factor, center at origin
Point reflectYEqualsX()                 //reflects across line y = x
Point reflectHorizontal(double xValue)  //reflects across line x = xValue
Point reflectVertical(double yValue)    //reflects across line y = yValue
Point rotate(int angle)  //angle must be a multiple of 90°
Point dilate(double factor, Point center) //center of dilation variable
Point rotate(double angle) //rotates by angle (degrees) around origin
Point rotate(double angle, Point center) //rotates by angle around center
Point reflect(Point a, Point b) //reflects across line through a and b
```