```java
/** Represents a Cartesian (x,y) point */

import java.util.*;
import java.text.*;

public class Point {
    private double x, y;            // The coordinates of the point

    // Default constructor
    public Point( ) {
        x = 0.0;
        y = 0.0;
    }

    // A constructor that initializes the fields
    public Point(double x, double y) {
        this.x = x; this.y = y;
    }

    // A constructor that reads the fields from standard input
    public Point(Scanner s) {
        System.out.println ();
        System.out.print("Enter a point: ");
        x = s.nextDouble();
        y = s.nextDouble();
        System.out.println();
    }

    // Observer methods
    public double get_x( ) { return x; }
    public double get_y( ) { return y; }

    // Methods that operate on the x and y fields

    // Computes the distance of this point from the origin
    public double distanceFromOrigin( ) {
        return Math.sqrt(x*x + y*y);
    }

    // Computes the distance between this point and a given point
    public double distanceFromPoint(Point p) {
        double a, b;

        // Calculate differences in x and y coordinates
        a = this.get_x( ) - p.get_x( ); // Difference in x coordinates
        b = this.get_y( ) - p.get_y( ); // Difference in y coordinates

        // Use Pythagorean Theorem to calculate square of distance between Points
        return Math.sqrt(a*a + b*b);    // sqrt calculates square root
    }

    // Finds the mid point between this point and a given point
    public Point midPoint(Point p) {
        double x_midpoint, y_midpoint;

        // Compute the x and y midpoints
        x_midpoint = (this.get_x( ) + p.get_x( )) / 2;
        y_midpoint = (this.get_y( ) + p.get_y( )) / 2;

        // Construct a new Point and return it
        Point midpoint = new Point(x_midpoint, y_midpoint);
        return midpoint;
    }

    public void shift(double delta_x, double delta_y) {
        x += delta_x;
        y += delta_y;
     }

    public void rotate90( ) {
        double new_x;
        double new_y;

        new_x = y;  // For a 90 degree clockwise rotation the new y is -1
        new_y = -x; // times original x, and the new x is the original y
        x = new_x;
        y = new_y;
    }

    public String toString( ) {
        DecimalFormat df = new DecimalFormat("0.0");
```

```
83              return "(" + df.format(x) + "," + df.format(y) + ")";
84          }
85
86          // Reads the fields of a point from standard input
87          public static Point inputPoint(Scanner s) {
88              Point p = new Point(s);
89              return p;
90          }
91
92          // Adds two points together
93          public static Point addPoints(Point p1, Point p2) {
94              double x_sum, y_sum;
95              // Compute the x and y of the sum
96              x_sum = p1.get_x( ) + p2.get_x( );
97              y_sum = p1.get_y( ) + p2.get_y( );
98
99              Point sum = new Point(x_sum, y_sum);
100             return sum;
101         }
102
103         public boolean equals(Point p) {
104             return
105                 (this.get_x( ) == p.get_x( ))
106                 &&
107                 (this.get_y( ) == p.get_y( ));
108         }
109
110         public static void main(String[] args) {
111             Scanner in = new Scanner(System.in);
112             DecimalFormat df = new DecimalFormat("0.00");
113             Point p1 = new Point(1.0,5.1);
114             Point p2 = new Point(2.5,1.3);
115             Point p3 = new Point(in);
116             Point p4 = new Point();
117
118             System.out.println ("points p1, p2, p3 and p4 are:");
119             System.out.println ("p1 = " + p1.toString());
120             System.out.println ("p2 = " + p2.toString());
121             System.out.println ("p3 = " + p3.toString());
122             System.out.println ("p4 = " + p4.toString());
123
124             System.out.println ();
125             System.out.print ("Distance of P3 " + p3.toString());
126             System.out.println (" from the origin: " + df.format(p3.distanceFromOrigin()))
127
128             p4.shift(10,35);
129             System.out.println ();
130             System.out.println ("P4 after shifting: " + p4.toString());
131
132             Point p5 = inputPoint(in);
133             System.out.println ();
134             System.out.println ("p5 = " + p5.toString());
135
136             Point p6 = addPoints(p1, p2);
137             System.out.println ();
138             System.out.println ("p1 + p2 = " + p6.toString());
139
140             System.out.println ();
141             System.out.println (p3.toString() + " equals " + p5.toString() + " is " +
    p3.equals(p5));
142
143             p3.rotate90();
144             System.out.println ();
145             System.out.println ("P3 after rotation = " + p3.toString());
146
147             System.out.println ();
148             System.out.println (p3.toString() + " equals " + p5.toString() + " is " +
    p3.equals(p5));
149
150             System.out.println ();
151             System.out.print ("The distance between " + p1.toString() + " and ");
152             System.out.println (p2.toString() + " = " + df.format(p1.distanceFromPoint(p2)
153             System.out.println ();
154             System.out.print ("A midpoint between " + p1.toString() + " and ");
155             System.out.println (p2.toString() + " is " + p1.midPoint(p2).toString());
156
157             System.out.println ();
158             System.out.println ("P1 equals P2 is " + p1.equals(p2));
159         }
160     }
161
162
```