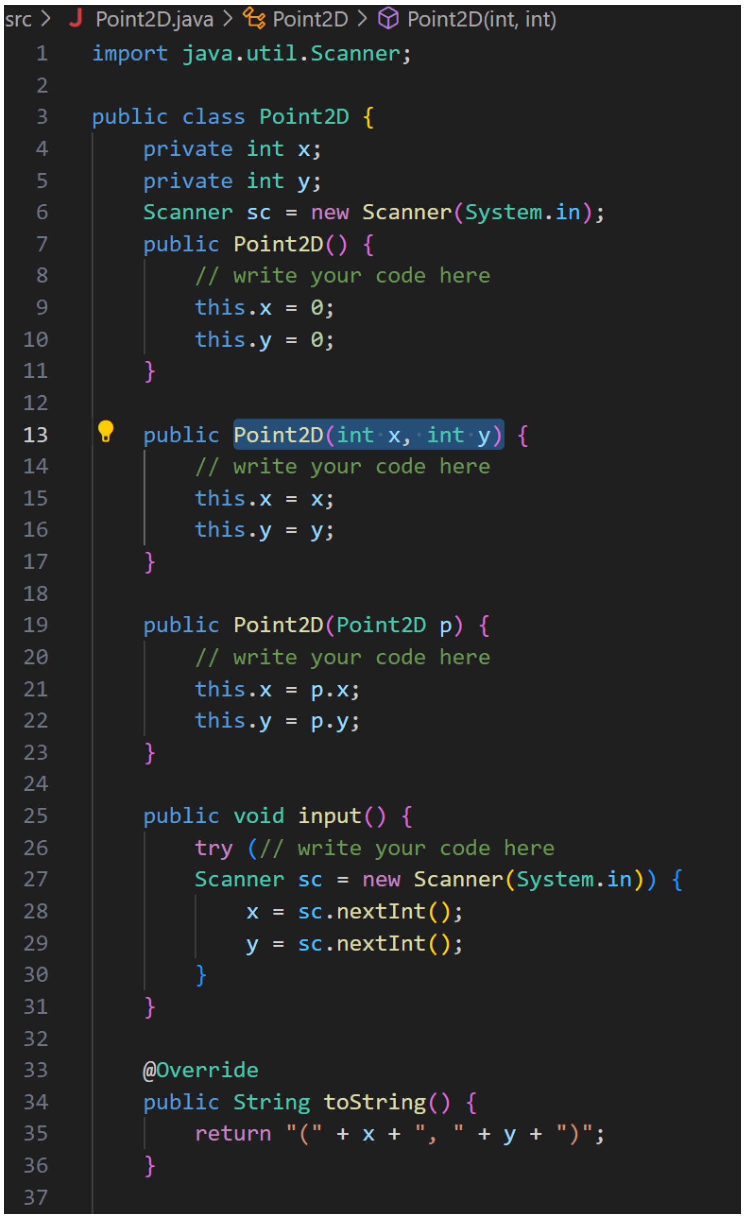
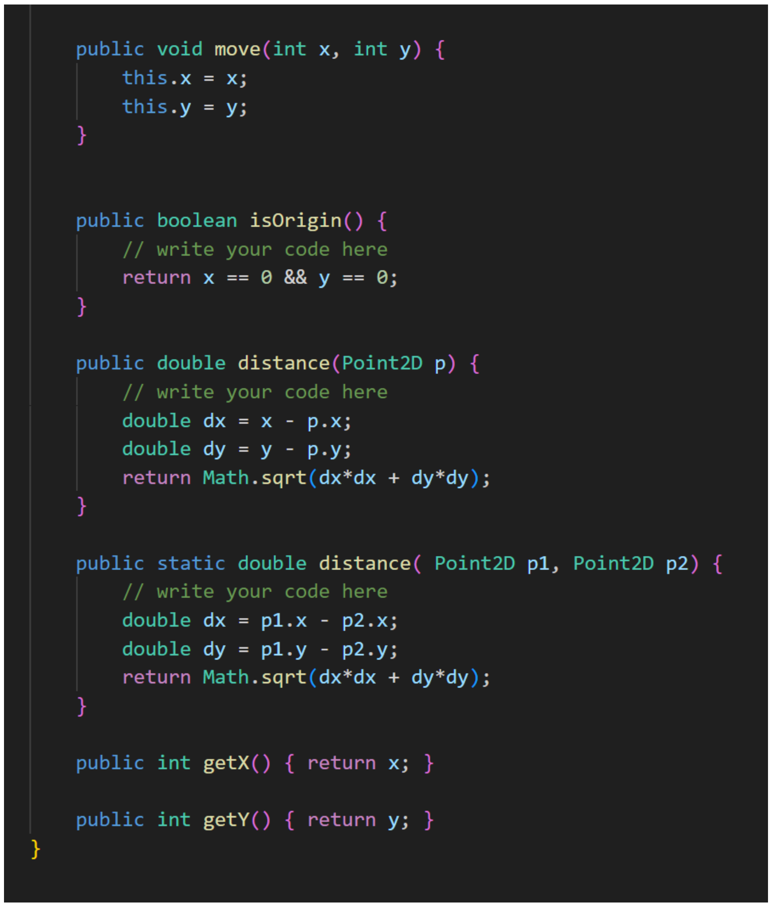
Task 1: Point 2D

Point2D



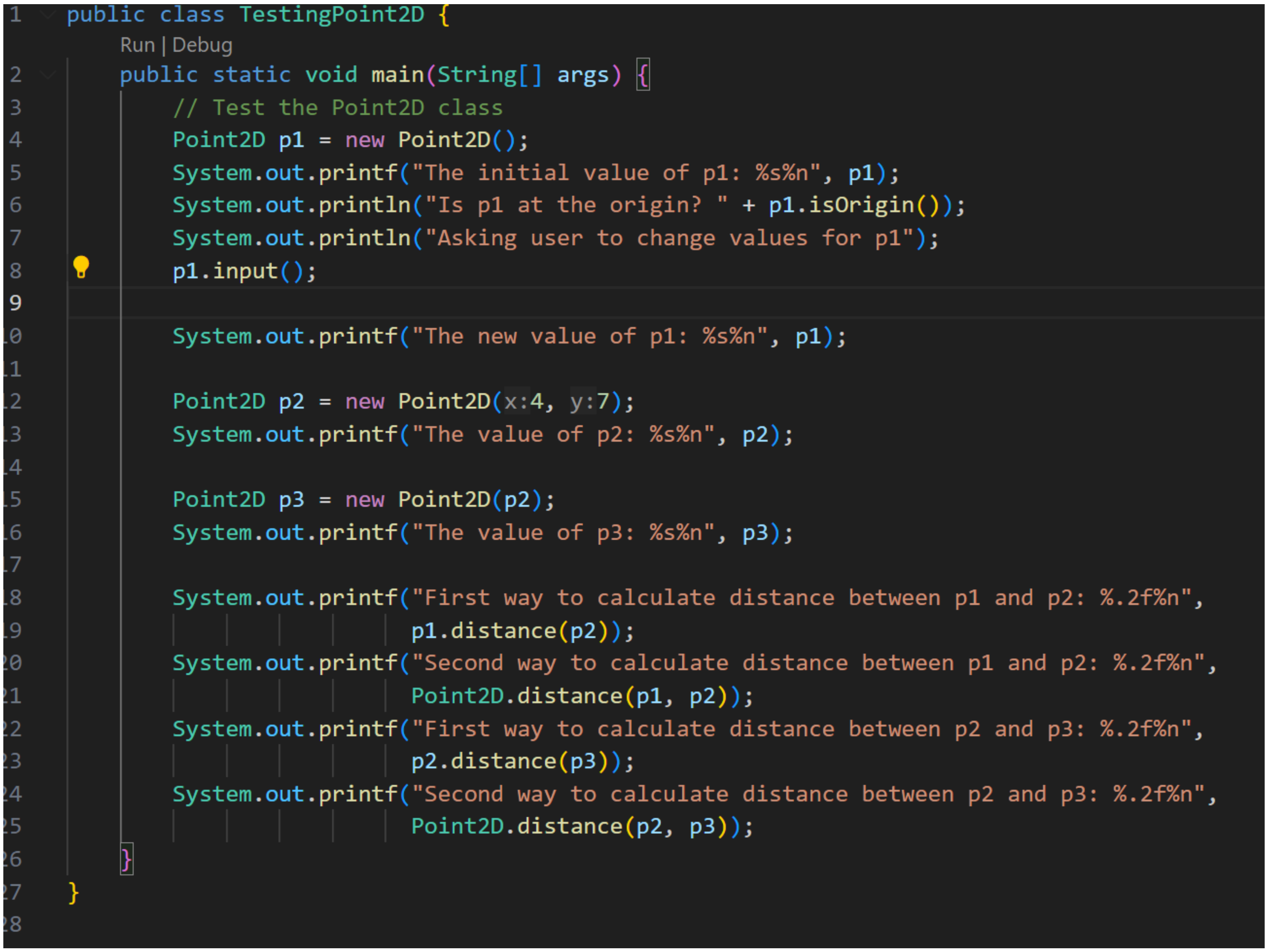
src > Point2D.java > Point2D > Point2D(int, int)

```java
import java.util.Scanner;

public class Point2D {
    private int x;
    private int y;
    Scanner sc = new Scanner(System.in);
    public Point2D() {
        // write your code here
        this.x = 0;
        this.y = 0;
    }

    public Point2D(int x, int y) {
        // write your code here
        this.x = x;
        this.y = y;
    }

    public Point2D(Point2D p) {
        // write your code here
        this.x = p.x;
        this.y = p.y;
    }

    public void input() {
        try (// write your code here
        Scanner sc = new Scanner(System.in)) {
            x = sc.nextInt();
            y = sc.nextInt();
        }
    }

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

```



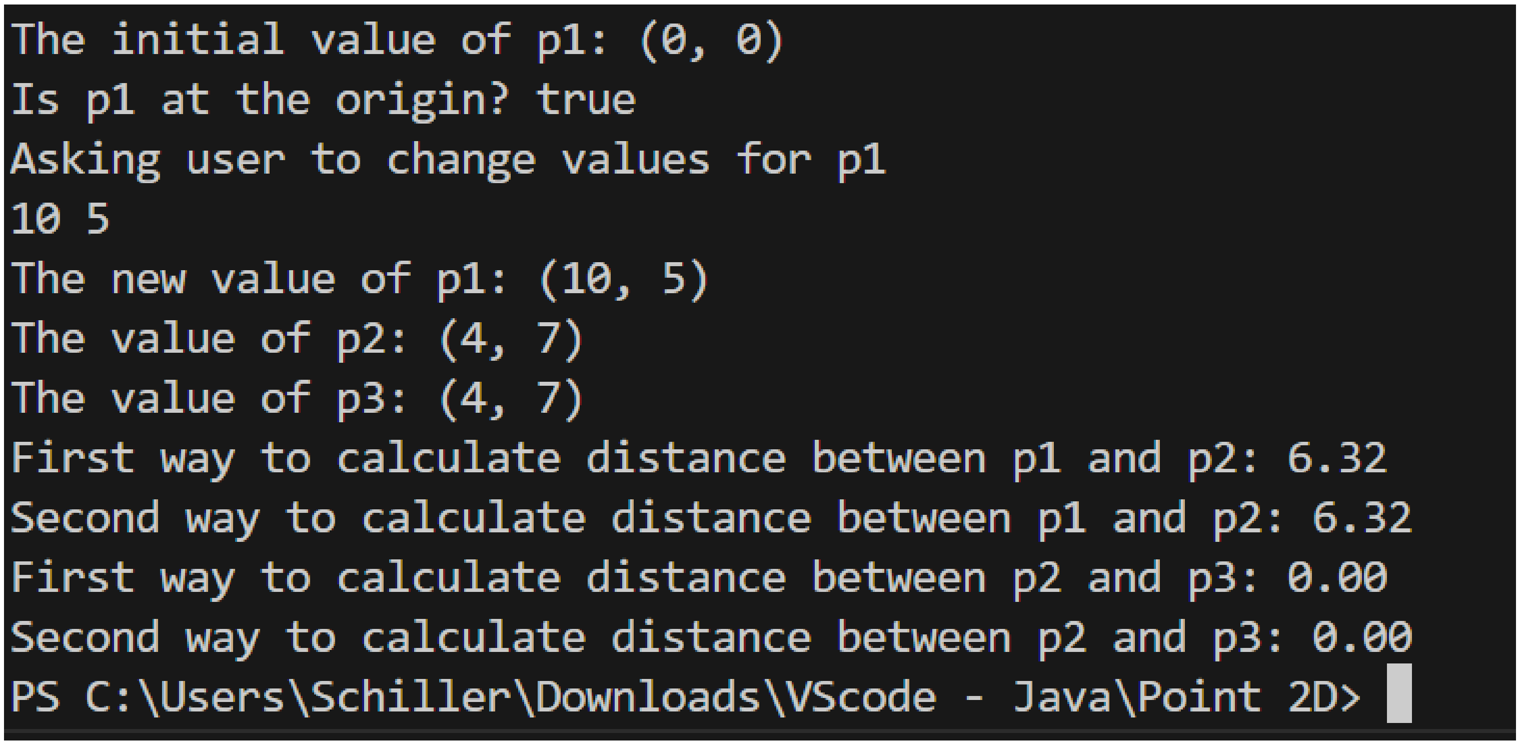
```java
    public void move(int x, int y) {
        this.x = x;
        this.y = y;
    }


    public boolean isOrigin() {
        // write your code here
        return x == 0 && y == 0;
    }

    public double distance(Point2D p) {
        // write your code here
        double dx = x - p.x;
        double dy = y - p.y;
        return Math.sqrt(dx*dx + dy*dy);
    }

    public static double distance( Point2D p1, Point2D p2) {
        // write your code here
        double dx = p1.x - p2.x;
        double dy = p1.y - p2.y;
        return Math.sqrt(dx*dx + dy*dy);
    }

    public int getX() { return x; }

    public int getY() { return y; }
}
```

TestingPoint2D



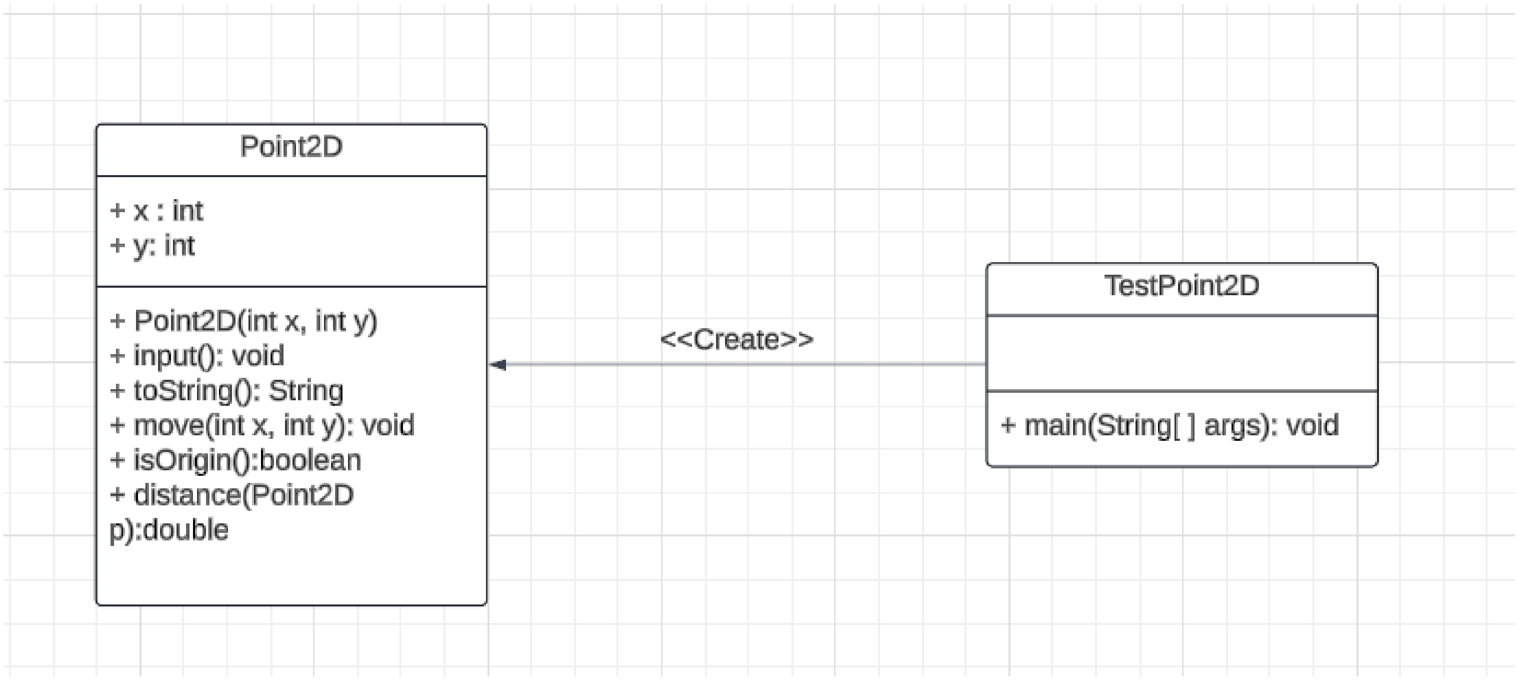
```
public class TestingPoint2D {
    Run | Debug
    public static void main(String[] args) {
        // Test the Point2D class
        Point2D p1 = new Point2D();
        System.out.printf("The initial value of p1: %s%n", p1);
        System.out.println("Is p1 at the origin? " + p1.isOrigin());
        System.out.println("Asking user to change values for p1");
        p1.input();

        System.out.printf("The new value of p1: %s%n", p1);

        Point2D p2 = new Point2D(x:4, y:7);
        System.out.printf("The value of p2: %s%n", p2);

        Point2D p3 = new Point2D(p2);
        System.out.printf("The value of p3: %s%n", p3);

        System.out.printf("First way to calculate distance between p1 and p2: %.2f%n",
                            p1.distance(p2));
        System.out.printf("Second way to calculate distance between p1 and p2: %.2f%n",
                            Point2D.distance(p1, p2));
        System.out.printf("First way to calculate distance between p2 and p3: %.2f%n",
                            p2.distance(p3));
        System.out.printf("Second way to calculate distance between p2 and p3: %.2f%n",
                            Point2D.distance(p2, p3));
    }
}
```

Output



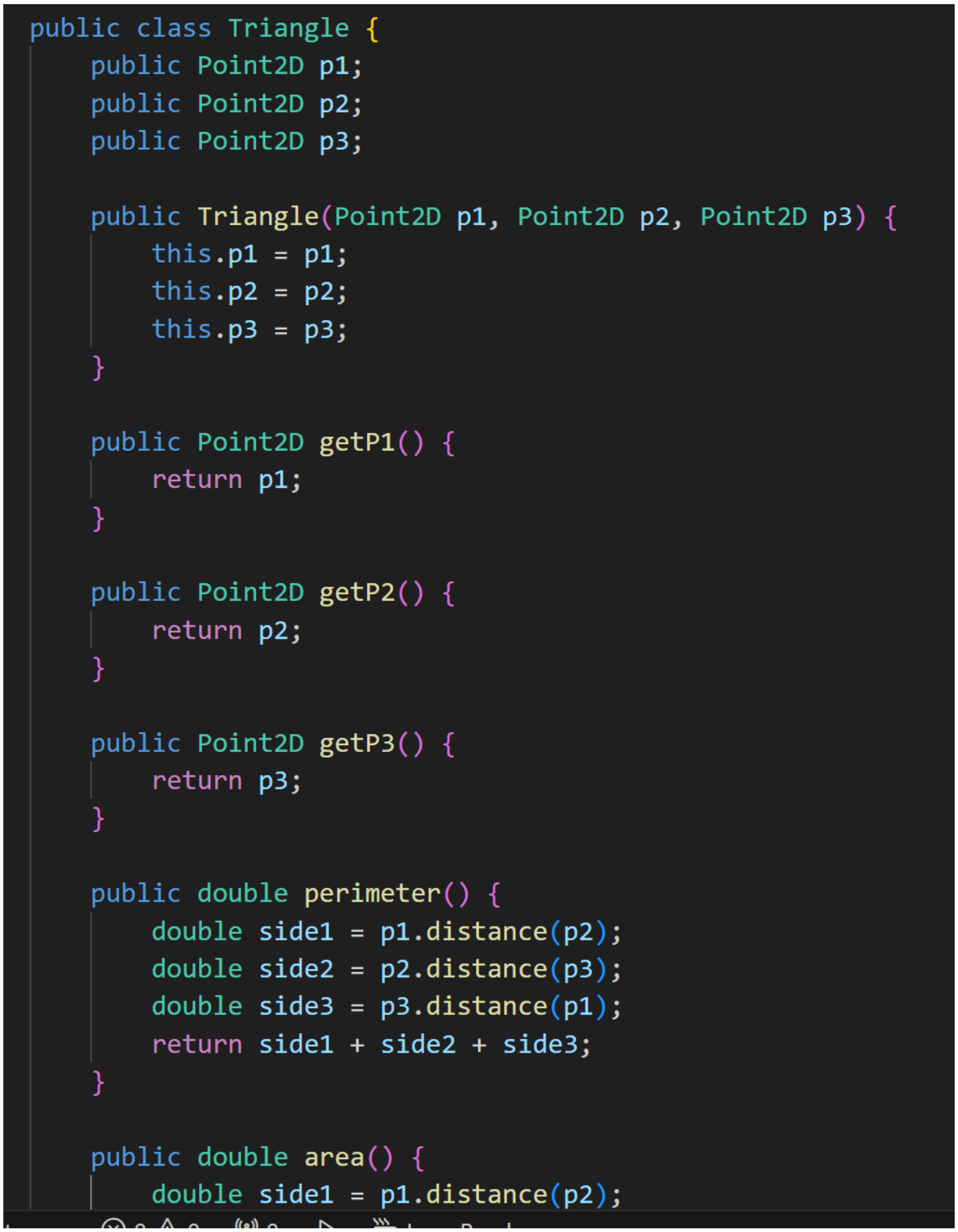
```
The initial value of p1: (0, 0)
Is p1 at the origin? true
Asking user to change values for p1
10 5
The new value of p1: (10, 5)
The value of p2: (4, 7)
The value of p3: (4, 7)
First way to calculate distance between p1 and p2: 6.32
Second way to calculate distance between p1 and p2: 6.32
First way to calculate distance between p2 and p3: 0.00
Second way to calculate distance between p2 and p3: 0.00
PS C:\Users\Schiller\Downloads\VScode - Java\Point 2D>
```

UML

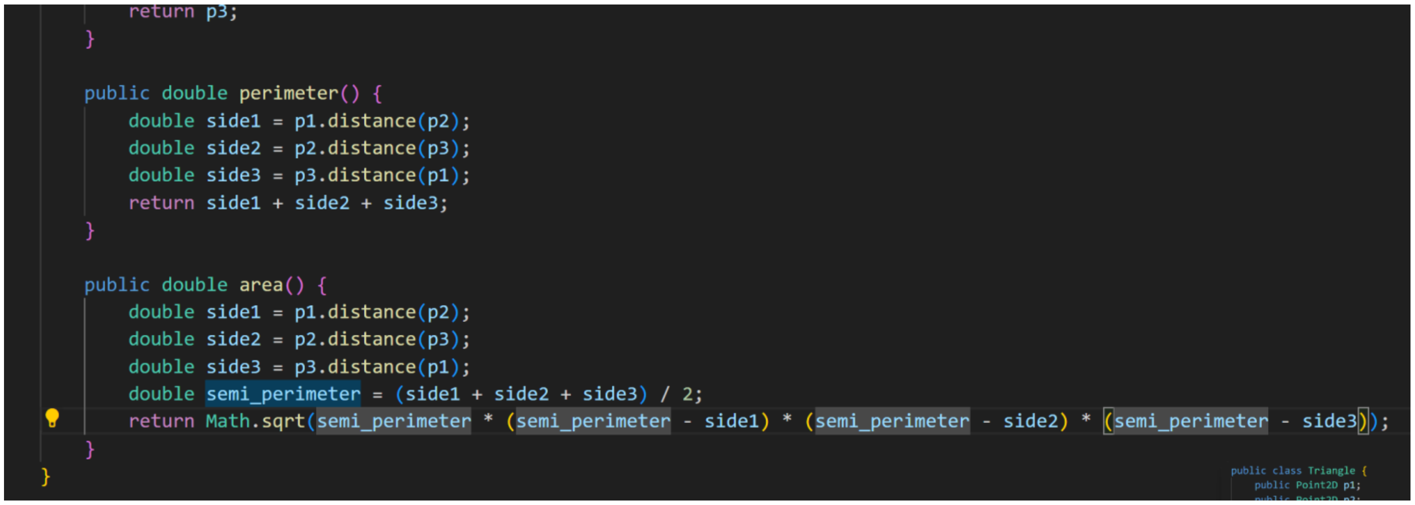


Task 2: Rectangle verification

Triangle

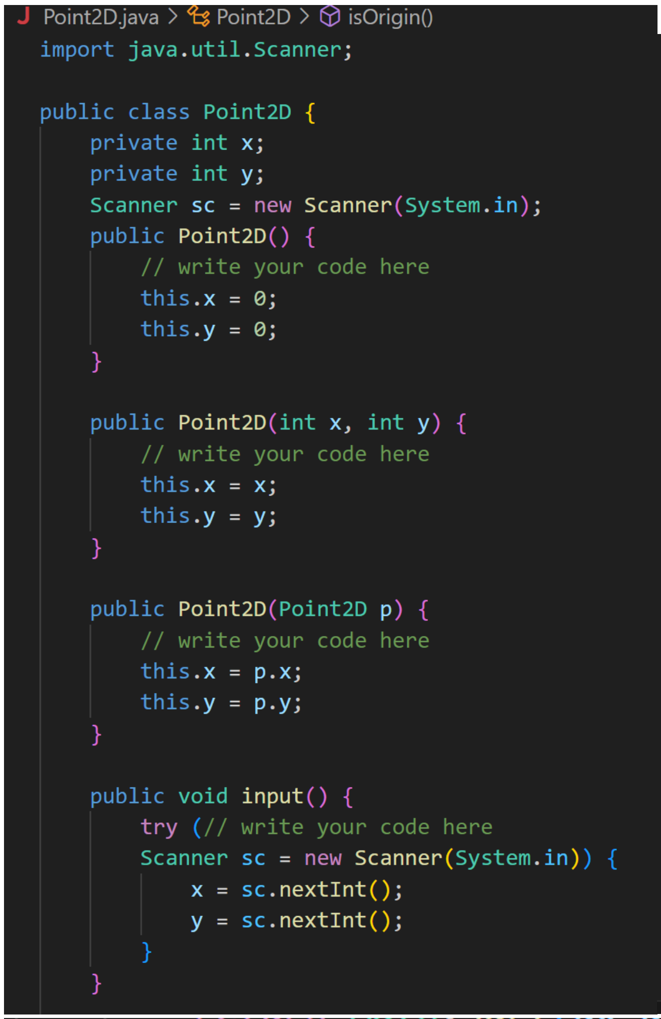


```java
public class Triangle {
    public Point2D p1;
    public Point2D p2;
    public Point2D p3;

    public Triangle(Point2D p1, Point2D p2, Point2D p3) {
        this.p1 = p1;
        this.p2 = p2;
        this.p3 = p3;
    }

    public Point2D getP1() {
        return p1;
    }

    public Point2D getP2() {
        return p2;
    }

    public Point2D getP3() {
        return p3;
    }

    public double perimeter() {
        double side1 = p1.distance(p2);
        double side2 = p2.distance(p3);
        double side3 = p3.distance(p1);
        return side1 + side2 + side3;
    }

    public double area() {
        double side1 = p1.distance(p2);
```



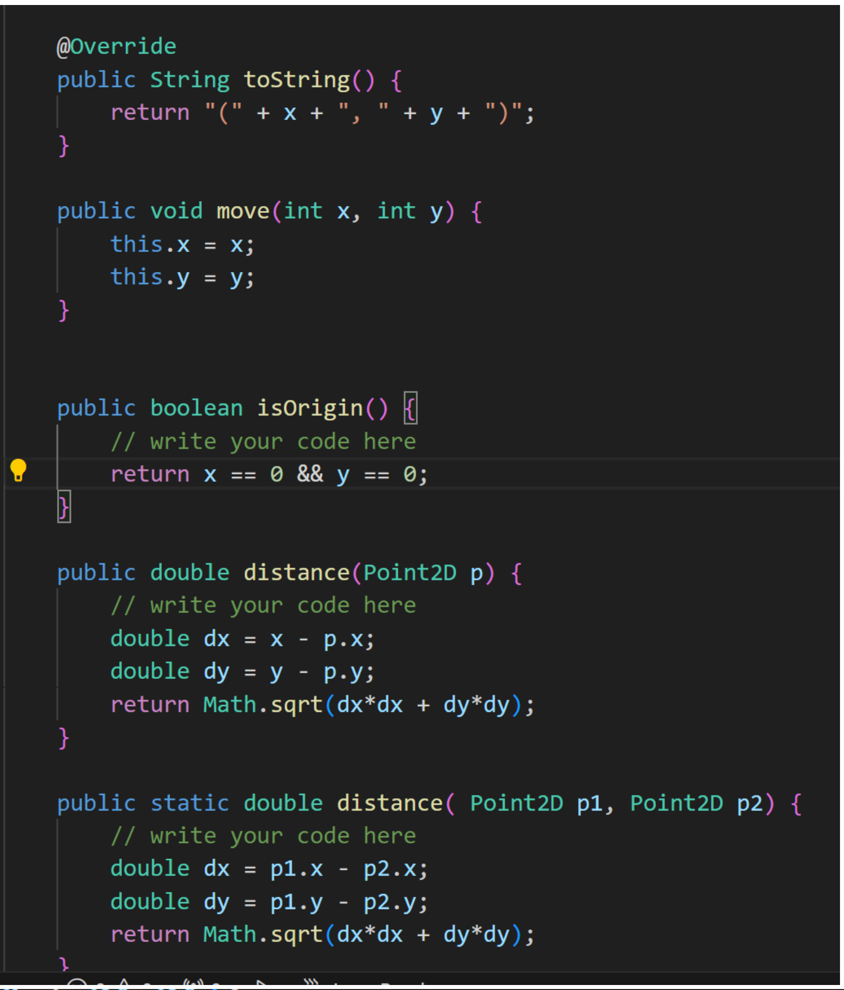
```java
        return p3;
    }

    public double perimeter() {
        double side1 = p1.distance(p2);
        double side2 = p2.distance(p3);
        double side3 = p3.distance(p1);
        return side1 + side2 + side3;
    }

    public double area() {
        double side1 = p1.distance(p2);
        double side2 = p2.distance(p3);
        double side3 = p3.distance(p1);
        double semi_perimeter = (side1 + side2 + side3) / 2;
        return Math.sqrt(semi_perimeter * (semi_perimeter - side1) * (semi_perimeter - side2) * (semi_perimeter - side3));
    }
}
```

Point2D

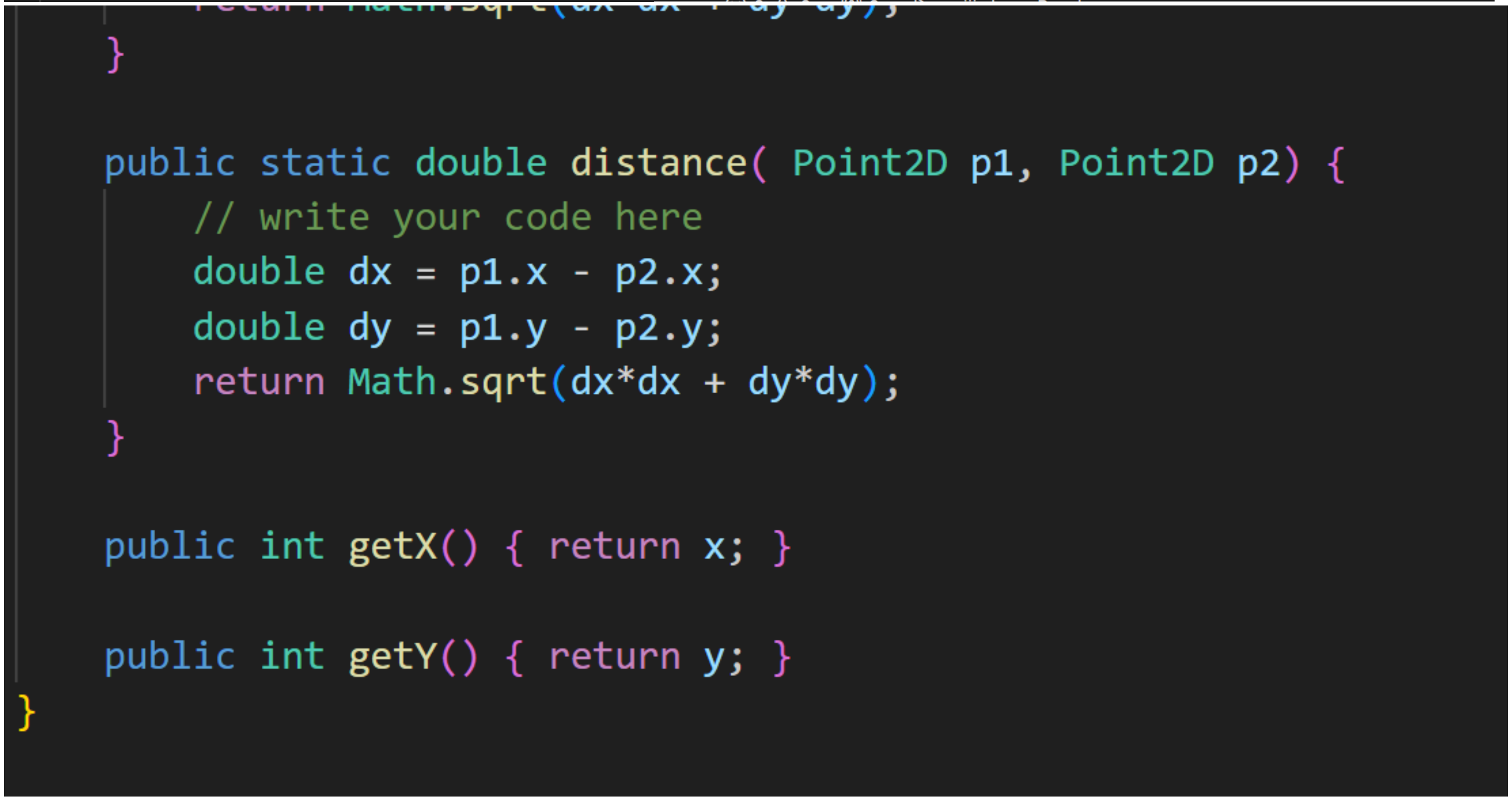


Point2D.java > Point2D > isOrigin()

```java
import java.util.Scanner;

public class Point2D {
    private int x;
    private int y;
    Scanner sc = new Scanner(System.in);
    public Point2D() {
        // write your code here
        this.x = 0;
        this.y = 0;
    }

    public Point2D(int x, int y) {
        // write your code here
        this.x = x;
        this.y = y;
    }

    public Point2D(Point2D p) {
        // write your code here
        this.x = p.x;
        this.y = p.y;
    }

    public void input() {
        try (// write your code here
        Scanner sc = new Scanner(System.in)) {
            x = sc.nextInt();
            y = sc.nextInt();
        }
    }
```

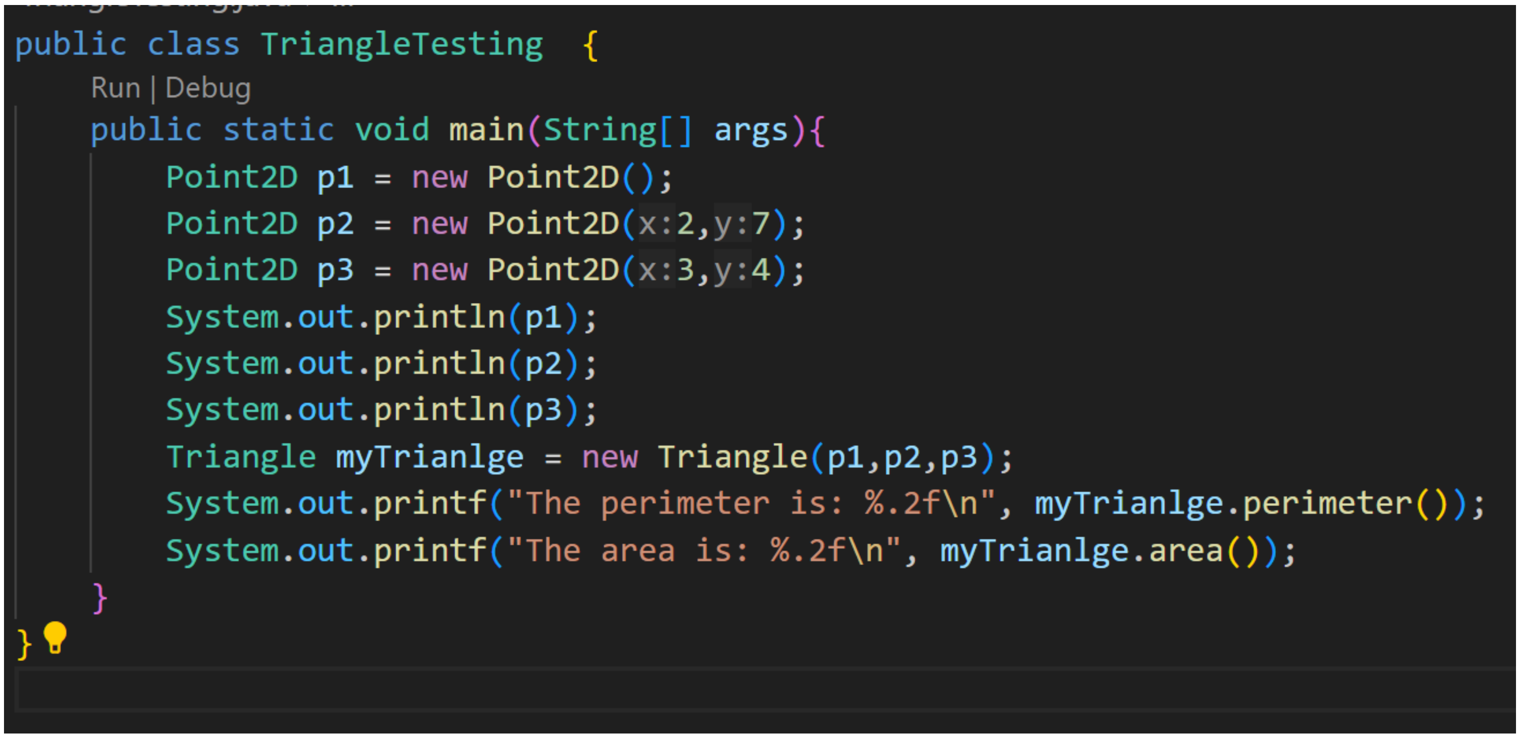


```java
    @Override
    public String toString() {
        return "(" + x + ", " + y + ")";
    }

    public void move(int x, int y) {
        this.x = x;
        this.y = y;
    }


    public boolean isOrigin() {
        // write your code here
        return x == 0 && y == 0;
    }

    public double distance(Point2D p) {
        // write your code here
        double dx = x - p.x;
        double dy = y - p.y;
        return Math.sqrt(dx*dx + dy*dy);
    }

    public static double distance( Point2D p1, Point2D p2) {
        // write your code here
        double dx = p1.x - p2.x;
        double dy = p1.y - p2.y;
        return Math.sqrt(dx*dx + dy*dy);
    }
```



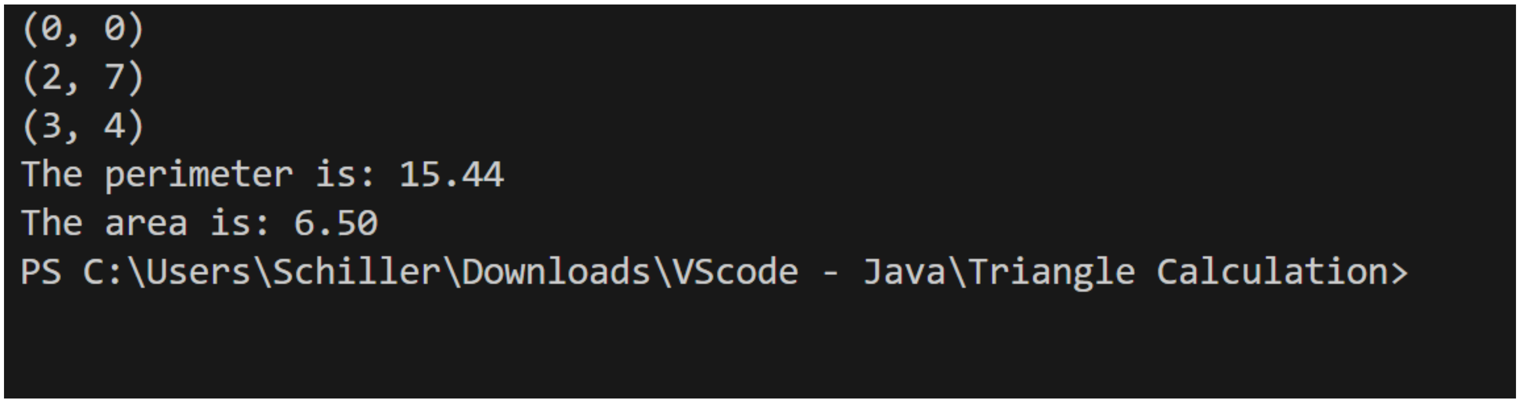
```java
    }

    public static double distance( Point2D p1, Point2D p2) {
        // write your code here
        double dx = p1.x - p2.x;
        double dy = p1.y - p2.y;
        return Math.sqrt(dx*dx + dy*dy);
    }

    public int getX() { return x; }

    public int getY() { return y; }
}
```

TestTriangle



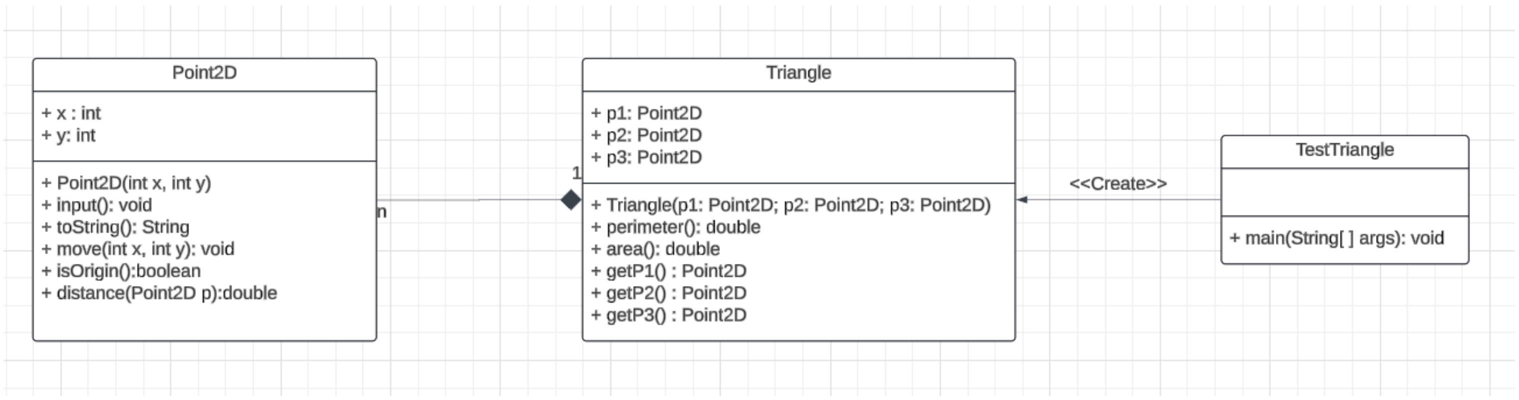
```
public class TriangleTesting  {
    Run | Debug
    public static void main(String[] args){
        Point2D p1 = new Point2D();
        Point2D p2 = new Point2D(x:2,y:7);
        Point2D p3 = new Point2D(x:3,y:4);
        System.out.println(p1);
        System.out.println(p2);
        System.out.println(p3);
        Triangle myTrianlge = new Triangle(p1,p2,p3);
        System.out.printf("The perimeter is: %.2f\n", myTrianlge.perimeter());
        System.out.printf("The area is: %.2f\n", myTrianlge.area());
    }
}
```

Output



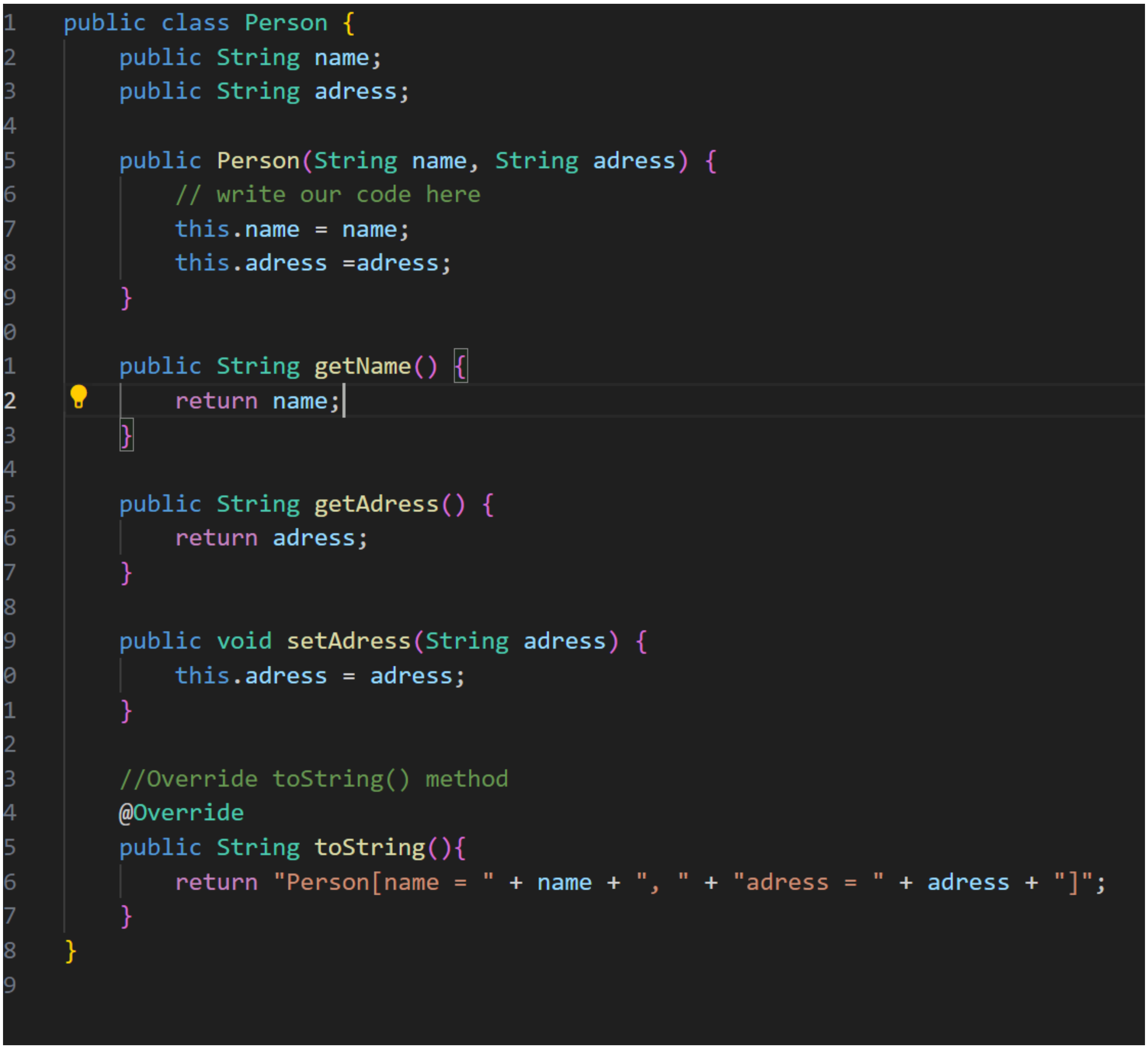
```
(0, 0)
(2, 7)
(3, 4)
The perimeter is: 15.44
The area is: 6.50
PS C:\Users\Schiller\Downloads\VScode - Java\Triangle Calculation>
```

UML



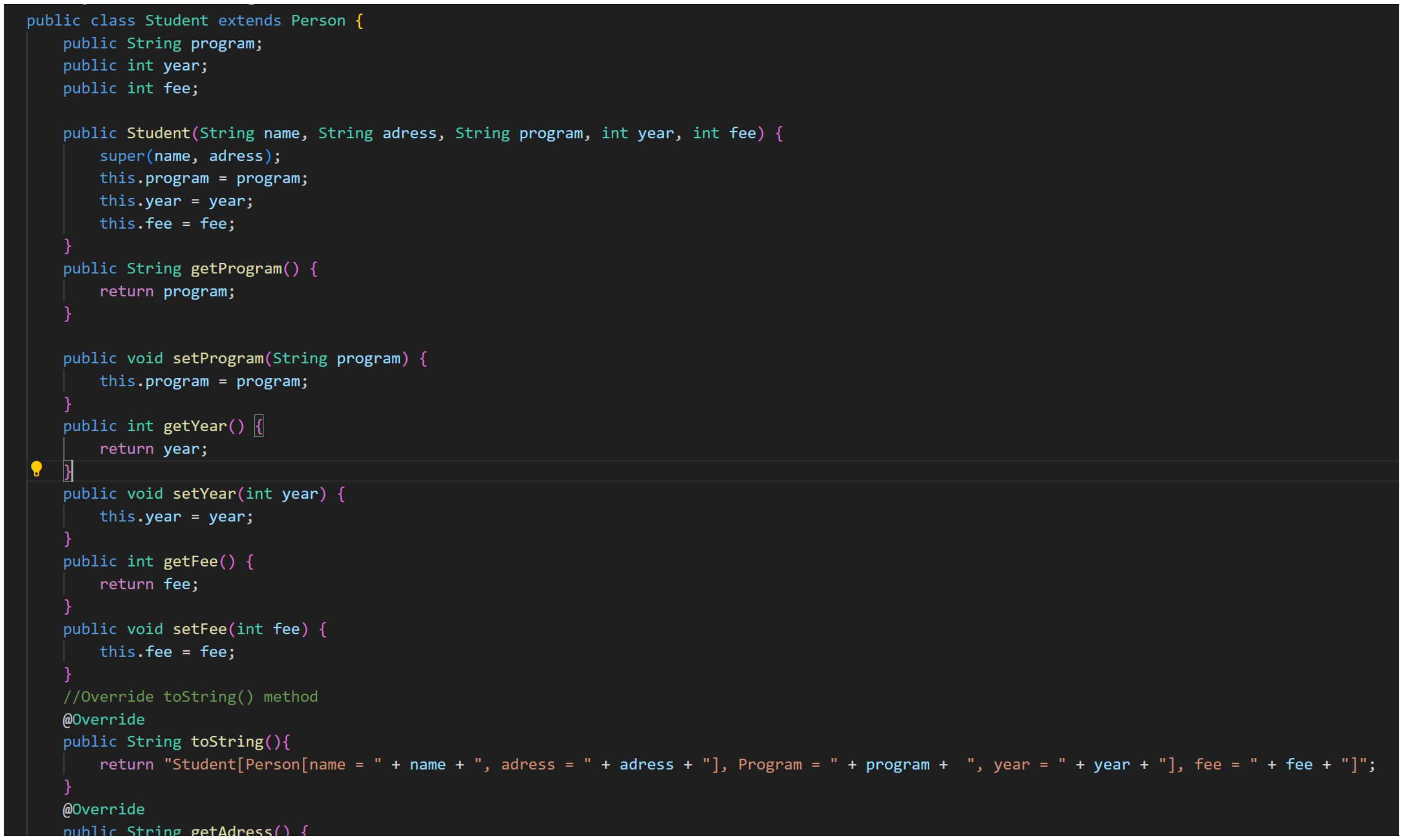
Task 3: Inheritance for Student and Staff

Person

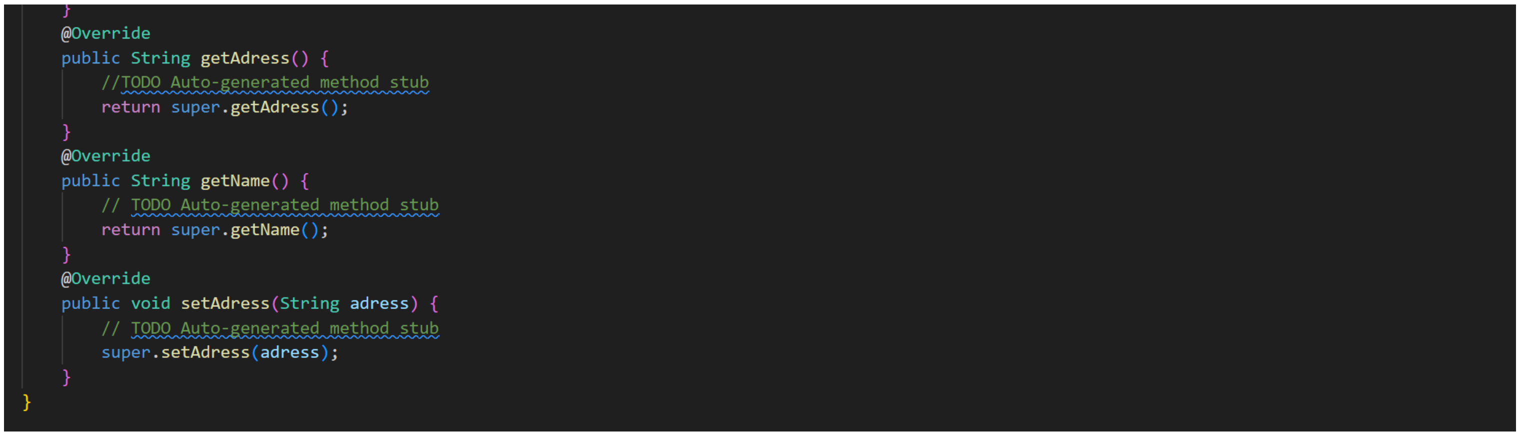


```java
public class Person {
    public String name;
    public String adress;

    public Person(String name, String adress) {
        // write our code here
        this.name = name;
        this.adress =adress;
    }

    public String getName() {
        return name;
    }

    public String getAdress() {
        return adress;
    }

    public void setAdress(String adress) {
        this.adress = adress;
    }

    //Override toString() method
    @Override
    public String toString(){
        return "Person[name = " + name + ", " + "adress = " + adress + "]";
    }
}

```

Student

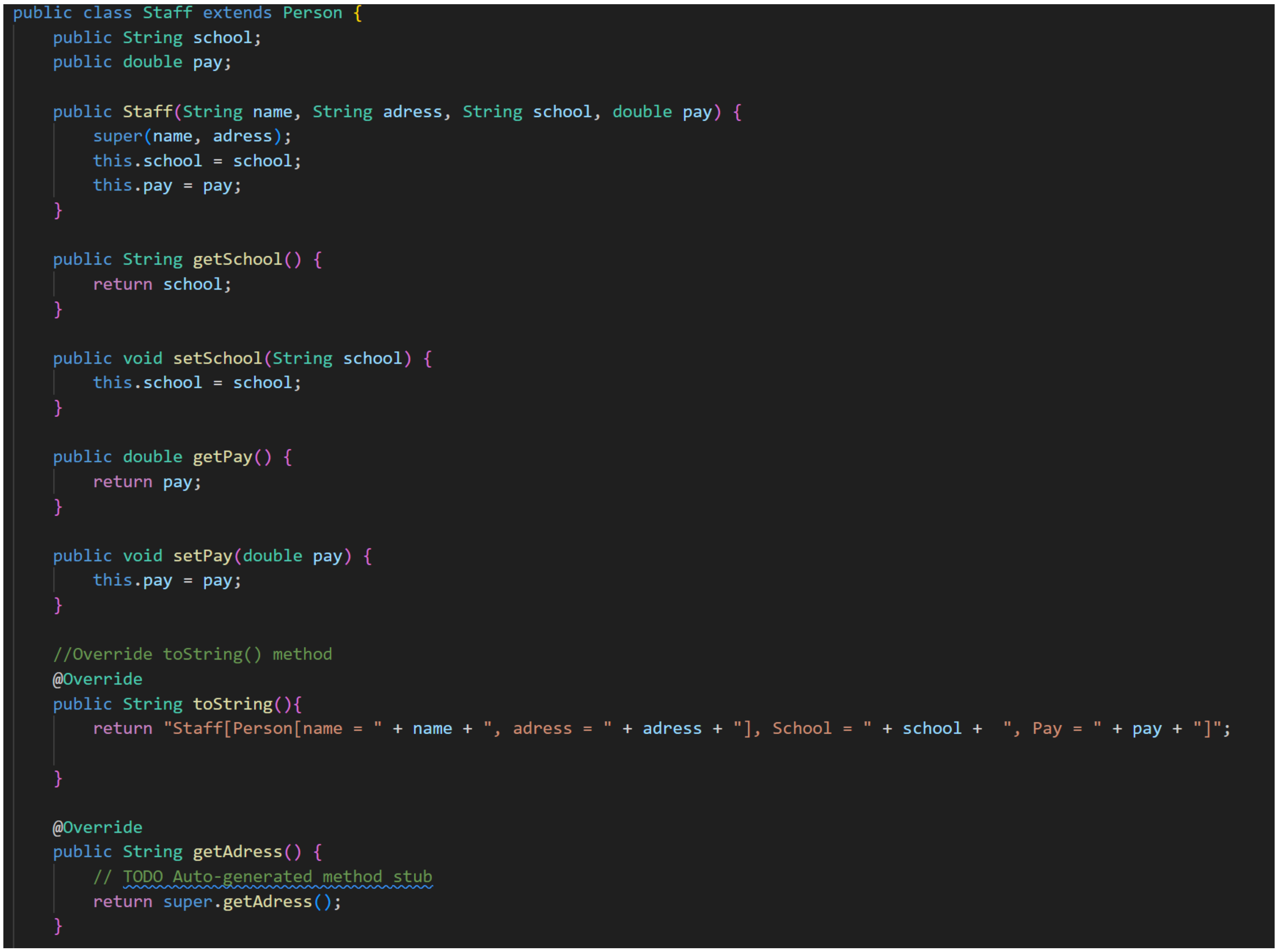


```java
public class Student extends Person {
    public String program;
    public int year;
    public int fee;

    public Student(String name, String adress, String program, int year, int fee) {
        super(name, adress);
        this.program = program;
        this.year = year;
        this.fee = fee;
    }
    public String getProgram() {
        return program;
    }

    public void setProgram(String program) {
        this.program = program;
    }
    public int getYear() {
        return year;
    }
    public void setYear(int year) {
        this.year = year;
    }
    public int getFee() {
        return fee;
    }
    public void setFee(int fee) {
        this.fee = fee;
    }
    //Override toString() method
    @Override
    public String toString(){
        return "Student[Person[name = " + name + ", adress = " + adress + "], Program = " + program +  ", year = " + year + "], fee = " + fee + "]";
    }
    @Override
    public String getAdress() {
```

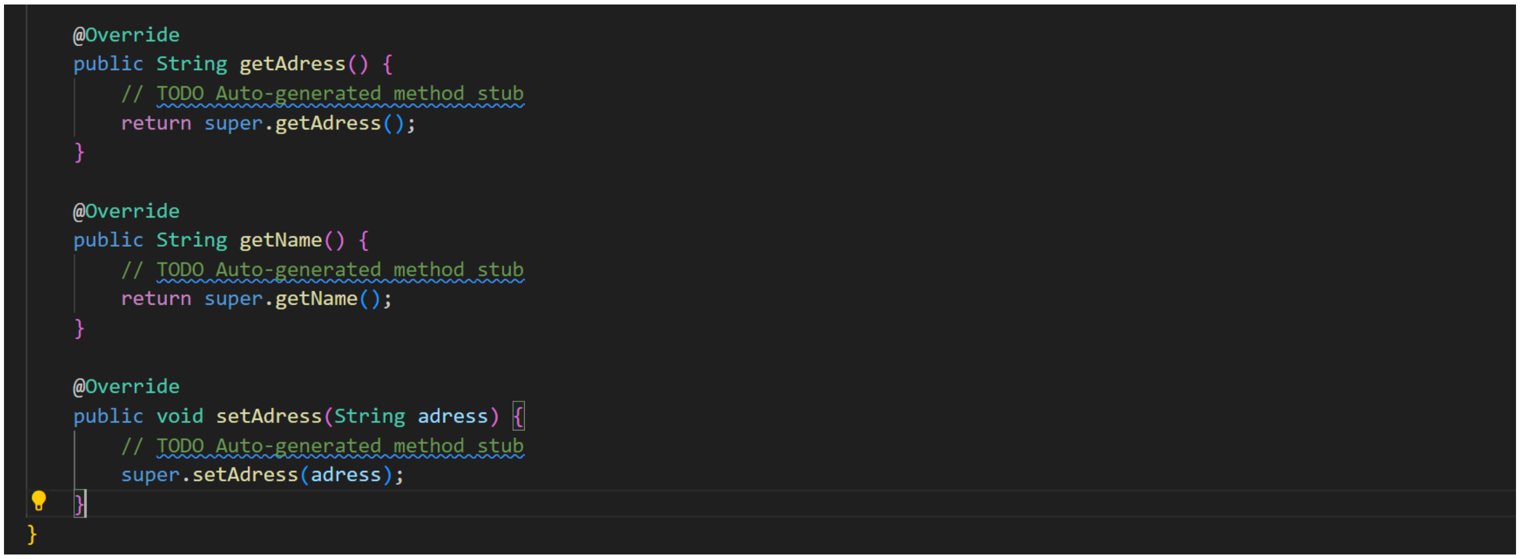


```java
    }
    @Override
    public String getAdress() {
        //TODO Auto-generated method stub
        return super.getAdress();
    }
    @Override
    public String getName() {
        // TODO Auto-generated method stub
        return super.getName();
    }
    @Override
    public void setAdress(String adress) {
        // TODO Auto-generated method stub
        super.setAdress(adress);
    }
}
```

Staff

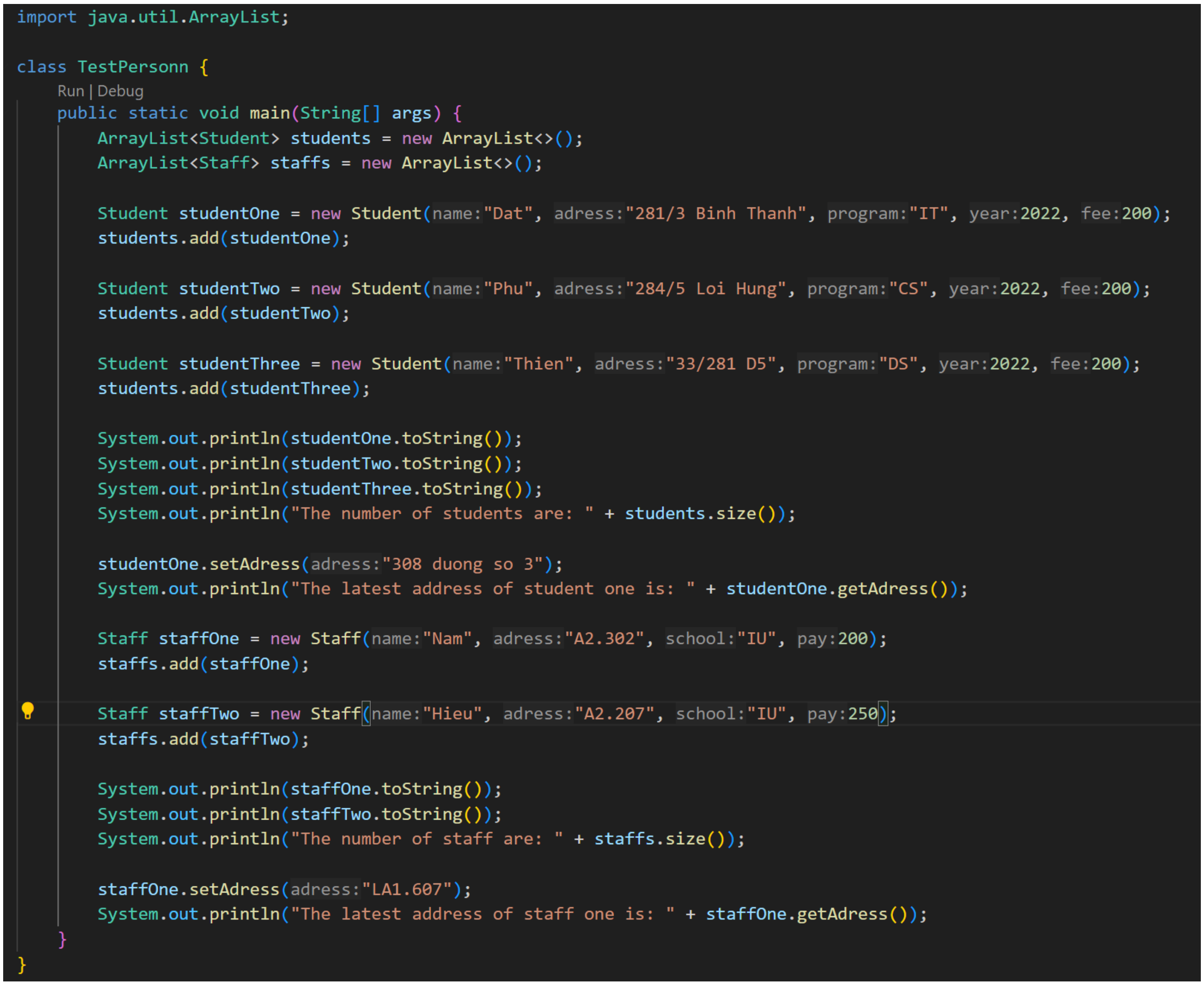


```java
public class Staff extends Person {
    public String school;
    public double pay;

    public Staff(String name, String adress, String school, double pay) {
        super(name, adress);
        this.school = school;
        this.pay = pay;
    }

    public String getSchool() {
        return school;
    }

    public void setSchool(String school) {
        this.school = school;
    }

    public double getPay() {
        return pay;
    }

    public void setPay(double pay) {
        this.pay = pay;
    }

    //Override toString() method
    @Override
    public String toString(){
        return "Staff[Person[name = " + name + ", adress = " + adress + "], School = " + school +  ", Pay = " + pay + "]";

    }

    @Override
    public String getAdress() {
        // TODO Auto-generated method stub
        return super.getAdress();
    }
```



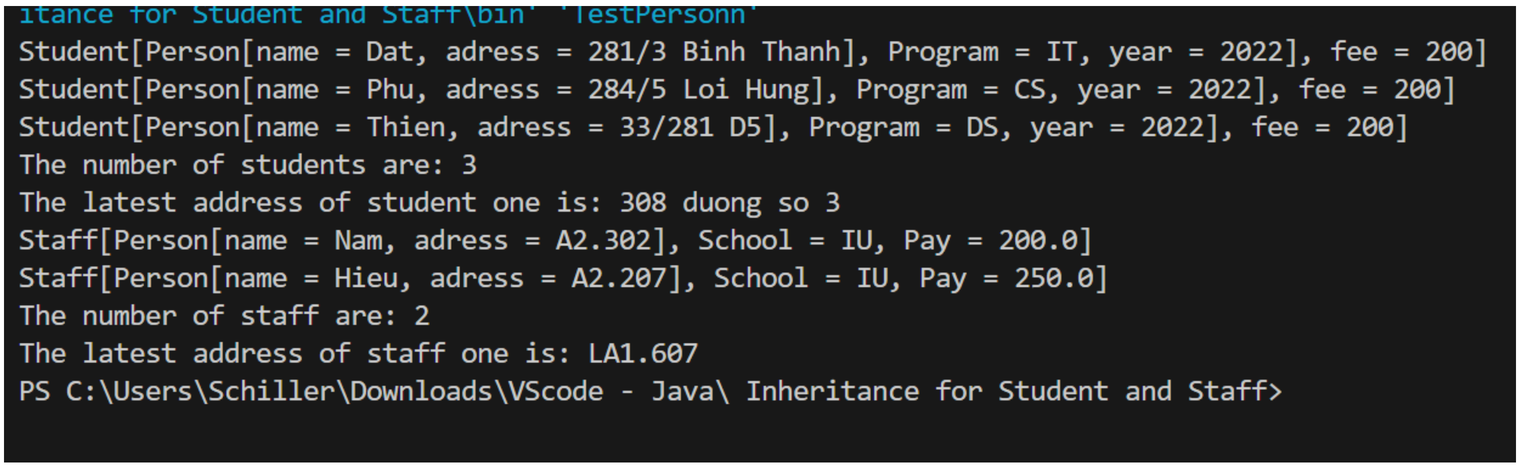
```java
    @Override
    public String getAdress() {
        // TODO Auto-generated method stub
        return super.getAdress();
    }

    @Override
    public String getName() {
        // TODO Auto-generated method stub
        return super.getName();
    }

    @Override
    public void setAdress(String adress) {
        // TODO Auto-generated method stub
        super.setAdress(adress);
    }
}
```

TestInheritance



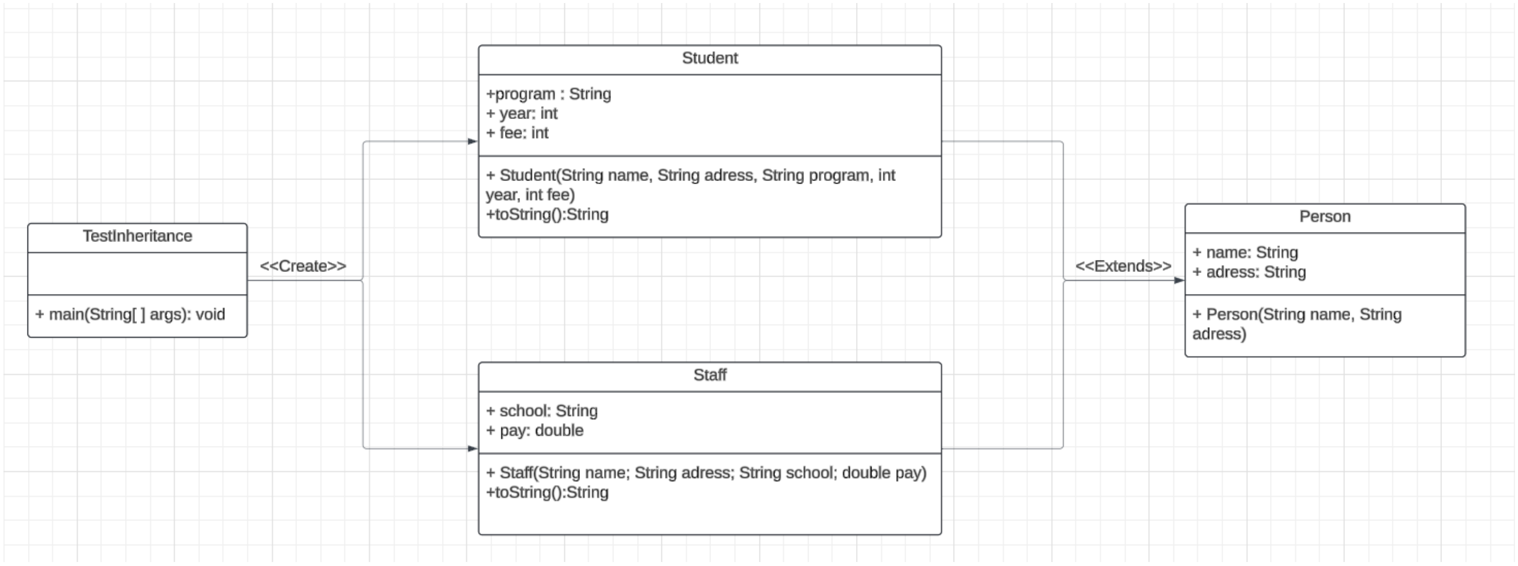
```java
import java.util.ArrayList;

class TestPersonn {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<Student> students = new ArrayList<>();
        ArrayList<Staff> staffs = new ArrayList<>();

        Student studentOne = new Student(name:"Dat", adress:"281/3 Binh Thanh", program:"IT", year:2022, fee:200);
        students.add(studentOne);

        Student studentTwo = new Student(name:"Phu", adress:"284/5 Loi Hung", program:"CS", year:2022, fee:200);
        students.add(studentTwo);

        Student studentThree = new Student(name:"Thien", adress:"33/281 D5", program:"DS", year:2022, fee:200);
        students.add(studentThree);

        System.out.println(studentOne.toString());
        System.out.println(studentTwo.toString());
        System.out.println(studentThree.toString());
        System.out.println("The number of students are: " + students.size());

        studentOne.setAdress(adress:"308 duong so 3");
        System.out.println("The latest address of student one is: " + studentOne.getAdress());

        Staff staffOne = new Staff(name:"Nam", adress:"A2.302", school:"IU", pay:200);
        staffs.add(staffOne);

        Staff staffTwo = new Staff(name:"Hieu", adress:"A2.207", school:"IU", pay:250);
        staffs.add(staffTwo);

        System.out.println(staffOne.toString());
        System.out.println(staffTwo.toString());
        System.out.println("The number of staff are: " + staffs.size());

        staffOne.setAdress(adress:"LA1.607");
        System.out.println("The latest address of staff one is: " + staffOne.getAdress());
    }
}
```

Output



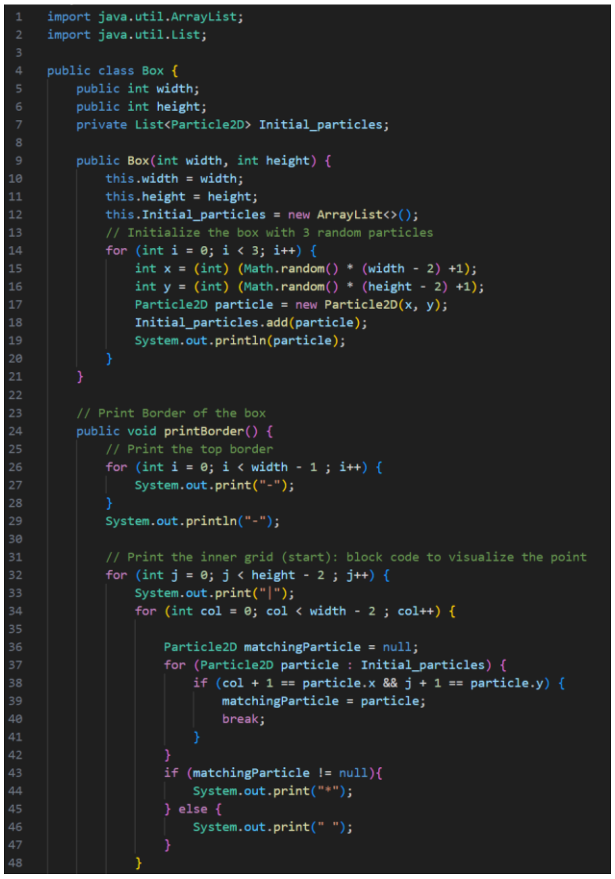
```
itance for Student and Staff\bin' 'TestPersonn'
Student[Person[name = Dat, adress = 281/3 Binh Thanh], Program = IT, year = 2022], fee = 200]
Student[Person[name = Phu, adress = 284/5 Loi Hung], Program = CS, year = 2022], fee = 200]
Student[Person[name = Thien, adress = 33/281 D5], Program = DS, year = 2022], fee = 200]
The number of students are: 3
The latest address of student one is: 308 duong so 3
Staff[Person[name = Nam, adress = A2.302], School = IU, Pay = 200.0]
Staff[Person[name = Hieu, adress = A2.207], School = IU, Pay = 250.0]
The number of staff are: 2
The latest address of staff one is: LA1.607
PS C:\Users\Schiller\Downloads\VScode - Java\ Inheritance for Student and Staff>
```

UML

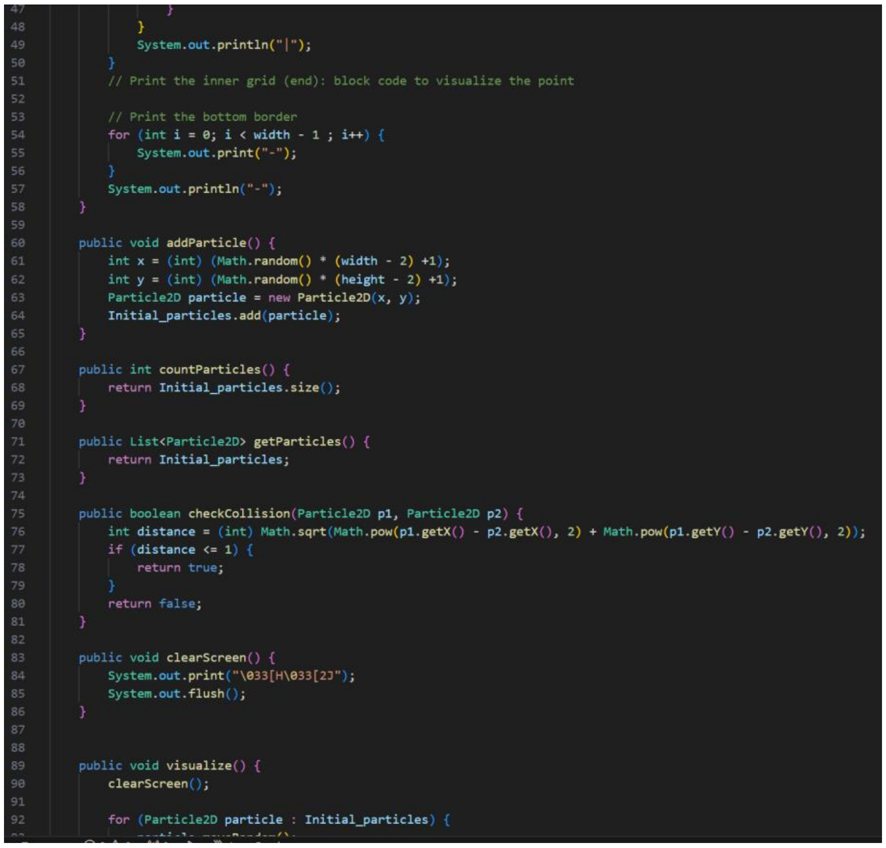

Student
+program : String
+ year: int
+ fee: int
+ Student(String name, String adress, String program, int year, int fee)
+toString():String
TestInheritance
+ main(String[ ] args): void
<<Create>>
<<Extends>>
Person
+ name: String
+ adress: String
+ Person(String name, String adress)
Staff
+ school: String
+ pay: double
+ Staff(String name; String adress; String school; double pay)
+toString():String

Task 4: Particle Behaviour in Box Simulation

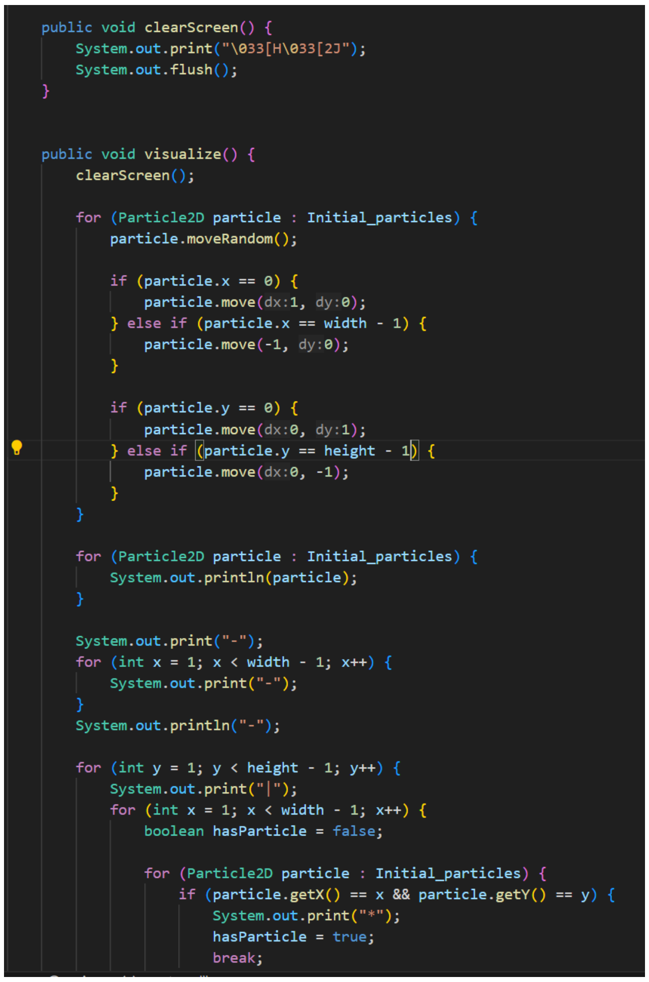
Box



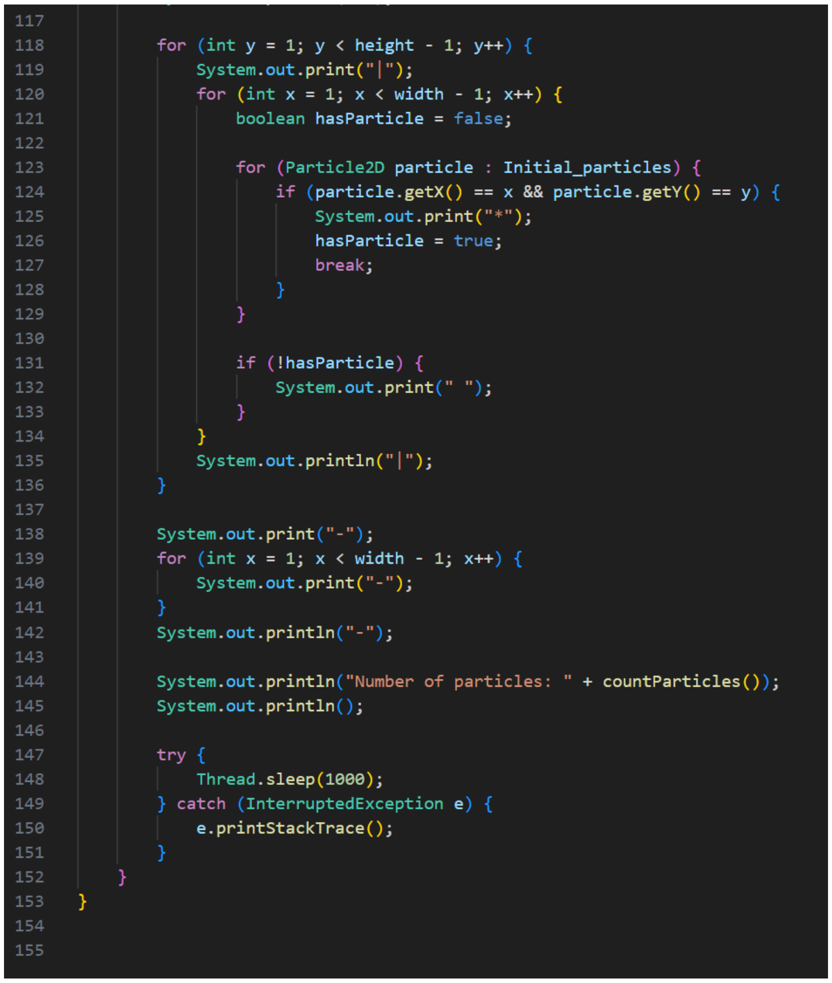
```java
import java.util.ArrayList;
import java.util.List;

public class Box {
    public int width;
    public int height;
    private List<Particle2D> Initial_particles;

    public Box(int width, int height) {
        this.width = width;
        this.height = height;
        this.Initial_particles = new ArrayList<>();
        // Initialize the box with 3 random particles
        for (int i = 0; i < 3; i++) {
            int x = (int) (Math.random() * (width - 2) +1);
            int y = (int) (Math.random() * (height - 2) +1);
            Particle2D particle = new Particle2D(x, y);
            Initial_particles.add(particle);
            System.out.println(particle);
        }
    }

    // Print Border of the box
    public void printBorder() {
        // Print the top border
        for (int i = 0; i < width - 1 ; i++) {
            System.out.print("-");
        }
        System.out.println("-");

        // Print the inner grid (start): block code to visualize the point
        for (int j = 0; j < height - 2 ; j++) {
            System.out.print("|");
            for (int col = 0; col < width - 2 ; col++) {

                Particle2D matchingParticle = null;
                for (Particle2D particle : Initial_particles) {
                    if (col + 1 == particle.x && j + 1 == particle.y) {
                        matchingParticle = particle;
                        break;
                    }
                }
                if (matchingParticle != null){
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
```



```java
            System.out.println("|");
        }
        // Print the inner grid (end): block code to visualize the point

        // Print the bottom border
        for (int i = 0; i < width - 1 ; i++) {
            System.out.print("-");
        }
        System.out.println("-");
    }

    public void addParticle() {
        int x = (int) (Math.random() * (width - 2) +1);
        int y = (int) (Math.random() * (height - 2) +1);
        Particle2D particle = new Particle2D(x, y);
        Initial_particles.add(particle);
    }

    public int countParticles() {
        return Initial_particles.size();
    }

    public List<Particle2D> getParticles() {
        return Initial_particles;
    }

    public boolean checkCollision(Particle2D p1, Particle2D p2) {
        int distance = (int) Math.sqrt(Math.pow(p1.getX() - p2.getX(), 2) + Math.pow(p1.getY() - p2.getY(), 2));
        if (distance <= 1) {
            return true;
        }
        return false;
    }

    public void clearScreen() {
        System.out.print("\033[H\033[2J");
        System.out.flush();
    }


    public void visualize() {
        clearScreen();

        for (Particle2D particle : Initial_particles) {
```

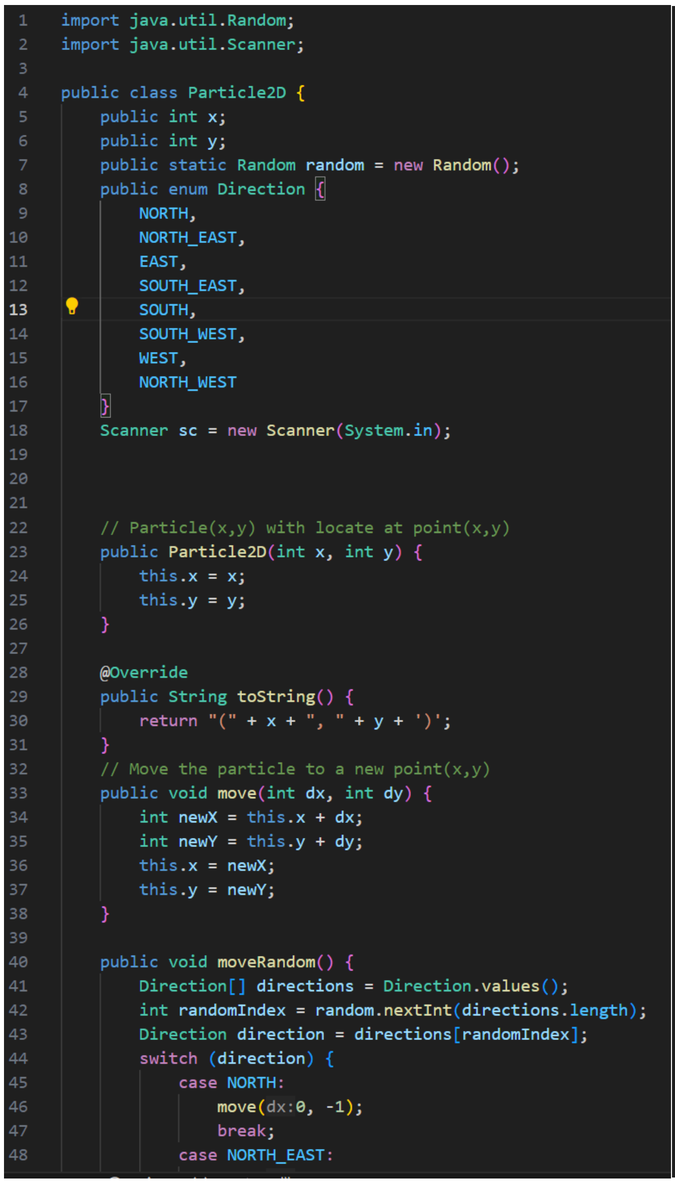


```java
    public void clearScreen() {
        System.out.print("\033[H\033[2J");
        System.out.flush();
    }


    public void visualize() {
        clearScreen();

        for (Particle2D particle : Initial_particles) {
            particle.moveRandom();

            if (particle.x == 0) {
                particle.move(dx:1, dy:0);
            } else if (particle.x == width - 1) {
                particle.move(-1, dy:0);
            }

            if (particle.y == 0) {
                particle.move(dx:0, dy:1);
            } else if (particle.y == height - 1) {
                particle.move(dx:0, -1);
            }
        }

        for (Particle2D particle : Initial_particles) {
            System.out.println(particle);
        }

        System.out.print("-");
        for (int x = 1; x < width - 1; x++) {
            System.out.print("-");
        }
        System.out.println("-");

        for (int y = 1; y < height - 1; y++) {
            System.out.print("|");
            for (int x = 1; x < width - 1; x++) {
                boolean hasParticle = false;

                for (Particle2D particle : Initial_particles) {
                    if (particle.getX() == x && particle.getY() == y) {
                        System.out.print("*");
                        hasParticle = true;
                        break;
```

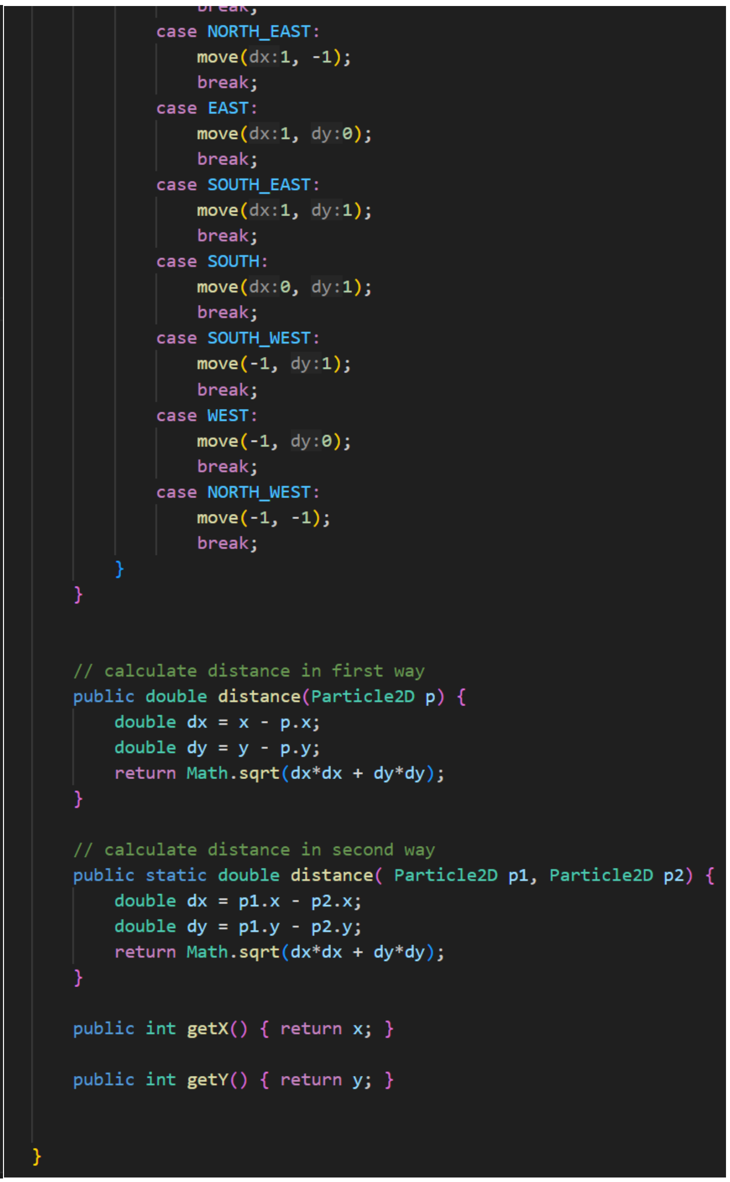


```java
        for (int y = 1; y < height - 1; y++) {
            System.out.print("|");
            for (int x = 1; x < width - 1; x++) {
                boolean hasParticle = false;

                for (Particle2D particle : Initial_particles) {
                    if (particle.getX() == x && particle.getY() == y) {
                        System.out.print("*");
                        hasParticle = true;
                        break;
                    }
                }

                if (!hasParticle) {
                    System.out.print(" ");
                }
            }
            System.out.println("|");
        }

        System.out.print("-");
        for (int x = 1; x < width - 1; x++) {
            System.out.print("-");
        }
        System.out.println("-");

        System.out.println("Number of particles: " + countParticles());
        System.out.println();

        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Particlle2D

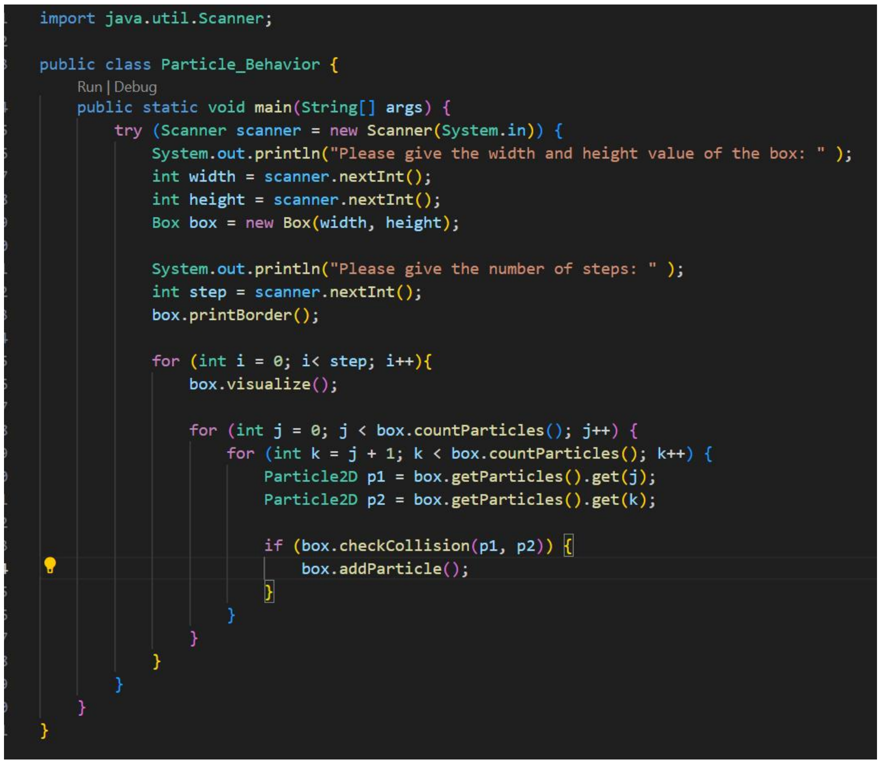


```java
import java.util.Random;
import java.util.Scanner;

public class Particle2D {
    public int x;
    public int y;
    public static Random random = new Random();
    public enum Direction {
        NORTH,
        NORTH_EAST,
        EAST,
        SOUTH_EAST,
        SOUTH,
        SOUTH_WEST,
        WEST,
        NORTH_WEST
    }
    Scanner sc = new Scanner(System.in);



    // Particle(x,y) with locate at point(x,y)
    public Particle2D(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public String toString() {
        return "(" + x + ", " + y + ')';
    }
    // Move the particle to a new point(x,y)
    public void move(int dx, int dy) {
        int newX = this.x + dx;
        int newY = this.y + dy;
        this.x = newX;
        this.y = newY;
    }

    public void moveRandom() {
        Direction[] directions = Direction.values();
        int randomIndex = random.nextInt(directions.length);
        Direction direction = directions[randomIndex];
        switch (direction) {
            case NORTH:
                move(dx:0, -1);
                break;
            case NORTH_EAST:
```



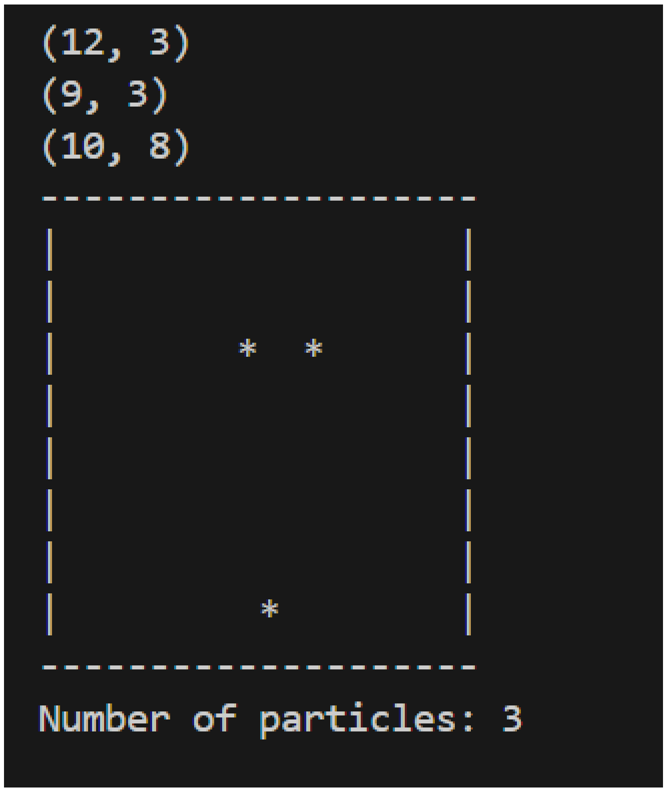
```java
            case NORTH_EAST:
                move(dx:1, -1);
                break;
            case EAST:
                move(dx:1, dy:0);
                break;
            case SOUTH_EAST:
                move(dx:1, dy:1);
                break;
            case SOUTH:
                move(dx:0, dy:1);
                break;
            case SOUTH_WEST:
                move(-1, dy:1);
                break;
            case WEST:
                move(-1, dy:0);
                break;
            case NORTH_WEST:
                move(-1, -1);
                break;
        }
    }


    // calculate distance in first way
    public double distance(Particle2D p) {
        double dx = x - p.x;
        double dy = y - p.y;
        return Math.sqrt(dx*dx + dy*dy);
    }

    // calculate distance in second way
    public static double distance( Particle2D p1, Particle2D p2) {
        double dx = p1.x - p2.x;
        double dy = p1.y - p2.y;
        return Math.sqrt(dx*dx + dy*dy);
    }

    public int getX() { return x; }

    public int getY() { return y; }


}
```

TestBehaviour



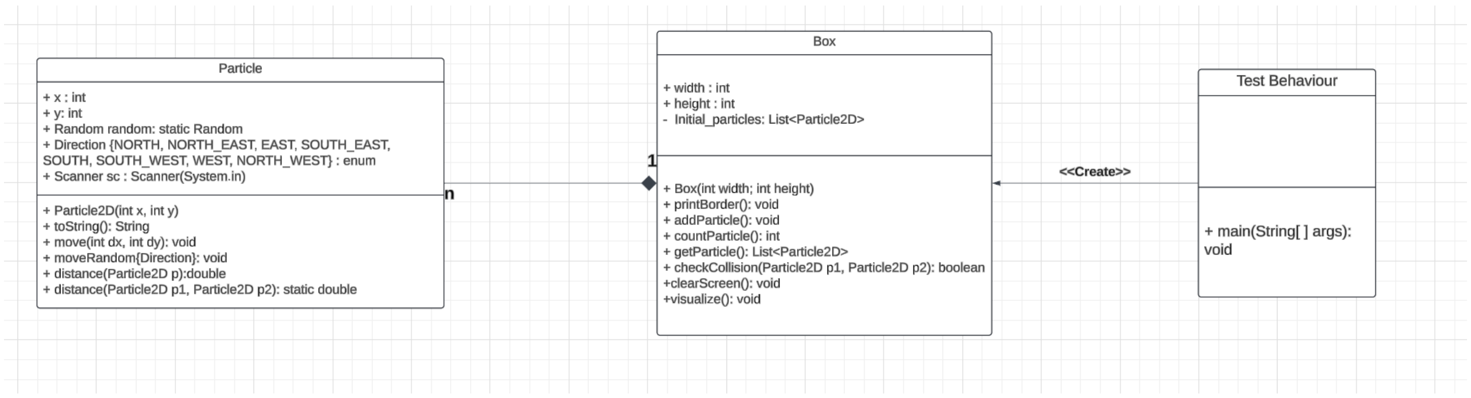
```java
import java.util.Scanner;

public class Particle_Behavior {
    Run | Debug
    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(System.in)) {
            System.out.println("Please give the width and height value of the box: " );
            int width = scanner.nextInt();
            int height = scanner.nextInt();
            Box box = new Box(width, height);

            System.out.println("Please give the number of steps: " );
            int step = scanner.nextInt();
            box.printBorder();

            for (int i = 0; i< step; i++){
                box.visualize();

                for (int j = 0; j < box.countParticles(); j++) {
                    for (int k = j + 1; k < box.countParticles(); k++) {
                        Particle2D p1 = box.getParticles().get(j);
                        Particle2D p2 = box.getParticles().get(k);

                        if (box.checkCollision(p1, p2)) {
                            box.addParticle();
                        }
                    }
                }
            }
        }
    }
}
```

Output



```
(12, 3)
(9, 3)
(10, 8)
--------------------
|                  |
|                  |
|        *  *      |
|                  |
|                  |
|                  |
|                  |
|         *        |
--------------------
Number of particles: 3
```

UML


Particle
+ x : int
+ y: int
+ Random random: static Random
+ Direction {NORTH, NORTH_EAST, EAST, SOUTH_EAST, SOUTH, SOUTH_WEST, WEST, NORTH_WEST} : enum
+ Scanner sc : Scanner(System.in)
+ Particle2D(int x, int y)
+ toString(): String
+ move(int dx, int dy): void
+ moveRandom{Direction}: void
+ distance(Particle2D p):double
+ distance(Particle2D p1, Particle2D p2): static double
n
1
Box
+ width : int
+ height : int
- Initial_particles: List<Particle2D>
+ Box(int width; int height)
+ printBorder(): void
+ addParticle(): void
+ countParticle(): int
+ getParticle(): List<Particle2D>
+ checkCollision(Particle2D p1, Particle2D p2): boolean
+clearScreen(): void
+visualize(): void
<<Create>>
Test Behaviour
+ main(String[ ] args): void