CMPS 12A - Assignment 4

Submission deadline: November 23, 2018 at 11:59 pm

Instructions:

To help you debug your code, we are providing **PlaneTest.java**. This file contains some of the test cases we are using on Stepik for the Plane.java problem. To use the test class, create a class called PlaneTest.java in your Eclipse project and then paste the code from PlaneTest.java into that file.
In order to submit your code, you just need to submit your Plane.java (which includes all of the classes mentioned below) on Stepik. Please note that you are not allowed to use any library other than Math library.

Plane (100 points)

We want to write a Java program to implement classes and methods for geometrical shapes using inheritance and polymorphism.

Following is the skeleton and description of the classes that you need to complete.

```
class Point {
    public Point(double x, double y);
```
Creates an instance of point using its x and y coordinates.

```
    public double getX();
```
Returns the x coordinate of the point.

```
    public double getY();
```
Returns the y coordinate of the point.

```
    public double getDistance(Point point);
```
Returns the Euclidean distance of the point from a given point.
}

```
abstract class Shape {
    public abstract double getArea();
```
It calculates and returns the area of a shape. Note that the method is abstract and all concrete subclasses of shape should implement it.

```
    public abstract double getPerimeter();
```
It calculates and returns the perimeter of a shape. Again, this method is abstract and all concrete subclasses of shape should implement it.
}

```
interface Symmetric {
    Point getPointOfSymmetry();
```
Returns an object of type Point which is the point of symmetry (or center of symmetry) of a shape. This method is abstract and each concrete class that implements the interface should implement it.
}

```
class Triangle extends Shape {
      public Triangle(Point firstPoint, Point secondPoint, Point thirdPoint);
```
Creates a triangle object using 3 points.

```
      public Point getFirstPoint();
```
Returns the first point of the triangle which is the first point given in the constructor.

```
      public Point getSecondPoint();
```
Returns the second point of the triangle which is the second point given in the constructor.

```
      public Point getThirdPoint();
```
Returns the third point of the triangle which is the third point given in the constructor.

```
}
```

```
class Rectangle extends Shape {
      public Rectangle(Point topLeftPoint, double length, double width);
```
Creates a rectangle using the top-left corner of the rectangle and its length and width.

```
      public Point getTopLeftPoint();
```
Returns the top-left corner of the rectangle.

```
      public double getLength();
```
Returns the length of the rectangle.

```
      public double getWidth();
```
Returns the width of the rectangle.

```
}
```

```
class Trapezoid extends Shape {
      public Trapezoid(Point topLeftPoint, Point bottomLeftPoint, double topSide, double bottomSide);
```
Creates a trapezoid using top-left and bottom-left corners as well as the length of the top and bottom sides.

```
      public Point getTopLeftPoint();
```
Returns the top-left corner of the trapezoid.

```
      public Point getBottomLeftPoint();
```
Returns the bottom-left corner of the trapezoid.

```
      public double getTopSide();
```
Returns the length of the top side of the trapezoid.

```
      public double getBottomSide();
```
Returns the length of the bottom side of the trapezoid.

```
}
```

```
class Circle extends Shape implements Symmetric {
      public Circle(Point center, double radius);
```
Creates a circle using its center and radius.

```
      public Point getCenter();
```
Returns center of the circle.

```
      public double getRadius();
```
Returns radius of the circle.

```
}
```

```
class EquilateralTriangle extends Triangle implements Symmetric {
      public EquilateralTriangle(Point topPoint, double side);
```
Creates an equilateral triangle using the top point and its side length. Note that the other two corners (points) have the same 'y' coordinate. Also, getFirstPoint(), getSecondPoint(), and getThirdPoint() methods should return the top, bottom-left, and bottom-right corners, respectively.

```
      public Point getTopPoint();
```
Returns the top corner of the equilateral triangle.

```
      public double getSide();
```
Returns the side length of the equilateral triangle.
```
}
```

```
class Square extends Rectangle implements Symmetric {
      public Square(Point topLeft, double side);
```
Creates a square using the top-left corner and its side length.

```
      public double getSide();
```
Returns the side length of the square.
```
}
```

```
public class Plane {
      public Plane();
```
Creates an empty plane.

```
      public Shape[] getShape();
```
Returns an array of shape objects on the plane. The order of objects is not important.

```
      Public void addShape(Shape shape);
```
Takes a shape object and adds it to the array of shapes on the plane. (Remember from lab 6?)

```
      public double getSumOfAreas();
```
Calculates the sum of the areas of all the shapes on the plane. If there is any overlapped area, it will be counted twice.

```
      public double getSumOfPerimeters();
```
Calculates the sum of the perimeters of all the shapes on the plane.

```
      public Point getCenterOfPointOfSymmetries();
```
Returns the center of all points of symmetry of symmetric shapes. Center of symmetry of a set of points is a point with x coordinate being the average of x coordinate of the points, and y coordinate being the average of y coordinates of the points. If there is no symmetric shapes on the plane, returns null. You might need to read about 'instanceof' operator to complete this method (Suggested link).

You can implement other methods in the above-mentioned classes. Except from the methods listed above, all of the extra methods and all class fields should be defined as **private**. You have to use concepts from **inheritance** and **polymorphism** whenever applicable. Your mark for this question is not solely based on the correct output, but it is also based on the correct usage of inheritance and polymorphism.

Bonus Question: SnakeLadder (up to extra 20 points)

In this question we want to simulate snakes and ladders game. Here are some of the rules:

- The game has 2 to 4 players.
- If a player wants to enter a cell occupied by another player, the player attempting to enter the cell will be moved to the first cell. Note that we can have more that one player in the first cell.
- Players take turn to roll the dice. Then, they move their piece forward the number of spaces shown on the dice.
- If the player lands at the bottom of a ladder, it must move up to the top of the ladder.
- If the player lands on the head of a snake, it must slide down to the bottom of the snake.
- The first player to get to the last cell of the board is the winner. But there's a twist! If the player rolls too high, the piece "bounces" off the last cell and moves back to the first cell. A player can only win by rolling the exact number needed to land on the last cell.
- For simplicity, you can assume that we do not have chained snakes and ladders in the game.

You can find simulation of Snakes & Ladders game (or Chutes & Ladders) Here. Below is the skeleton of the classes you need to implement:

```
class Game {
      public Game(Board board, Player[] players);
```
Creates a game with the board and players passed to constructor.

```
      public Game(Player[] players);
```
Creates a game with a 10-by-10 board and the players which are passed to it.

```
      public Player currentPlayer();
```
Specifies the current player.

```
      public void addPlayer(Player p);
```
Adds a player to the list of players and the player will be placed in the first cell.

```
      public boolean winner();
```
Determines whether the current player has reached the last cell.

```
      public void movePlayer(int n);
```
Moves current player 'n' cells forward.

```
      public boolean play(int moveCount);
```
This is the main method in your game, which moves the current player to the correct cell and also checks whether the game is finished or not. If found a winner, returns true, otherwise returns false.

```
      public Board getBoard();
```
Returns board of the game.

```
}
```

```
class Player{
      public Player(String name);
```
This is the constructor of the player class.

```
      public void setPosition(int position);
```
Moves the player to the specified position.

```
      public int getPosition();
```

Returns position of the player.

```
        public String getName();
```
Returns player's name.

```
        public String toString();
```
Returns "Name @ cell-number". e.g: narges @ 12.

```
}
```

```
class Cell {
        public Cell(int number);
```
Creates a cell and assigns its number.

```
        public void setOccupied(boolean occupied);
```
Fills or releases the cell.

```
        public boolean isOccupied();
```
Determines whether this cell is occupied.

```
        public Ladder getLadder();
```
Returns the Ladder which is in the current cell. Returns null if there is no Ladder with start in this cell.

```
        public Snake getSnake();
```
Returns the Snake which is in the current cell. Returns null if there is no Snake with head in this cell.

```
        public void setLadder(Ladder ladder);
```
Sets a Ladder with start in the current cell.

```
        public void setSnake(Snake snake);
```
Returns the Snake with head in the current cell.

```
        public int getNumber();
```
Returns number of the cell.

```
}
```

```
class Board {
        public Board(int n);
```
Creates an n-by-n board.

```
        public void setCellToLadder(int startPosition, int endPosition);
```
Puts a ladder on the map of the game. You need to associate the ladder with the cell at startPosition.

```
        public void setCellToSnake(int headPosition, int tailPosition);
```
Puts a snake on the map of the game. You need to associate the snake with the cell at headPosition.

```
        public Cell[] getCells();
```
Returns all cells of the board.

```
}
```

```
class Snake {
        public Snake(int headPosition, int tailPosition );
```
Creates a snake with specified location of head and tail.

```
        public int getTail();
```
Returns the position of the snake's tail.

```
        public String toString();
```
Returns: head – tail. e.g: 20 – 11.

```
}
```

```
class Ladder {
      public Ladder (int startPosition, int endPosition );
```
Creates a ladder which connects cells with specified location of start and end of the ladder.

```
      public int getTop();
```
Returns position of the top (end) of the ladder.

```
      public String toString();
```
Returns: start – end.. e .g: 10 – 18.

```
}
```

Here is a sample main method to test your program (we are not releasing the full test class for this bonus question):

```
public static void main(String[] args){
      Player[] players = new Player[3];
      players [0] = new Player("abc");
      players [1] = new Player("def");
      players [2] = new Player("ghi");
      Board myBoard = new Board(5);
      myBoard.setCellToLadder(7, 17);
      myBoard.setCellToLadder(5, 14);
      myBoard.setCellToSnake(19, 8);
      myBoard.setCellToSnake(16, 4);
      Game game = new Game(myBoard, players);
      Player p = new Player(“jkl");
      game.addPlayer(p);
      boolean b = false;
      Random random = new Random();
      int n;
      while (!b) {
            n = random.nextInt(6) + 1;
            b = game.play(n);
      }
      System.out.println(game.currentPlayer().getName());
}
```

Again, you can implement other methods in the classes mentioned above. Except from the methods listed, all of the extra methods and all class fields should be defined as **private**. Do not submit the sample main method given above on Stepik.