CMPS 11 - Assignment 3

Submission deadline: February 16th, 2018 at 11:59 pm

Bingo.java (60 points)

We want to write a Java program to simulate a bingo game. Players use cards that feature five columns of five cells each, with every cell containing a number between 1 to 90. In each step of the game a number between 1 and 90 is called randomly and players need to mark the number in their card if they have it. The first player who marks all of the numbers in one row of one of his/her cards is the winner of the game. Following is the skeleton of the classes that you need to complete:

```
public class Bingo {
public Bingo(Player[] players);
```

Creates an object of Bingo game by taking an array of players.

```
public String play(int number);
```

This is the main method for playing the game. It takes a number (the number that is called) and marks the cells with that number in all players' cards. It also returns the name of the winner, if any. If there is no winner, it return an empty string. If there is more than one winner, it returns the name of all winners as one string, separated by space. For example, if player1, player2, and player3 win the game in one turn, it returns "player1 player2 player3".

```
}
class Player {
public Player(String name, Card[] cards);
```

Creates a player by taking the name of the player and an array of bingo cards for the player.

```
public String getName();
```

Returns the name of the player.

```
public Card[] getCards();
```

Returns player's bingo cards.

```
public boolean isWinner();
```

Checks if the player is a winner.

```
public void markNumber(int number);
```

Takes a number and marks that number in all bingo cards of the player.

```
}
class Card {
public Card(int[][] numbers);
```

Uses an array of 5 by 5 and fills the bingo card by those numbers.

```
public int getNumber(int Row, int Column);
```

Returns the number in the cell at row and column of the bingo card.

```
public boolean isMarked(int row, int column);
```

If the cell at row and column of the card is marked, returns true, otherwise, returns false.

```
public void markNumber(int number);
```

Takes a number and if the number exists in the card, marks that cell of the card.

```
}
```

Following is a sample main method to test the program:

```
public static void main(String[] args){
int[][] numbers1 = {{10, 30, 45, 66, 82}, {3, 25, 11, 63, 78},
{22, 4, 13, 46, 90}, {5, 23, 12, 6, 85}, {1, 88, 67, 2, 44}};
Card[] cards1 = new Card[1];
Cards1[0] = new Card(numbers1);
int[][] numbers2 = {{11, 31, 46, 67, 83}, {4, 26, 12, 64, 79},
{23, 5, 14, 47, 90}, {6, 24, 13, 7, 86}, {2, 89, 68, 3, 45}};
Card[] cards2 = new Card[1];
cards2[0] = new Card(numbers2);
Player[] players = new Player[2];
players[0] = new Player("Player1", cards1);
Players[1] = new Player("Player2", cards2);
Bingo bingo = new Bingo(players);
Random random = new Random();
String winner = "";
while(winners.equals("")){
     int number = random.nextInt(90) + 1;
     winners = bingo.play(number);
}
System.out.println(winners);
}
```

Path.java (40 points)

In this question we want to implement a class to simulate a 2D path composed of multiple 2D points. Here is the skeleton of the class:

```
class Path{
Path();
```

Creates an empty path (without any point). The length of an empty path is zero.

```
Path addPoint(int x, int y);
```

Adds a 2D point with (x, y) dimensions to the end of the path and returns the updated path object. Remember that you do not need to create a new path, you just need to update the path by adding another point and returning the updated path. With this technique you can add multiple points to a path in one line: `myPath.addPoint(7, 2).addPoint(2, 6);`

```
String getPoint(int i);
```

Returns one string in the format of "(x, y)" which x and y are dimensions of i-th point in the path.

```
int numOfPoints();
```

Returns the number of points in the path.

```
boolean removePoint(int i);
```

Removes i-th point from the path. Indexing of the points on the path starts from zero. If the i-th point does not exist, it returns false, otherwise, returns true.

```
void addPath(Path p);
```

Adds path p to the current path by adding all the points of path p to the end of the current path object by keeping the order of the points in path p.

```
double getLength();
```

Returns the length of the path by summing all the distance between any two consecutive points.

```
double getDistance();
```

Returns the distance between the first and last point on the path.

```
boolean isLonger(Path p);
```

Returns true if length of the current path object is greater than the length of the the object passed as an argument.

```
}
```