Problem 1 Polygon

In this problem, you have to create two classes: Point and Polygon. The polygon class contains an array of points. The details of class Point and Polygon are given below:

1.a. **`Point.`** Write a data type Point that implements the following API:

`public class Point`	
	`Point(int x, int y)`
`double`	`distance(Point p)`
`Point`	`midPoint(Point p)`
`int`	`getX()`
`int`	`getY()`
`String`	`toString()`

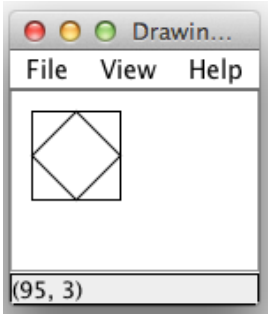
Hint. Given two points p1(x1, y1) and p2(x2, y2), the distance between p1 and p2 is equal to: $\sqrt{(x1-x2)^2+(y1-y2)^2}$. The midpoint of p1 and p2 has coordinates: $\left(\frac{x1+x2}{2}, \frac{y1+y2}{2}\right)$

1.b. **`Polygon.`** Write a data type `Polygon` that implements the following API:

`public class Polygon`	
	// Constructor. Create an array of points with size 1000, i.e., nb of points can be between 0 and 1000. `Polygon()`
`void`	`addPoint(Point p)` *// add a point to the polygon*
`void`	*// Draw the polygon.* *// Hint. You may use g.drawLine(double x0, double y0, double x1, double y1)* `draw(Graphics g)`
`Polygon`	*// Create a polygon that consists of all the consecutive midpoints of the calling polygon* `midPoints()`
`double`	*// Return the perimeter of the polygon that is equal to the sum of the consecutive lines' distance.* *// i.e., distance between p1 and p2 + distance between p2 and p3 + … + distance between last point and p1* `perimeter()`
`String`	`toString()` *// string representation such as (x1, y1)-(x2,y2)-(x3,y3)*

In the `Polygon` class write the following main method that tests your implementation:

```
public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(100, 100);
        Polygon polygon = new Polygon();
        Point p1 = new Point(10, 10);
        Point p2 = new Point(60, 10);
        Point p3 = new Point(60, 60);
        Point p4 = new Point(10, 60);
        polygon.addPoint(p1);
        polygon.addPoint(p2);
        polygon.addPoint(p3);
        polygon.addPoint(p4);
        System.out.println(polygon); // prints (10,10)-(60,10)-(60,60)-(10,60)
        System.out.println(polygon.perimeter()); // prints 200.0
        Polygon midPointsPolygon = polygon.midPoints();
        polygon.draw(panel.getGraphics());
        midPointsPolygon.draw(panel.getGraphics());
}
```



Problem 2 Bank

2.a. Write a java class `Account` that contains a definition for a simple bank account. Your class should implement the API below (*note:* you may add any additional method that you see fit):

Class: Account
- id: int // positive integer between 0 and 10000 - name: String - balance: double

Account(id, name, initialBalance); // constructor
Account(name); // constructor, where id is set randomly, balance is set to 0.0
withdraw(amount); // withdraws a specified amount from the account
deposit(amount); // deposits the specified amount to the account.
toString(); // string representation of the account

2.b. Write a java class `Bank.java` that simulates a set of bank accounts. Your class should contain an array of accounts. ***This array is expanded as needed*** (in case of opening and closing an account). Your class should implement the API below (add any class method that you see fit):

```
// constructor, from a file. Each line of the file contains account id, name, and
// initial balance, e.g.:
// 123 Shadi 1000
// 222 Samir 2000
public Bank(String fileName)

// Remove the account from the array given its id.
// The array must be resized. Returns false if the account id does not exist and true
// otherwise
public boolean close(int id)

//create a new account. The array must be resized. Returns false if the id already
// exists, true otherwise.
public boolean open(int id, String name, double balance)

// Transfer funds from one account to another given their ids. Returns true if the
// transfer is done successfully.
public boolean transfer(int idFrom, int idTo, double balance)

// returns a String representing the accounts
public String toString()
```

2.c. Write a program `TestBank.java` that prompts the user to enter a choice as shown in the sample output below. You can enter: 1 for opening a new account; 2 for closing/removing an account; 3 for checking the balance of a given account; 4 for withdrawing money from an account; 5 for deposing money to an account; 6 for transferring money to another account; 7 for loading the accounts from a file (you also need to input the file name); 8 for exit. ***Once exit, your program must display the information of all the accounts.***

```
Main Menu (1: Open - 2: Close - 3: Balance - 4: Withdraw - 5: Deposit -
6: Transfer  - 7: Load - 8: Exit)
Enter a choice: 1
Enter the account info (id, name, initial balance): 10 Chadi 100
The account has been created successfully.
```

```
Main Menu (1: Open - 2: Close - 3: Balance - 4: Withdraw – 5: Deposit -
6: Transfer  - 7: Load - 8: Exit)
Enter a choice: 3
Enter the account id: 7
The balance is 100.0

Main Menu (1: Open - 2: Close - 3: Balance - 4: Withdraw – 5: Deposit -
6: Transfer  - 7: Load - 8: Exit)
Enter a choice: 6
Enter idFrom, idTo, Balance: 7 10 100
The transfer has been done successfully.
...
```

Problem 1 `Longest Sorted Sequence`

Given a list of numbers stored in an **input file**, you have to output the longest sorted (i.e., increasing order) sequence of integers in the list. For example, in the list below:

`3, 8, 10, 1, 9, 14, -3, 0, 14, 207, 56, 98, 12`

the longest sorted sequence should be: `-3, 0, 14, 207`

Notice that the sequence may contain duplicates. For example, if the list of numbers is:

`17, 42, 3, 5, 5, 5, 8, 2, 4, 6, 1, 19`

the longest sorted sequence becomes: `3, 5, 5, 5, 8`

Input Format
The input file starts with a number *N* indicating the number of integers in the list. The next line contains the *N* numbers.

Output Format
Display the longest non-decreasing sorted sequence on the screen.

Sample Input file 1: lss-1.in
```
13
3 8 10 1 9 14 -3 0 14 207 56 98 12
```

Sample Output 1
```
LSS = -3, 0, 14, 207
```

Sample Input file 2: lss-2.in
```
12
17 42 3 5 5 5 8 2 4 6 1 19
```

Sample Output 2
```
LSS = 3, 5, 5, 5, 8
```

Submission Instructions

- As usual, submit your commented source code and sample runs in a zip file named s*#*_asst12_*netid*, where *#* is your section number and *netid* stands for your AUBnet user name.