Exercise One: Classes and Objects

1. (Easy String Compression)(30 Points)

Given an array of characters, compress it following the rules shown in the following examples. After compression, print the new length of compressed string and the string itself to console.

Example 1:

Input:

["a","a","b","b","c","c","c"]

Output:

6, "a2b2c3"

Explanation:

"aa" is replaced by "a2". "bb" is replaced by "b2". "ccc" is replaced by "c3".

Example 2:

Input:

["a"]

Output:

1, "a"

Explanation:

Nothing is replaced.

Example 3:

Input:

["a","b","b","b","b","b","b","b","b","b","b","b","b"]

Output:

4, "ab12"

Explanation:

Since the character "a" does not repeat, it is not compressed. "bbbbbbbbbbbb" is replaced by "b12".

Notice each digit has its own entry in the array.

Requirements:

- You should implement a class named **StringCompression** and write a test program that tests the examples as shown above.

2. (The Polynomial Class)(30 Points)

Design a class named **Polynomial**, which represents a quadratic polynomial with one variable, i.e., $F(x)=ax^2+bx+c$. It can be used to calculate F(x) for the given x, or find the solution x satisfying F(x)=y where y is known.

It contains:

- Fields :
  - *a* : **double**, The quadratic coefficient
  - *b* : **double**, The monomial coefficient
  - *c* : **double**, The constant
- Methods :
  - Polynomial(double a,double b,double c) : the constructor which will initialize all the variables
  - setA(double a), setB(double b), setC(double c) : several **set methods** to set the variables
  - showPolynomial() : output the polynomial to the console in the form of "$F(x)=ax^2+bx+c$"
  - getY(double x) : calculate and return F(x)
  - hasSolution(double y) : return a **boolean** value *true* if F(x)=y has a solution in real number system, and *false* otherwise
  - showSolution(double y) : calculate the solution of F(x)=y, and output "solution : " follow by the x to console if any solution exists, output "no solution" to console otherwise.

Requirements:

- You should implement the class and pass the test program as shown below.
- The Polynomial Class must contain methods and fields mentioned above, and it also works if your implement has more methods and fields than the requirement.

Test program:

- test code :

```java
/**
 *  @File:    PolynomialDemo.java
 *  @Brief:   A demo to test the Polynomial Class
 *  @Author:  KaiMing Zhu
 *  @Date:    2020/09/12
 *  @Course:  Java2020
 **/


public class PolynomialDemo {
    public static void main(String[] args)
    {
        Polynomial f = new Polynomial(10,9,8);
        Polynomial f1 = new Polynomial(1, 0, 0);
        Polynomial f2 = new Polynomial(1, 0, 2);
        int[] tempArray = {0, 1, 3};

        // Set Method Test
        System.out.println("Set Method Test");
        f.showPolynomial();
        f.setA(-10);
        f.showPolynomial();
        f.setB(-1908);
        f.showPolynomial();
        f.setC(190908.87);
        f.showPolynomial();


        // F(x) Calculation Test
        System.out.println("\nCalculation Test");
        f1.showPolynomial();
        f2.showPolynomial();
        System.out.println("");
        for(int i = 0; i < tempArray.length;i++) {
            System.out.println("f1(" + String.valueOf(tempArray[i]) +
")=" + String.valueOf(f1.getY(tempArray[i])));
            System.out.println("f2(" + String.valueOf(tempArray[i]) +
")=" + String.valueOf(f2.getY(tempArray[i])));
        }

        // F(x)=0 Solution Test
        System.out.println("\nSolution Test");
        for(int i = 0; i < tempArray.length;i++){
            System.out.println("y = " + String.valueOf(tempArray[i]));
            boolean result1 = f1.hasSolution(tempArray[i]);
            boolean result2 = f2.hasSolution(tempArray[i]);
            System.out.print("f1:");
            System.out.println(result1);
            if(result1){
                f1.showSolution(tempArray[i]);
            }
            System.out.print("f2:");
            System.out.println(result2);
            if(result2){
                f2.showSolution(tempArray[i]);
            }
            System.out.println("");
            }

      }
}
```

- input : None
- output :

```
Set Method Test
F(x)=10.0x² +9.0x+8.0
F(x)=-10.0x² +9.0x+8.0
F(x)=-10.0x² +-1908.0x+8.0
F(x)=-10.0x² +-1908.0x+190908.87

Calculation Test
F(x)=1.0x² +0.0x+0.0
F(x)=1.0x² +0.0x+2.0

f1(0)=0.0
f2(0)=2.0
f1(1)=1.0
f2(1)=3.0
f1(3)=9.0
f2(3)=11.0

Solution Test
y = 0
f1:true
solution: -0.0
f2:false

y = 1
f1:true
solution: 1.0, -1.0
f2:false

y = 3
f1:true
solution: 1.7320508075688772, -1.7320508075688772
f2:true
solution: 1.0, -1.0
```

3. (The Game Class and the Player Class)(40 Points)

Simulate the game **rock-paper-scissors**, and output its process. This practice combines with two classes, the **Game** class and the **Player** class.

The Player class contains:

- Fields :
  - *name* : **String**, the name of the player
  - *victoryTimes* : **int,** the cumulative times that the player has won the game

- *gameTimes* : **int,** the cumulative times that the player has participated the game

- Methods :
  - player(String name) : the constructor which will initialize the variables
  - getName(), getvictoryTimes(), getGameTimes() : several **get method** to get the variables
  - reset(): reset victoryTimes and gameTimes to 0
  - recordGame(boolean isVictory): add 1 to gameTimes, and add 1 to victoryTimes if isVictory is true
  - chooseShape() : return a **random int value** in range $[0,2]$, representing rock, paper and scissors the player chooses respectively
  - showMetrics() : output the name follow by a ":" and the rate of winning to console, e.g. "ZhangSan : 0.495"

The Game class contains:

- Variables :
  - *firstPlayer* : **Player,** the first Player in this game
  - *secondPlayer* : **Player,** the second Player in this game
- Methods :
  - setFirstPlayer(), setSecondPlayer() : several **set method** to set the variables
  - getFirstPlayer(), getSecondPlayer() : several **get method** to get the variables
  - run(int n): simulate the **rock-paper-scissors** game. The players will play exact **n** times. You should use **chooseShape** to get shapes the players choose and then judge the victory. Use **recordGame** to update the member variables once a player wins. Output every game's result in the form of "(firstPlayer's name) : (firstPlayer's shape)   (secondPlayer's name) : (secondPlayer's shape)   result : (winner's name) wins". Finally show each player's metrics. Check the following example for more details.

Example:

```
Input : 3


Output :

ZhangSan : rock    LiSi : paper    result : LiSi wins

ZhangSan : scissors    LiSi : paper    result : ZhangSan wins

ZhangSan : scissors    LiSi : scissors    result : Draw

ZhangSan : 0.333

LiSi : 0.333
```

Requirements:

- You should implement the class and pass the test program as shown below.
- The Game Class and the Player Class must contain methods and fields mentioned above, and it also works if your implement has more methods and fields than the requirements.

Test program:

- test code :

```java
public class RockPaperScissors {
    public static void main(String[] args) {
        Player zhangSan = new Player("ZhangSan");
        Player liSi = new Player("LiSi");
        Player wangWu = new Player("WangWu");
        Game game = new Game();

        System.out.println("Game.run Test");
        game.setFirstPlayer(zhangSan);
        game.setSecondPlayer(liSi);
        game.run(5);

        System.out.println("\nPlayer.reset test");
        System.out.println("before reset:");
        System.out.printf("victoryTimes : %d\n",
liSi.getVictoryTimes());
        System.out.printf("gameTimes : %d\n", liSi.getGameTimes());
        liSi.reset();
        System.out.println("after reset:");
        System.out.printf("victoryTimes : %d\n",
liSi.getVictoryTimes());
        System.out.printf("gameTimes : %d\n", liSi.getGameTimes());

        System.out.println("\nGame.setFirstPlayer test");
        game.setFirstPlayer(wangWu);
        game.run(5);
    }
}
```

- input : None
- output :

```
Game.run Test
ZhangSan : rock   LiSi : paper   result : LiSi
ZhangSan : scissors   LiSi : paper   result : ZhangSan
ZhangSan : paper   LiSi : paper   result : Draw
ZhangSan : scissors   LiSi : paper   result : ZhangSan
ZhangSan : scissors   LiSi : scissors   result : Draw
ZhangSan : 0.400000
LiSi : 0.200000

Player.reset test
before reset:
victoryTimes : 1
gameTimes : 5
after reset:
victoryTimes : 0
gameTimes : 0

Game.setFirstPlayer test
WangWu : rock   LiSi : scissors   result : WangWu
WangWu : scissors   LiSi : scissors   result : Draw
WangWu : rock   LiSi : paper   result : LiSi
WangWu : paper   LiSi : scissors   result : LiSi
WangWu : rock   LiSi : rock   result : Draw
WangWu : 0.200000
LiSi : 0.400000
```

Note that because of the randomness, the shapes that players choose maybe different from the example above.

Hints: You can use the library *java.util.Random* to create several random int