

Assignment 1: A Rock-Paper-Scissors Game

Deadline: 23:59pm, April 23, 2023

In this assignment, you are required to write a simple Rock-Paper-Scissors game in Java. This involves classes/objects, strings, functions, array of objects, and flow control. Your tasks are divided into two parts, and you are required to complete both.

1 Basic Level: Initializing a Random Sequence of Shapes (50%)

Your first task is to write a program that can initialize a sequence of shapes, which is used for playing the Rock-Paper-Scissors game, as introduced in the advanced level task. The program takes two integers from the user, where the first one is the total number of rounds, and the second one is the seed for random number generation. Finally, it prints the entire sequence of shapes. We provide a skeleton of code in the appendix, i.e., `Shape_sample.java`, and you need to complete the program.

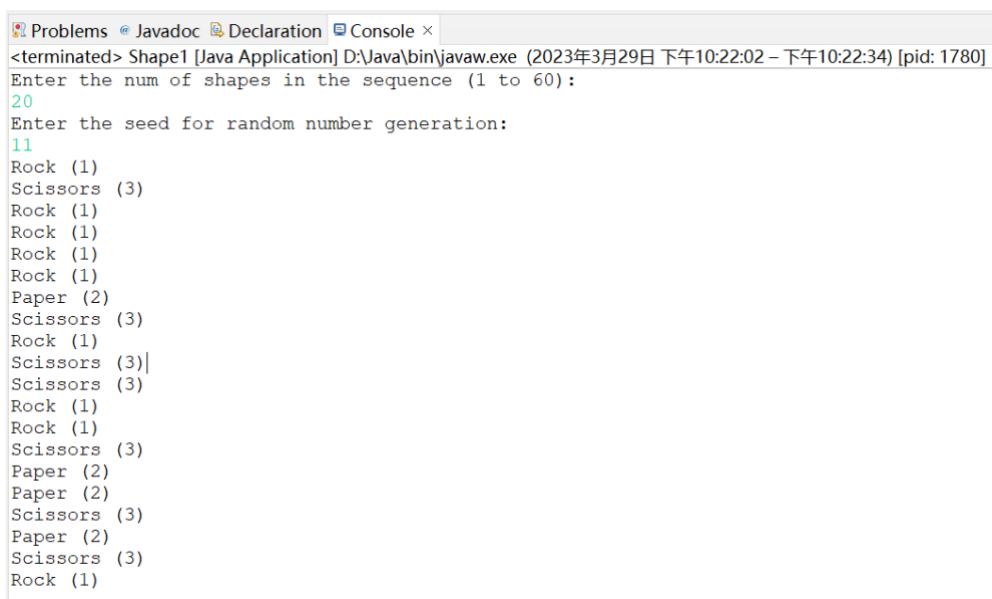
Now to begin, a shape has two attributes: *name*, and *value*. There are 3 shapes, and their names are: “Rock”, “Paper”, and “Scissors”. The value of a shape simply depends on its name: a “Rock” has a value of 1, “Paper” 2, and “Scissors” 3. All the shape names are already defined in the `main()` function as an array of `String`.

A class called `Shape` is provided in the skeleton code. You should add the above attributes defined as *private* members. *name* is a string type and *private* members. *value* is an integer type. You also need to implement these access functions and the default constructor in order to complete the class definition of `Shape`.

To initialize a random sequence of shapes, you need to implement one function. A sequence is simply an array of `Shape` objects as you can see in `public static void main(String[] args).initSequence(shapeSeq, shapeName, num, seed)` is the function to initialize the sequence of shapes, where the integer value `num` indicates the number of shapes in this sequence, and the integer value `seed` is used for random number generation. When you create each shape, you should randomly select its value from 1 to 3, and set its name according to the value as mentioned before.

You also need to complete a help function called `printSequence` which simply takes the sequence and the current number of shapes, and prints each `Shape` object sequentially. This is called in the `main()` function to check the output of your code.

A sample run of the program with 20 as the number of shapes and 11 as the seed is shown below in Figure 1.



```
Problems  Javadoc  Declaration  Console  x
<terminated> Shape1 [Java Application] D:\Java\bin\javaw.exe (2023年3月29日 下午10:22:02 - 下午10:22:34) [pid: 1780]
Enter the num of shapes in the sequence (1 to 60):
20
Enter the seed for random number generation:
11
Rock (1)
Scissors (3)
Rock (1)
Rock (1)
Rock (1)
Rock (1)
Paper (2)
Scissors (3)
Rock (1)
Scissors (3)|
Scissors (3)
Rock (1)
Rock (1)
Scissors (3)
Paper (2)
Paper (2)
Scissors (3)
Paper (2)
Scissors (3)
Rock (1)
```

Figure 1: A run of the basic level program with 20 as the num and 11 as the seed.

What you need to do: Based on `Shape_sample.java`, complete the class definition of `Shape`. Do not add/remove attributes or member functions. Finish the implementation of the default constructor and access functions. Finish the implementation of the `initSequence` function and the `printSequence` function. Complete the `main()` function based on the sample output shown in Figure 1.

What you need to submit: Submit the completed source file.

2 Advanced Level: Playing the Rock-Paper-Scissors Game (50%)

Your next task is to extend your program in the basic level, so it can play the Rock-Paper-Scissors game with you as the only player.

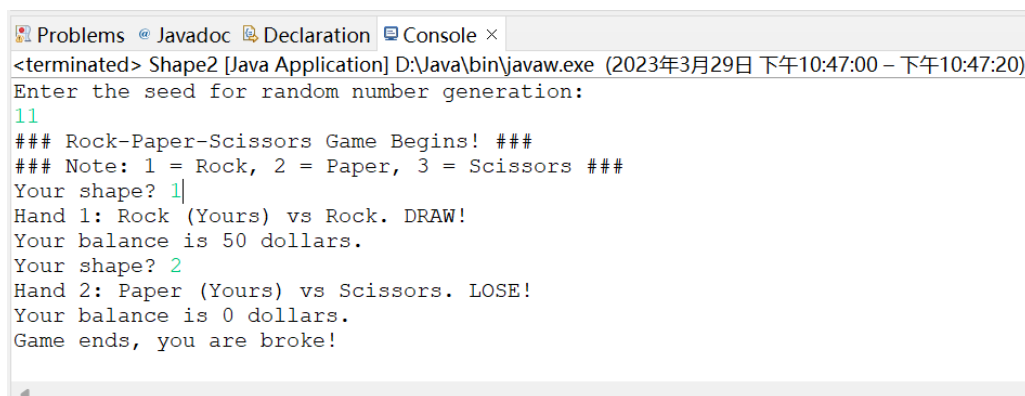
The game is as follows. At first, the program asks the user the seed for random number generation. It initializes a random sequence of shapes, just like the basic level. Note that you are not required to input the number of shapes, and the number is fixed at 20 now. Then the game starts. Initially, you have 50 dollars. In each round, the program asks you to enter an integer that indicates a shape you will choose, where 1 stands for “Rock”, 2 for “Paper”, and 3 for “Scissors”. Here we would just assume that the input integer is 1 or 2 or 3, for simplicity. Then the program will deal one hand to you by drawing a shape in order from the sequence of shapes initialized by the seed, and showing it to you. The game continues until one of these conditions is met: (1) Your balance becomes 0. Then the game ends automatically since you are

broke. (2) You choose to leave the game with a positive balance. You can do this by entering “-1” as your input, and the game will end, and your balance will be shown. Note that you must play at least three hands before you can leave the game. (3) The entire sequence of 20 shapes is drawn. The game automatically ends, and your balance is shown as well.

The rules of comparison between shapes are simple: rock beats scissors, paper beats rock, and scissors beats paper. If you win, your balance will increase by 50 dollars. If you lose, it will decrease by 50 dollars. If the program and you perform the same shape, your balance will not be changed.

Some sample runs of the game are shown below. **Note your program output should be EXACTLY the same as the sample output below, using the same seed, if you are using jdk.17.0.6.**

Run 1: Figure 2 shows a run with 11 as the seed. We lose the game after 2 hand. The result of your game is shown right after a hand is dealt to you.



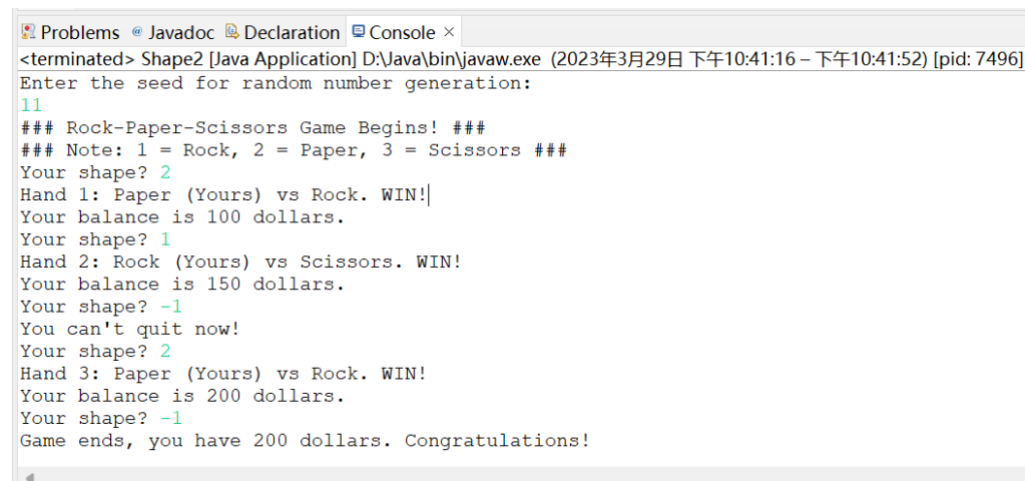
```

Problems @ Javadoc Declaration Console x
<terminated> Shape2 [Java Application] D:\Java\bin\javaw.exe (2023年3月29日 下午10:47:00 – 下午10:47:20)
Enter the seed for random number generation:
11
### Rock-Paper-Scissors Game Begins! ###
### Note: 1 = Rock, 2 = Paper, 3 = Scissors ###
Your shape? 1
Hand 1: Rock (Yours) vs Rock. DRAW!
Your balance is 50 dollars.
Your shape? 2
Hand 2: Paper (Yours) vs Scissors. LOSE!
Your balance is 0 dollars.
Game ends, you are broke!

```

Figure 2: A run of the advanced level Rock-Paper-Scissors program with 11 as the seed.

Run 2: Figure 3 shows a run with 11 as the seed. We wanted to quit at the third hand, but the system does not allow that. We can quit after playing at least 3 hands.



```

Problems @ Javadoc Declaration Console x
<terminated> Shape2 [Java Application] D:\Java\bin\javaw.exe (2023年3月29日 下午10:41:16 – 下午10:41:52) [pid: 7496]
Enter the seed for random number generation:
11
### Rock-Paper-Scissors Game Begins! ###
### Note: 1 = Rock, 2 = Paper, 3 = Scissors ###
Your shape? 2
Hand 1: Paper (Yours) vs Rock. WIN!
Your balance is 100 dollars.
Your shape? 1
Hand 2: Rock (Yours) vs Scissors. WIN!
Your balance is 150 dollars.
Your shape? -1
You can't quit now!
Your shape? 2
Hand 3: Paper (Yours) vs Rock. WIN!
Your balance is 200 dollars.
Your shape? -1
Game ends, you have 200 dollars. Congratulations!

```

Figure 3: A run of the advanced level Rock-Paper-Scissors program with 11 as the seed.

Run 3: Figure 4 shows a run with 11 as the seed. We played five hands and chose to quit with 150 dollars.



```
Problems @ Javadoc Declaration Console x
<terminated> Shape2 [Java Application] D:\Java\bin\javaw.exe (2023年3月29日 下午10:49:05 - 下午10:49:41) [pid: 9636]
Enter the seed for random number generation:
11
### Rock-Paper-Scissors Game Begins! ###
### Note: 1 = Rock, 2 = Paper, 3 = Scissors ###
Your shape? 2
Hand 1: Paper (Yours) vs Rock. WIN!
Your balance is 100 dollars.
Your shape? 1
Hand 2: Rock (Yours) vs Scissors. WIN!
Your balance is 150 dollars.
Your shape? 2
Hand 3: Paper (Yours) vs Rock. WIN!
Your balance is 200 dollars.
Your shape? 3
Hand 4: Scissors (Yours) vs Rock. LOSE!
Your balance is 150 dollars.
Your shape? 1
Hand 5: Rock (Yours) vs Rock. DRAW!
Your balance is 150 dollars.
Your shape? -1
Game ends, you have 150 dollars. Congratulations!
```

Figure 4: A run of the advanced level Rock-Paper-Scissors program with 11 as the seed.

What you need to do: Based on your solution of the basic level task, implement the Rock-Paper-Scissors game as specified above. Do not remove attributes or member functions of Shape, or `initSequence` or other access functions. You may add new member functions to Shape and new non-member functions to the program. **Again, your output must be EXACTLY the same as the above figures if you are using jdk17.0.6.** You are encouraged to try different seeds and play different bets to test your own solution (it's a game anyway :)).

What you need to submit: Submit the completed source file.

Submission guideline: You should submit two java files (or a ZIP file if your email server does not support attaching code files) via email to **Miss Man Li** (liman@mail.nwpu.edu.cn) before the deadline. Make sure that your code is runnable with Java 17.

Appendix

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

public class Shape {
    public static final int MAX_NUM_OF_SHAPES = 60;

    // TODO: Complete the class definition of Shape

    // TODO: Complete the code that initializes a random sequence of shapes
    // You need to set its name and value for each shape:
    // Rock = 1, Paper = 2, Scissors = 3
    public static void initSequence(Shape[] shapeSeq, String[] shapeName, int
num, int seed) {
        Random random = new Random(seed);
    }

    // TODO: Complete the code that prints the sequence of shapes sequentially
    public static void printSequence(Shape[] shapeSeq, int num) {
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Shape[] shapeSeq = new Shape[MAX_NUM_OF_SHAPES];
        String[] shapeName = { "Rock", "Paper", "Scissors" };

        int num = 0;
        // read the num from the console
        System.out.println("Enter the num of shapes in the sequence (1 to 60):
");
        num = sc.nextInt();

        // TODO: Check the range of num

        int seed = 0;
        System.out.println("Enter the seed for random number generation: ");
        seed = sc.nextInt();

        initSequence(shapeSeq, shapeName, num, seed);
        printSequence(shapeSeq, num);
    }
}
```