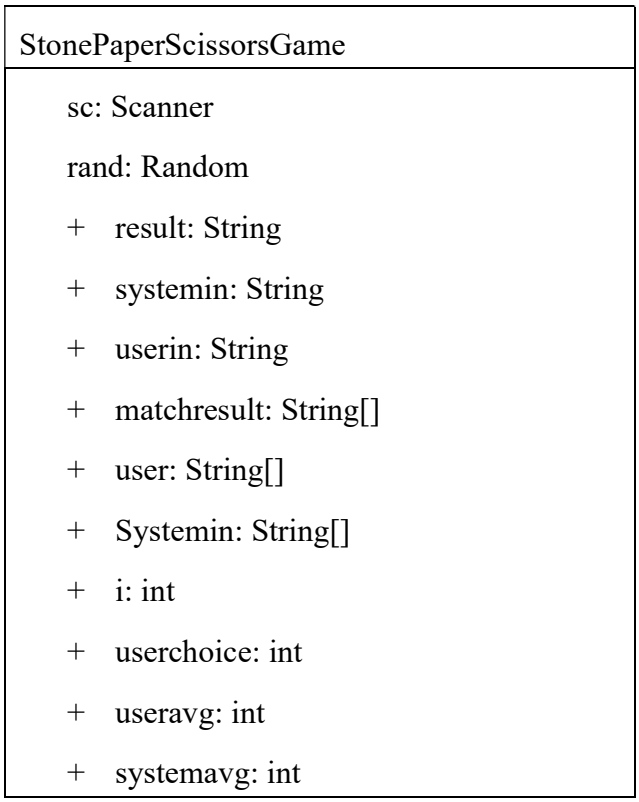
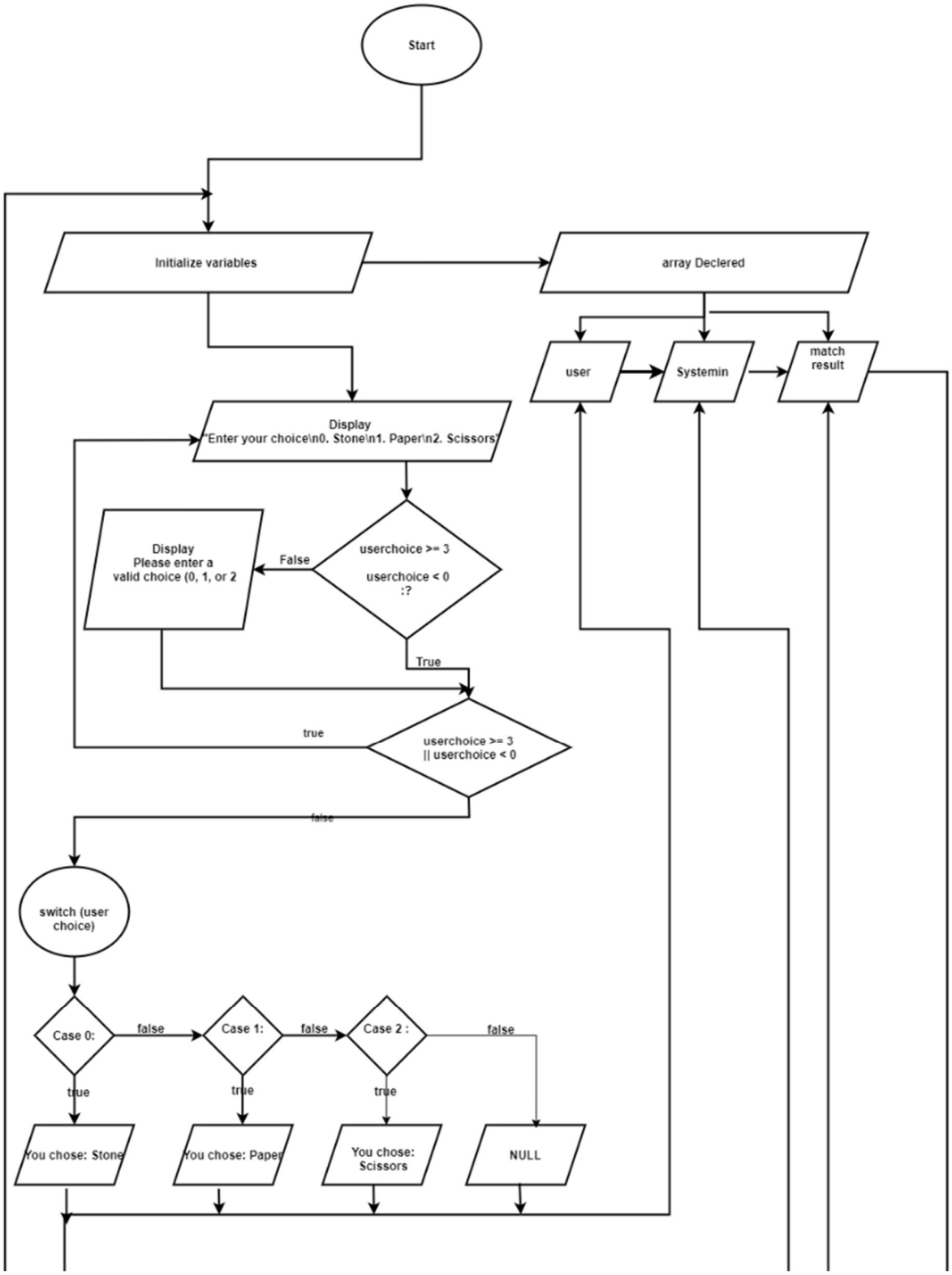
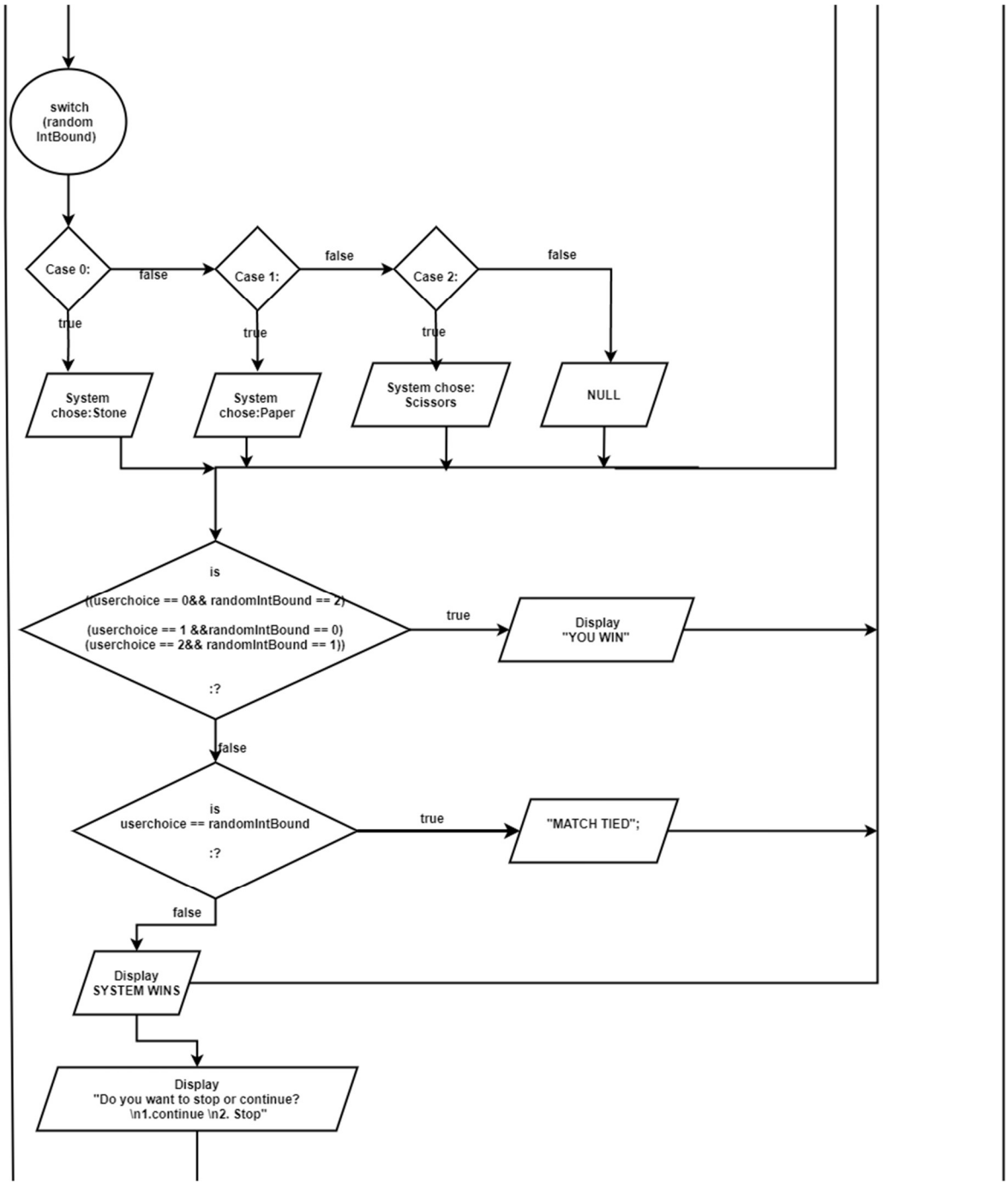
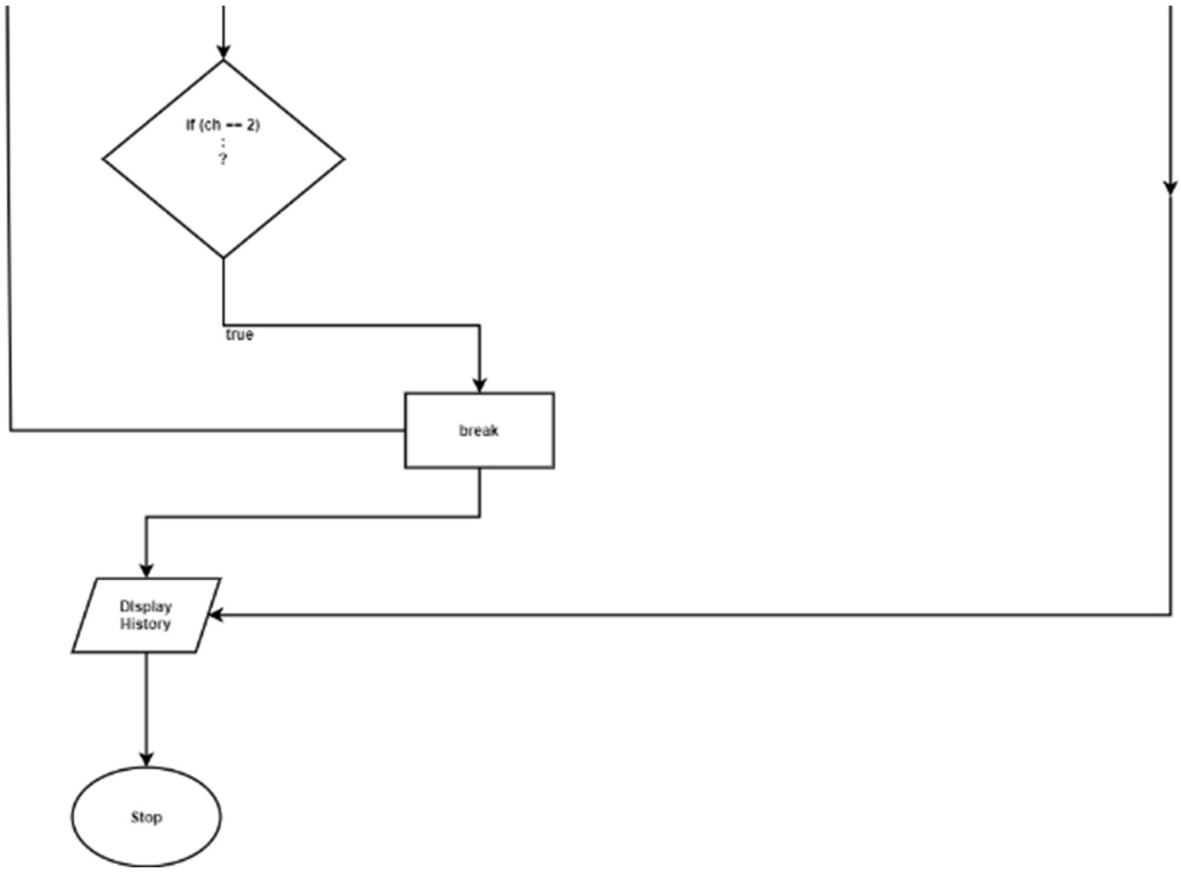
1.UML Diagram



2.Flowchart


Start
Initialize variables
array Declered
user
Systemin
match result
Display "Enter your choice\n0. Stone\n1. Paper\n2. Scissors"
Display Please enter a valid choice (0, 1, or 2
False
userchoice >= 3 userchoice < 0 :?
True
true
userchoice >= 3 || userchoice < 0
false
switch (user choice)
Case 0:
false
Case 1:
false
Case 2 :
false
true
true
true
You chose: Stone
You chose: Paper
You chose: Scissors
NULL


switch
(random
IntBound)
Case 0:
false
Case 1:
false
Case 2:
false
true
true
true
System
chose:Stone
System
chose:Paper
System chose:
Scissors
NULL
is
((userchoice == 0&& randomIntBound == 2)
(userchoice == 1 &&randomIntBound == 0)
(userchoice == 2&& randomIntBound == 1))
:?
true
Display
"YOU WIN"
false
is
userchoice == randomIntBound
:?
true
"MATCH TIED";
false
Display
SYSTEM WINS
Display
"Do you want to stop or continue?
\n1.continue \n2. Stop"


If (ch == 2)
;
?
true
break
Display
History
Stop

3.Code of StonePaperScissorsGame

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

public class StonePaperScissorsGame{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      Random rand = new Random();

      // Variables to store user/system input and results
      String result, systemin, userin;
      String[] matchresult = new String[100];
      String[] user = new String[100];
      String[] Systemin = new String[100];

      int i = 0;  // Index for storing game outcomes
      int userchoice,useravg=0,systemavg=0;

      // Main loop for multiple rounds
      while (true) {
         // Input validation for user choice
         do {
            System.out.println("Enter your choice\n0. Stone\n1. Paper\n2. Scissors : ");
            userchoice = sc.nextInt();
            if (userchoice >= 3 || userchoice < 0) {
               System.out.println("Please enter a valid choice (0, 1, or 2): ");
            }
         } while (userchoice >= 3 || userchoice < 0);

         // Randomly generate system's choice (0 for Stone, 1 for Paper, 2 for Scissors)
```

```
int randomIntBound = rand.nextInt(3);

// Store user's choice
switch (userchoice) {
   case 0:
      userin = "Stone";
      System.out.println("You chose: Stone");
      break;
   case 1:
      userin = "Paper";
      System.out.println("You chose: Paper");
      break;
   case 2:
      userin = "Scissors";
      System.out.println("You chose: Scissors");
      break;
   default:
      userin = "";
}

// Store system's choice
switch (randomIntBound) {
   case 0:
      systemin = "Stone  ";
      System.out.println("System chose: Stone");
      break;
   case 1:
      systemin = "Paper";
      System.out.println("System chose: Paper");
      break;
   case 2:
```

```
            systemin = "Scissors";
            System.out.println("System chose: Scissors");
            break;
        default:
            systemin = "";
    }

    // Determine the result and store it
    if ((userchoice == 0 && randomIntBound == 2) ||
        (userchoice == 1 && randomIntBound == 0) ||
        (userchoice == 2 && randomIntBound == 1)) {
        result = "YOU WIN";
        useravg++;
    } else if (userchoice == randomIntBound) {
        result = "MATCH TIED";
    } else {
        result = "SYSTEM WINS";
        systemavg++;
    }

    // Display the result of the current round
    System.out.println(result);

    // Store the choices and result in arrays
    user[i] = userin;
    Systemin[i] = systemin;
    matchresult[i] = result;
    i++;  // Increment the index for next round

    // Ask if user wants to continue or stop
    System.out.println("Do you want to stop or continue? \n1.continue \n2. Stop");
```

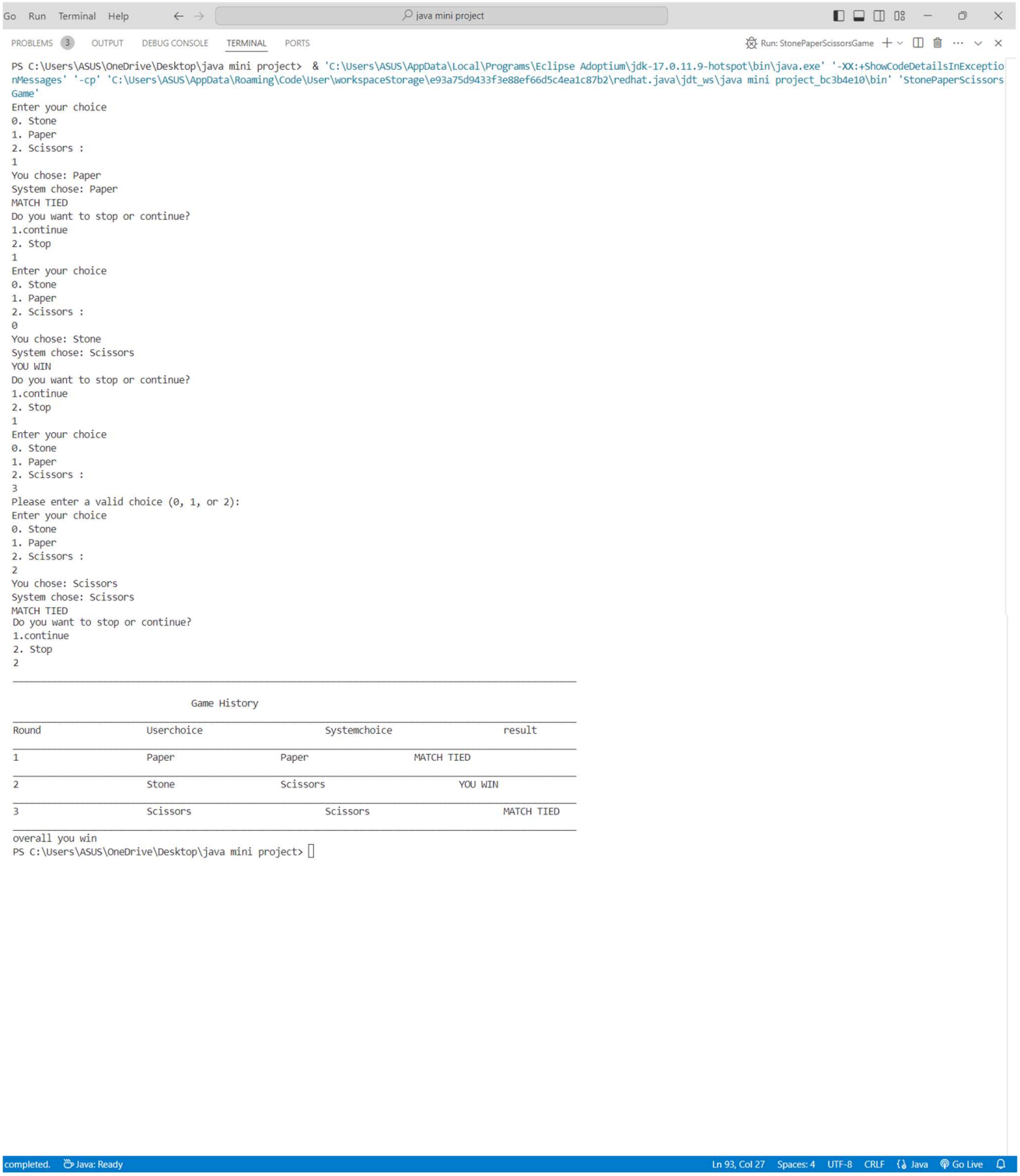
```
            int ch = sc.nextInt();
            if (ch == 2) {
                break;
            }
        }

        // Display the history of the game
         System.out.println("_____
_____");
        System.out.println("\n\t\t\t\tGame History\t\t\t\t\t\t\t\t");
         System.out.println("_____
_____");
        System.out.println("Round\t\t\tUserchoice\t\t\tSystemchoice\t\t\tresult ");
         System.out.println("_____
_____");
        for (int j = 0; j < i; j++) {

           System.out.println((j+1) + "\t\t\t" + user[j] + "\t\t\t" + Systemin[j] + "\t\t\t" + matchresult[j]);
             System.out.println("_____
_____");
        }
        if(useravg>systemavg){
            System.out.println("overall you win ");
        }else if(useravg<systemavg){
            System.out.println("overall System win ");
        }else{
            System.out.println("overall match tied ");
        }

    }
}
```

4.OUTPUT


Go Run Terminal Help
java mini project
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run: StonePaperScissorsGame
PS C:\Users\ASUS\OneDrive\Desktop\java mini project> & 'C:\Users\ASUS\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.11.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ASUS\AppData\Roaming\Code\User\workspaceStorage\e93a75d9433f3e88ef66d5c4ea1c87b2\redhat.java\jdt_ws\java mini project_bc3b4e10\bin' 'StonePaperScissorsGame'
Enter your choice
0. Stone
1. Paper
2. Scissors :
1
You chose: Paper
System chose: Paper
MATCH TIED
Do you want to stop or continue?
1.continue
2. Stop
1
Enter your choice
0. Stone
1. Paper
2. Scissors :
0
You chose: Stone
System chose: Scissors
YOU WIN
Do you want to stop or continue?
1.continue
2. Stop
1
Enter your choice
0. Stone
1. Paper
2. Scissors :
3
Please enter a valid choice (0, 1, or 2):
Enter your choice
0. Stone
1. Paper
2. Scissors :
2
You chose: Scissors
System chose: Scissors
MATCH TIED
Do you want to stop or continue?
1.continue
2. Stop
2
Game History
Round Userchoice Systemchoice result
1 Paper Paper MATCH TIED
2 Stone Scissors YOU WIN
3 Scissors Scissors MATCH TIED
overall you win
PS C:\Users\ASUS\OneDrive\Desktop\java mini project>
completed. Java: Ready
Ln 93, Col 27 Spaces: 4 UTF-8 CRLF Java Go Live

5.Explanation of code

1. Importing Libraries

import java.util.Random;

import java.util.Scanner;

- Random: This library is used to generate random numbers, which are needed for the computer’s (system's) choice in the game.
- Scanner: This is used to take input from the user.

2. Main Class Declaration

public class StonePaperScissorsGame {

public static void main(String[] args) {

- The class `StonePaperScissorsGame` contains the entire program logic.
- The `main` method is the entry point of the program, where execution begins.

3. Initializing Variables

Scanner sc = new Scanner(System.in);

Random rand = new Random();

- Scanner sc: This initializes the `Scanner` object to read user input from the console.
- Random rand: This creates a `Random` object to generate a random choice for the system.

String result, systemin, userin;

String[] matchresult = new String[100];

String[] user = new String[100];

String[] Systemin = new String[100];

- `result`, `systemin`, `userin`: These are used to store the result of each round, the system's choice, and the user's choice, respectively.
- Arrays `matchresult`, `user`, `Systemin`: These store the results, the user's choices, and the system's choices for up to 100 rounds.

int i = 0;

int userchoice, useravg = 0, systemavg = 0;

- i: Index variable to track the number of rounds.
- userchoice: Stores the user's choice as an integer (0 for Stone, 1 for Paper, 2 for Scissors).

- useravg, systemavg: These track the number of rounds the user and system win, respectively.

4. Game Loop

```
while (true) {
```

- This while (true) loop allows the game to run continuously, prompting the user for input until they decide to stop.

5. User Input Validation

```
do {
   System.out.println("Enter your choice\n0. Stone\n1. Paper\n2. Scissors : ");
   userchoice = sc.nextInt();
   if (userchoice >= 3 || userchoice < 0) {
      System.out.println("Please enter a valid choice (0, 1, or 2): ");
   }
} while (userchoice >= 3 || userchoice < 0);
```

- The program asks the user to input their choice (0 for Stone, 1 for Paper, 2 for Scissors).
- It uses a `do-while` loop to ensure the input is valid (between 0 and 2). If the input is invalid, the user is prompted again.

6. System's Random Choice

```
int randomIntBound = rand.nextInt(3);
```

- The system's choice is generated randomly using `rand.nextInt(3)` which produces 0, 1, or 2 (for Stone, Paper, or Scissors).

7. Mapping Choices

```
switch (userchoice) {
   case 0: userin = "Stone"; System.out.println("You chose: Stone"); break;
   case 1: userin = "Paper"; System.out.println("You chose: Paper"); break;
   case 2: userin = "Scissors"; System.out.println("You chose: Scissors"); break;
}
```

- The switch statement converts the user's numeric input (0, 1, or 2) into the corresponding string ("Stone", "Paper", or "Scissors").
- The same process is repeated for the system's choice:

```
switch (randomIntBound) {
    case 0: systemin = "Stone"; System.out.println("System chose: Stone"); break;
    case 1: systemin = "Paper"; System.out.println("System chose: Paper"); break;
    case 2: systemin = "Scissors"; System.out.println("System chose: Scissors"); break;
}
```

8. Determining the Winner

```
if ((userchoice == 0 && randomIntBound == 2) ||
    (userchoice == 1 && randomIntBound == 0) ||
    (userchoice == 2 && randomIntBound == 1)) {
    result = "YOU WIN";
    useravg++;
} else if (userchoice == randomIntBound) {
    result = "MATCH TIED";
} else {
    result = "SYSTEM WINS";
    systemavg++;
}
```

- This block compares the user's choice and the system's choice to determine the result:
  - ❖ User wins:If the user's choice beats the system's choice (e.g., Stone vs Scissors).
  - ❖ Match tied: If both the user and the system choose the same option.
  - ❖ System wins:In all other cases.
- The respective win counters (useravg and systemavg) are incremented accordingly.

9. Storing and Displaying Results

```
user[i] = userin;
Systemin[i] = systemin;
matchresult[i] = result;
i++;
```

- After each round, the user's choice, system's choice, and result are stored in their respective arrays.

- i++: The round counter is incremented to track multiple rounds.

10. Asking to Continue or Stop

```
System.out.println("Do you want to stop or continue? \n1.continue \n2. Stop");
int ch = sc.nextInt();
if (ch == 2) {
   break;
}
```

- The user is asked whether they want to continue or stop playing:
- If the user enters `2`, the loop breaks, and the game ends.

11. Displaying Game History

```
System.out.println("_____
_____");
System.out.println("\n\t\t\t\tGame History\t\t\t\t\t\t\t\t");
System.out.println("_____
_____");
System.out.println("Round\t\t\tUserchoice\t\t\tSystemchoice\t\t\tresult ");
System.out.println("_____
_____");
for (int j = 0; j < i; j++) {
   System.out.println((j+1) + "\t\t\t" + user[j] + "\t\t\t" + Systemin[j] + "\t\t\t" + matchresult[j]);

System.out.println("_____
_____");
}
```

- The entire game history (user choice, system choice, and result for each round) is displayed in a tabular format.

12. Overall Result

```
if (useravg > systemavg) {
   System.out.println("overall you win ");
} else if (useravg < systemavg) {
   System.out.println("overall System win ");
} else {
   System.out.println("overall match tied ");
}
```

- Once the user chooses to stop, the overall winner is declared based on the `useravg` and `systemavg` counters.
  - ❖ If the user wins more rounds than the system, the user wins overall.
  - ❖ If the system wins more, the system is declared the overall winner.
  - ❖ If both win an equal number of rounds, the overall result is a tie.

13. Program End

- After the game results and history are displayed, the program terminates, as the user has chosen to stop.

CONCLUSIONS

The program implements a Stone-Paper-Scissors game in Java, allowing users to play multiple rounds against the computer until they decide to exit. Each round uses `Random` for generating the computer's choice and `Scanner` for user input. Arrays are employed to record each round's details, including the user's choice, the computer's choice, and the result of the game (win, lose, or draw). This array-based structure enables the program to store game history, potentially allowing future extensions to calculate player statistics or display detailed summaries of past rounds. Overall, the program showcases effective use of Java's arrays for data tracking, creating an engaging, interactive experience.