

```

1 import java.util.Random;
2 import java.util.Scanner;
3
4 /**
5  * Rock Paper Scissors Game - Blueprint Class
6  *
7  * @author Jasur Shukurov
8  * @version Version 3.1
9  * @version 12/12/2018
10 */
11
12
13 public class Player {
14
15     // Initializing variables
16     private static int wins = 0;
17     private static int losses = 0;
18     private static int ties = 0;
19
20     private static boolean stop = false;
21
22     private static Scanner scn = new Scanner(System.in);
23
24     /*
25     * determineWinner - method which finds who won the game
26     *
27     * @param p1 - player1 choice
28     * @param p2 - player2 choice
29     */
30
31     public static String determineWinner(String p1, String p2) {
32         if (p1.equalsIgnoreCase(p2)) {
33             ties();
34             return "Draw!";
35         } else if ((p1.equalsIgnoreCase("rock") && p2.equalsIgnoreCase("scissors"))
36             || (p1.equalsIgnoreCase("paper") && p2.equalsIgnoreCase("rock"))
37             || (p1.equalsIgnoreCase("scissors") && p2.equalsIgnoreCase("paper"))) {
38             wins();
39             return "Congratulation Player 1 won!";
40         } else {
41             losses();
42             return "Oh, sorry Player 2 won!";
43         }
44     }
45
46     /*
47     * ties - method which adds one to ties Statistics
48     */
49     private static void ties() {
50         ties++;
51     }
52
53     /*
54     * losses - method which adds one to losses Statistics
55     */
56     private static void losses() {
57         losses++;
58     }
59
60     /*
61     * wins - method which adds one to wins Statistics
62     */
63     private static void wins() {
64         wins++;
65     }
66
67     /*
68     * printStats() - this method generates game statistics
69     *
70     * @return String, returns game statistics
71     */
72     public static String printStats() {
73
74
75
76
77
78
79
80

```

```

81         return "\nYour Statistics:\nTies: " + ties + "\nWins: " + wins + "\nLosses: " + losses;
82     }
83 }
84
85 /*
86  * readUserChoice() - this method reads user input
87  *
88  * @return String, returns user input
89  *
90  */
91 public static String generateComputerChoice() {
92     Random rnd = new Random();
93     int player2 = rnd.nextInt(3) + 1;
94
95     if (player2 == 1) {
96         return "rock";
97     } else if (player2 == 2) {
98         return "paper";
99     } else {
100         return "scissors";
101     }
102 }
103
104 /*
105  * readUserChoice() - this method reads user input
106  *
107  * @return String, returns user input
108  *
109  */
110 private static String readUserChoice() {
111     System.out.println("Please type rock, paper or scissors: ");
112     String player1 = scn.nextLine();
113
114     while (!valid(player1)) {
115         System.out.println("Invalid input, please try again: ");
116         player1 = scn.nextLine();
117     }
118
119     return player1;
120 }
121
122 /*
123  * valid() - this method checks if user input is valid or not.
124  *
125  * @return boolean, if user input is valid it returns true, otherwise it
126  * returns false
127  */
128 private static boolean valid(String player1) {
129     switch (player1.toLowerCase()) {
130         case "rock":
131             return true;
132         case "scissors":
133             return true;
134         case "paper":
135             return true;
136         default:
137             return false;
138     }
139 }
140
141 /*
142  * playAgain() - method which asks from user, does s(he) want to play again
143  * ?
144  *
145  */
146 private static void playAgain() {
147     System.out.println("Do you want to play again? ");
148     String userInput = scn.nextLine();
149     if (userInput.equalsIgnoreCase("No")) {
150         stop = true;
151     }
152 }
153
154 /*
155  */
156
157
158
159
160

```

```

161  * choosingType() - method which asks user to choose what type of game (s)he
162  * wants to play. It can be Human against Human, Computer vs Human or
163  * Computer against Computer.
164  *
165  * @return int, it returns user choice
166  */
167  public static int choosingType() {
168
169      int type = scn.nextInt();
170      scn.nextLine();
171      while (type < 1 || type > 3) {
172          System.out.println("You typed invalid value, please try again!");
173          type = scn.nextInt();
174          scn.nextLine();
175      }
176
177      return type;
178  }
179
180
181  /*
182  * start() - is method which starts game
183  */
184  public String start() {
185
186      int type = choosingType();
187
188      if (type == 1)
189          return playerVsComputer();// player againsts computer
190      else if (type == 2)
191          return playerVsPlayer();// player againsts player
192      else
193          return computerVsComputer();// computer againsts computer
194  }
195
196
197  /*
198  * computerVsComputer() - computer plays against computer
199  */
200  private static String computerVsComputer() {
201
202      while (stop == false) {
203          playComputerAgainstComputer();
204          playAgain();
205      }
206
207      return printStats();
208  }
209
210  /*
211  * playComputerAgainstComputer()
212  *
213  * @return String
214  */
215  public static void playComputerAgainstComputer() {
216      String player1 = generateComputerChoice();
217      String player2 = generateComputerChoice();
218
219      System.out.println("Computer 1 chose - " + player1);
220      System.out.println("Computer 2 chose - " + player2);
221
222      String winner = determineWinner(player1, player2);
223
224      System.out.println(winner);
225  }
226
227  /*
228  * playerVsPlayer() - method to play against another player
229  */
230  private static String playerVsPlayer() {
231
232      while (stop == false) {
233          playAgainstPlayer();
234          playAgain();
235      }
236      return printStats();
237  }
238
239  /*
240  * playAgainstPlayer()

```

```

241  *
242  * @return String
243  *
244  */
245  public static void playAgainstPlayer() {
246      String player1 = readUserChoice();
247      String player2 = readUserChoice();
248      String winner = determineWinner(player1, player2);
249
250      System.out.println(winner);
251  }
252
253  /*
254   * playerVsComputer() - method to play against computer
255   *
256   */
257  private static String playerVsComputer() {
258      while (stop == false) {
259          playAgainstComputer();
260          playAgain();
261      }
262      return printStats();
263  }
264
265  /*
266   * playAgainstComputer()
267   *
268   * @return String
269   *
270   */
271  public static void playAgainstComputer() {
272      String player1 = readUserChoice();
273      String player2 = generateComputerChoice();
274      System.out.println("Computer chose - " + player2);
275      String winner = determineWinner(player1, player2);
276
277      System.out.println(winner);
278  }
279
280  /*
281   * printInstruction() - returns instruction
282   *
283   * @return String
284   */
285  public String printInstruction() {
286      String instruction = "Instruction!!!\nPlease type \n1 if you want play against Computer\n2 "
287          + "to play against another Player \n3 to Computer play against Computer!\n";
288      return instruction;
289  }
290  }
291
292 }
293

```