```java
//  Kevin Andrew Hance
//  March 8th, 2018
//  CPSC 224: Object Oriented Programming (Zhang)
//  HW Assignment #5: Yahtzee
//
//  This program emulates the popular dice game, Yahtzee. The program rolls dice for the user,
//  displays the output, then allows more chances to re-roll undesired dice before showing the
//  user what their options are for scoring in Yahtzee with that specific hand of dice. Then,
//  the user can choose which category they wish to score in. At the end of the game, the
//  scoreboard is printed out for the user to see, bonuses included.


import java.io.*;

public class LizardSpockYahtzee {

    public static void main(String args[]) throws FileNotFoundException {

        //initialize default game settings incase file isn't read correctly
        int handSize = 5;
        int maxDieValue = 6;
        int maxRolls = 3;


        Scanner s = new Scanner(System.in);

        char playAgain = 'y';

        int[] cfgArr = runInitialization();
        maxDieValue = cfgArr[0];
        handSize = cfgArr[1];
        maxRolls = cfgArr[2];

        Hand h = new Hand(handSize, maxDieValue);
        Scoreboard scr = new Scoreboard(maxDieValue, h);

        for(int i = 0; i < 13 + (maxDieValue - 6); i++)
        {
            String keep = "";
            for(int j = 0; j < handSize; j++) {
                keep += "n";
            }
            //setup to roll all dice in the first roll
            int turn = 1;
            while (turn < maxRolls && keep.contains("n"))
            {
                //roll dice not kept
                h.rollDice(keep);
                //output roll
                System.out.print("Your roll was: ");
                h.printHand();
                //if not the last roll of the hand prompt the user for dice to keep
                if(turn < maxRolls)
                {
                    do
                    {
                        System.out.print("enter dice to keep (y or n) ");
```

```
60                      keep = s.nextLine();
61                  } while (keep.length() !=handSize);
62              }
63              System.out.println();
64              turn++;
65          }
66
67          // start scoring:
68          // hand needs to be sorted to check for straights
69          h.sortHand();
70
71          // output sorted hand
72          System.out.print("Here is your sorted hand : ");
73          h.printHand();
74
75          //scr.print out and then ask the user what they want to put the score in, then
  repeat
76
77          boolean repeatScoreChoice = false;
78          do {
79              // print out all of the options the user has for scoring
80              scr.printScoreboard(maxDieValue, h);
81              // take in user input for what category they want to score in
82              char scoreOption = s.nextLine().toCharArray()[0];
83              // score option int converts a 1-digit numeric character into the integr
  equivalent
84              int scoreOptionInt = ((int) scoreOption) - 48;
85              if((int) scoreOption < 58) {
86                  // if the user entered an integer (Upper Scoreboard, Aces - Max Die Value)
87                  if(scr.isScoreEmptyUpper(scoreOptionInt-1)) {
88                      scr.bigDieScore(scoreOptionInt);
89                      repeatScoreChoice = false;
90                  } else {
91                      System.out.println("Score has already been used");
92                      repeatScoreChoice = true;
93                  }
94              } else { // if the user entered a character (Lower Scoreboard)
95                  if(scoreOption == 'A') {
96                      if(scr.isScoreEmptyLower(0)) {
97                          scr.threeOfaKindScore();
98                          repeatScoreChoice = false;
99                      } else {
100                         System.out.println("Score has already been used");
101                         repeatScoreChoice = true;
102                     }
103                 }
104                 else if(scoreOption == 'B') {
105                     if(scr.isScoreEmptyLower(1)) {
106                         scr.fourOfaKindScore();
107                         repeatScoreChoice = false;
108                     } else {
109                         System.out.println("Score has already been used");
110                         repeatScoreChoice = true;
111                     }
112                 }
113                 else if(scoreOption == 'C') {
114                     if(scr.isScoreEmptyLower(2)) {
```

```
115                             scr.fullHouseScore();
116                             repeatScoreChoice = false;
117                         } else {
118                             System.out.println("Score has already been used");
119                             repeatScoreChoice = true;
120                         }
121                     }
122                     else if(scoreOption == 'D') {
123                         if(scr.isScoreEmptyLower(3)) {
124                             scr.smStraightScore();
125                             repeatScoreChoice = false;
126                         } else {
127                             System.out.println("Score has already been used");
128                             repeatScoreChoice = true;
129                         }
130                     }
131                     else if(scoreOption == 'E') {
132                         if(scr.isScoreEmptyLower(4)) {
133                             scr.lgStrightScore();
134                             repeatScoreChoice = false;
135                         } else {
136                             System.out.println("Score has already been used");
137                             repeatScoreChoice = true;
138                         }
139                     }
140                     else if(scoreOption == 'F') {
141                         if(scr.isScoreEmptyLower(5)) {
142                             scr.yahtzeeScore();
143                             repeatScoreChoice = false;
144                         } else {
145                             System.out.println("Score has already been used");
146                             repeatScoreChoice = true;
147                         }
148                     }
149                     else if(scoreOption == 'G') {
150                         if(scr.isScoreEmptyLower(6)) {
151                             scr.chanceScore();
152                             repeatScoreChoice = false;
153                         } else {
154                             System.out.println("Score has already been used");
155                             repeatScoreChoice = true;
156                         }
157                     }
158                 }
159             } while (repeatScoreChoice);
160 
161         }
162 
163         // apply bonuses for upper scoreboard and extra yahtzees, then print out final
    scoreboard
164         scr.applyUpperBonus();
165         scr.applyYahtzeeBonus();
166         scr.printFinalScore();
167     }
168 
169     public static int[] runInitialization() throws FileNotFoundException {
170         // I know this is a bad way to access the file because it only works on my computer,
```

```
   but for some reason
171         // referencing "LizardSpockYahtzee\\src\\yahtzeeConfig.txt" or just
   "yahtzeeConfig.txt" caused a
172         // FileNotFoundException.
173         File f = new File("C:\\Users\\kevin\\eclipse-workspace\\LizardSpockYahtzee\\src\
   \yahtzeeConfig.txt");
174         //config game settings from file, give user option to change settings via console
   after file reading
175         int[] cfgArr = configureFromFile(f);
176         return cfgArr;
177     }
178 
179     public static int[] configureFromFile(File f) throws FileNotFoundException {
180         Scanner cfg = new Scanner(f);
181         // return an int array to main to configure settings
182         int[] cfgArr = {5,6,3};
183         // fill int array with values from file
184         if(cfg.hasNextInt())
185         {
186             cfgArr[0] = cfg.nextInt();
187             if(cfg.hasNextInt()) {
188                 cfgArr[1] = cfg.nextInt();
189                 if(cfg.hasNextInt()) {
190                     cfgArr[2] = cfg.nextInt();
191                 }
192             }
193         }
194         // let the user know what settings were specified in the file,
195         // then allow them to change it if they wish
196         Scanner console = new Scanner(System.in);
197         System.out.println("You will be playing with " + cfgArr[1] + " " + cfgArr[0] + "-sided
   dice.");
198         System.out.println("You will get " + cfgArr[2] + " rolls per hand.\n");
199         System.out.print("Enter 'y' if you would like to change the configuration: ");
200         // set changeConfigChar to first character entered by user (they could enter "yes" or
   "YEAH and it'd still work)
201         char changeConfigChar = console.nextLine().toLowerCase().toCharArray()[0];
202         if(changeConfigChar == 'y')
203         {
204             return configureFromUserInput();
205         }
206         return cfgArr;
207     }
208 
209 
210     // simple function to return int array based on user input representing game settings
211     public static int[] configureFromUserInput() {
212         Scanner c = new Scanner(System.in);
213         String str1 = "";
214         String str2 = "";
215         String str3 = "";
216         int[] cfgArr = {5,6,3};
217         System.out.print("Enter the number of sides on each die: ");
218         str1 = c.nextLine();
219         cfgArr[0] = Integer.parseInt(str1);
220         System.out.print("Enter the number of dice in play: ");
221         str2 = c.nextLine();
```

```
222         cfgArr[1] = Integer.parseInt(str2);
223         System.out.print("Enter the number of rolls per hand: ");
224         str3 = c.nextLine();
225         cfgArr[2] = Integer.parseInt(str3);
226         System.out.println();
227         return cfgArr;
228     }
229
230 }
```