```
 1 /**
 2  * A game of pig.
 3  * This includes all the functionality needed for a 2-die game of pig.
 4  * This is to be manipulate by {@code PigGUI} or run autonomously with the
 5  * {@code autoGame()} method.
 6  *
 7  * Pig Rules:
 8  *  -A player rolls 2 die until they decide to hold or they roll 1 or 2 ones
 9  *  -If they roll a single 1, they lose all of the current rounds score
10  *      and the turn ends
11  *  -If they roll double 1s, they lose their entire score and the turn ends
12  *  -If they hold they get the current round's points added to their total
13  *      score
14  *  -The 1st player to 100 wins
15  *
16  * @author Silas Agnew
17  * @version October 29, 2017
18  */
19
20 public class PigGame
21 {
22     private static final int WINNING_SCORE = 100;
23     private GVdie die1           = null;
24     private GVdie die2           = null;
25     private int playerScore      = 0;
26     private int cpuScore         = 0;
27     private int currentRoundScore = 0;
28     private boolean playerTurn   = true;
29
30     /**
31      * Constructs a {@code PigGame} object
32      */
33     public PigGame()
34     {
35         die1 = new GVdie();
36         die2 = new GVdie();
37
38         System.out.println("Welcome to Silas's 2-die Game of Pig.");
39     }
40
41     //-Accessors-----------------------------------------------------//
42
43     /**
44      * @return Player's score
45      */
46     public int getPlayerScore() { return playerScore; }
47
48     /**
49      * @return CPU's score
50      */
51     public int getComputerScore() { return cpuScore; }
52
53     /**
54      * @return Score of the current round
55      */
56     public int getCurrentRoundScore() { return currentRoundScore; }
```

```
 57
 58      /**
 59       * @return If it is the player's turn
 60       */
 61      public boolean isPlayerTurn() { return playerTurn; }
 62
 63      /**
 64       * @return If the player has won the game
 65       */
 66      public boolean playerWon()
 67      {
 68          return playerScore + (playerTurn ? currentRoundScore : 0) >= WINNING_SCORE;
 69      }
 70
 71      /**
 72       * @return If the CPU has won the game
 73       */
 74      public boolean computerWon()
 75      {
 76          return cpuScore + (!playerTurn ? currentRoundScore : 0) >= WINNING_SCORE;
 77      }
 78
 79      /**
 80       * Gets the die associated with the number passed
 81       * To avoid errors, {@code num <= 1} is die 1 and {@code num >= 2} is die 2
 82       * @param num Number associated with the die
 83       * @return The die
 84       */
 85      public GVdie getDie(int num) { return num >= 2 ? die2 : die1; }
 86
 87      /**
 88       * Calls {@link PigGame#rollDice()} method for the turn then accumulates the
 89       * score in the case of single or double 1s or victory
 90       */
 91      public void playerRolls()
 92      {
 93          playerTurn = true;
 94          rollDice();
 95          System.out.println(die1.getValue() + " " + die2.getValue() +
 96                  " Round Score: " + currentRoundScore);
 97
 98          if (!playerTurn || playerWon())
 99          {
100              playerScore += currentRoundScore;
101              currentRoundScore = 0;
102              playerTurn = false;
103              System.out.println("---- Your Score: " + playerScore + "\n");
104          }
105
106          if (playerWon())
107          {
108              playerScore += currentRoundScore;
109              System.out.println("You Won!");
110          }
111      }
112
```

```
113     /**
114      * Combines current and total scores and ends the players turn
115      */
116     public void playerHolds()
117     {
118         playerScore += currentRoundScore;
119         currentRoundScore = 0;
120         playerTurn = false;
121         System.out.println("---- Your Score: " + playerScore + "\n");
122     }
123 
124     /**
125      * Simulates a player's turn
126      * It will hold at 19
127      * @see PigGame#playerRolls() playerRolls
128      * @see PigGame#playerHolds() playerHolds
129      */
130     public void computerTurn()
131     {
132         playerTurn = false;
133         do {
134             rollDice();
135             System.out.println(die1.getValue() + " " + die2.getValue() +
136                     " Round Score: " + currentRoundScore);
137         } while (!playerTurn && currentRoundScore <= 19 && !computerWon());
138 
139         playerTurn = true;
140         cpuScore += currentRoundScore;
141         currentRoundScore = 0;
142         System.out.println("---- CPU Score: " + cpuScore + "\n");
143 
144         if (computerWon())
145         {
146             cpuScore += currentRoundScore;
147             System.out.println("CPU Won!");
148         }
149     }
150 
151     /**
152      * Restart the game by resetting all scores and die
153      */
154     public void restart()
155     {
156         die1.setBlank();
157         die2.setBlank();
158         playerScore = 0;
159         cpuScore = 0;
160         currentRoundScore = 0;
161         playerTurn = true;
162     }
163 
164     //-Helper/Test Methods---------------------------------//
165 
166     /**
167      * Rolls the die and handles 1s and computes score
168      */
```

```
169     private void rollDice()
170     {
171         die1.roll();
172         die2.roll();
173 
174         if (die1.getValue() == 1 && die2.getValue() == 1)
175         {
176             currentRoundScore = 0;
177             playerScore = 0;
178             playerTurn = !playerTurn;
179         }
180         else if (die1.getValue() == 1 || die2.getValue() == 1)
181         {
182             currentRoundScore = 0;
183             playerTurn = !playerTurn;
184         }
185         else
186             currentRoundScore += die1.getValue() + die2.getValue();
187     }
188 
189     /**
190      * Simulates a players turn
191      */
192     private void playerTurn()
193     {
194         while (playerTurn)
195         {
196             if (currentRoundScore < 19)
197                 playerRolls();
198             else playerHolds();
199         }
200     }
201 
202     /**
203      * Resets and runs a full game
204      */
205     public void autoGame()
206     {
207         int turns = 0;
208         restart();
209 
210         while (true)
211         {
212             turns++;
213             System.out.println("===== Turn " + turns + " =====");
214             playerTurn();
215             if (playerWon()) break;
216 
217             computerTurn();
218             if (computerWon()) break;
219         }
220         System.out.println("Total Turns: " + turns);
221     }
222 }
```