```
1  import javax.swing.*;
2  /**
3   * Bowling class for Unit 03 Lab 14: Bowling. The Bowling class allows a
4   * single user to keep track of their scores in ten-pin bowling. The user
5   * is welcomed to the Bowling with a dialog box, and is prompted to enter
6   * their bowling scores to a dialog box displaying a bowling frame, which
7   * ball is being thrown, and the total score. The numbers being entered
8   * must be in the interval [0,10] or the user will be prompted again and
9   * informed that the value inputed is invalid. Ten-pin bowling scoring
10  * rules are applied. After entering all scores, the user is given the
11  * total score.
12  *
13  * @author Josh Ibad
14  * @author James Park
15  * @version 13 December 2017
16  * @teacher Coglianese
17  * @period 2
18  */
19 public class Bowling extends JPanel
20 {
21     private int frame, ball, pins1, pins2, score;
22     private boolean spare, strike, strikeStreak;
23     /**
24      * Constructor for the Bowling class. All private variables are
25      * instantiated. The user is prompted with a message dialog box to
26      * enter their scores. Ten-pin bowling scoring rules are used. After
27      * all scores are entered, the user is shown their final score.
28      */
29     public Bowling()
30     {
31         pins1 = pins2 = score = 0;
32         frame = ball = 1;
33         spare = strike = strikeStreak = false;
34         JOptionPane.showMessageDialog(this,
35             "Welcome to \"Computer Science Bowling\"");
36         scoreGame();
37         bonusShots();
38         JOptionPane.showMessageDialog(this,
39             "Finished!\nScore " + score);
40     }
41     /**
42      * Keeps track of the score of the user according to ten-pin bowling
43      * rules. The user enters the number of pins which they knocked down
44      * with each ball, using a maximum of two balls per frame, for ten
45      * frames. For each pin knocked down, a point is awarded. If a user
46      * knocks down ten pins with one ball, they gain a strike, and the
47      * scores of the next two balls are counted twice. If a user knocks
48      * down ten pins with two balls in a frame, they gain a spare, and
49      * the score of the next ball is counted twice.
```

```
50      */
51     public void scoreGame()
52     {
53         while(frame <= 10)
54         {
55             firstShot();
56             if(spare)
57             {
58                 score += pins1;
59                 spare = false;
60             }
61             if(strike)
62                 score += pins1;
63             if(strikeStreak)
64             {
65                 score += pins1;
66                 strikeStreak = false;
67             }
68 
69             if(pins1<10)
70             {
71                 ball++;
72                 secondShot();
73                 if(strike)
74                 {
75                     score += pins2;
76                     strike = false;
77                 }
78                 ball = 1;
79                 if(pins1+pins2 == 10)
80                     spare = true;
81             }
82             else
83             {
84                 if(strike)
85                     strikeStreak = true;
86                 strike = true;
87             }
88             frame++;
89         }
90     }
91     /**
92      * Runs after the tenth frame, allows the user to record the
93      * scores from a bonus shot resulting from a strike or a
94      * spare gained in the tenth frame. According to ten-pin
95      * bowling rules, a spare in the tenth frame gives the user
96      * one bonus shot, while a strike gives the user two bonuses.
97      */
98     public void bonusShots()
```

```
99      {
100         if(spare || strike)
101         {
102             firstShot();
103         }
104         if(strike)
105         {
106             secondShot();
107         }
108     }
109     /**
110      * Records the bare score from the first shot in a frame.
111      * Prompts the user with an appropriate message generated
112      * by the message method and checks the validity of the
113      * inputed value, that is, that the value inputed is in the
114      * interval [0,10]. Does not yet count strikes and spares.
115      */
116     public void firstShot()
117     {
118         pins1 = Integer.parseInt(
119             JOptionPane.showInputDialog(message(false)));
120         while(pins1<0 || pins1>10)
121         {
122             pins1 = Integer.parseInt(
123                 JOptionPane.showInputDialog(message(true)));
124         }
125         score += pins1;
126     }
127     /**
128      * Records the bare score from the second shot in a frame.
129      * Prompts the user with an appropriate message generated
130      * by the message method and checks the validity of the
131      * inputed value, that is, that the value inputed is in the
132      * interval [0,9]. Does not yet count strikes and spares.
133      */
134     public void secondShot()
135     {
136         pins2 = Integer.parseInt(
137             JOptionPane.showInputDialog(message(false)));
138         while(pins2<0 || pins2>10)
139         {
140             pins2 = Integer.parseInt(
141                 JOptionPane.showInputDialog(message(true)));
142         }
143         score += pins2;
144     }
145     /**
146      * Creates the appropriate message to be displayed by the input
147      * dialog box composing of the total scores, the frame number, and
```

```
148      * ball number. Also, if the previous entry was invalid, the user
149      * will be informed that the previous entry was "Invalid!"
150      *
151      * @param invalid   Boolean value of whether or not the previous
152      *                  user entry was valid, in order to inform the
153      *                  user that the previous entry was invalid.
154      */
155     public String message(boolean invalid)
156     {
157         String message = "";
158         if(invalid)
159             message += "Invalid!\n";
160         message += "Score " + score;
161         message += "\nFrame " + frame;
162         message += ", Ball " + ball;
163         return message;
164     }
165     /**
166      * Creates an instance of the bowling class. The user may engage in a
167      * series of input and output interactions in order to keep track
168      * of their ten-pin bowling scores.
169      *
170      * @param args  An array of string values which is ran by the program
     .
171      */
172     public static void main(String[] args)
173     {
174         Bowling bowling = new Bowling();
175     }
176 }
177
```