```java
import java.util.ArrayList;
//Create array of players

/**
 *
 * @author Garrett Van Beek
 * Overall object for program to calculate payments after a poker game.
 */
public class PokerPayout {
    static ArrayList<player> Game = new ArrayList<player>();
    static public double initialTotal = 0;

    //Initialize Scanner
    static Scanner scnr1 = new Scanner(System.in);

    //Menu Methods
    /**
     * Creates a player object by reading input from the user.
     */
    public static void addPlayer() {

        System.out.println("Enter player name:");
        String tempname = scnr1.next();

        System.out.println("Enter player buy-in:");
        double buyin = scnr1.nextInt();

        player Player1 = new player(tempname, buyin);
        Game.add(Player1);

        initialTotal += Player1.getBalance();

    }
    /**
     * Changes the buy-in of an existing player.
     */
    public static void buyInPlayer() {
        System.out.println("Enter player name:");
        String tempname = scnr1.next();
        for (int i = 0; i < Game.size(); i++) {
            if (i == Game.size()) {
```

```
44                  System.out.println("Player not found");
45              }
46
47              if (Game.get(i).getName().equals(tempname)) {
48                  System.out.println("Enter additional buy in (do
  not enter total money player has spent");
49                  double buyin = scnr1.nextInt();
50                  Game.get(i).setBalance(buyin +
  Game.get(i).getBalance());
51              }
52          }
53      }
54      /**
55       * Prints a list of players alongside their balances.
56       */
57      public static void printPlayers() {
58          for (int i = 0; i < Game.size(); i++) {
59              System.out.println(Game.get(i).getName() + ": $" +
  Game.get(i).getBalance());
60          }
61      }
62      /**
63       * Calculates how much players who lost money should pay to
  players who earned money.
64       * Prints instructions to pay out players in the game
  properly.
65       */
66      public static void payoutPlayers() {
67          double finalTotal = 0;
68
69          for (int i = 0; i < Game.size(); i++) {
70              System.out.println("Enter " + Game.get(i).getName()
  +"'s final balance");
71              Game.get(i).finalBalance = scnr1.nextDouble();
72
73              finalTotal += Game.get(i).getFinalBalance();
74          }
75
76          //Makes sure finalTotal is equal to initialTotal
77          System.out.println(finalTotal + "==" +
  initialTotal);//helper print
78          if (finalTotal != initialTotal) {
```

```
79              System.out.println("Initial total money does not
    match final total money.");
80              menu();
81          }
82
83          //Seperates players into array of players who won money
    and lost money
84          ArrayList<player> losers = new ArrayList<player>();
85          ArrayList<player> winners = new ArrayList<player>();
86
87          for (int i = 0; i < Game.size(); i++) {
88              Game.get(i).difference = (Game.get(i).finalBalance -
    Game.get(i).balance);
89              if (Game.get(i).finalBalance > Game.get(i).balance)
    {
90                  winners.add(Game.get(i));
91              }
92              else if (Game.get(i).finalBalance <
    Game.get(i).balance) {
93                  losers.add(Game.get(i));
94              }
95
96          }
97          System.out.println("WINNERS");
98          for (int i =0; i < winners.size(); i++) {
99              System.out.println(winners.get(i).getName() +
100                     " " + winners.get(i).getFinalBalance() +
101                     " " + winners.get(i).getDifference());
102         }
103         System.out.println("LOSERS");
104         for (int i =0; i < losers.size(); i++) {
105             System.out.println(losers.get(i).getName() +
106                     " " + losers.get(i).getFinalBalance() +
107                     " " + losers.get(i).getDifference());
108         }
109
110         //Indicates how much each loser should pay each winner.
111         System.out.println("OWED");//helper print statement
112         for (int i = 0; i < winners.size(); i++) {
113             winners.get(i).owed = (-1.0) *
    (winners.get(i).getDifference());
114             System.out.println(winners.get(i).getName() + " " +
```

```
winners.get(i).getOwed());
115             }
116             for (int i = 0; i < losers.size(); i++) {
117                 losers.get(i).owed = (-1) *
(losers.get(i).getDifference());
118                 System.out.println(losers.get(i).getName() + " " +
losers.get(i).getOwed());
119             }
120
121             //iterate through winners array and take money from
losers array. Edits the owed value
122             for (int i = 0; i < winners.size(); i++) {
123                 if (winners.get(i).getOwed() < 0) {
124                     for(int j = 0; j < losers.size(); j++) {
125                         //if losers owe money and winners need money
126                         if (   (losers.get(j).getOwed() > 0) &&
(winners.get(i).getOwed() < 0)   ) {
127                             //if loser owes more money than the
winner made
128                             if(losers.get(j).getOwed() >=
Math.abs(winners.get(i).getOwed())  ) {
129                                 System.out.println(losers.get(j).get
Name() + " pay " +
130                                         Math.abs(winners.get(i).getO
wed()) + " to " + winners.get(i).getName());
131
132                                 losers.get(j).owed +=
winners.get(i).getOwed();
133                                 winners.get(i).owed -=
winners.get(i).getOwed();
134
135                                 System.out.println(winners.get(i).ge
tName() + winners.get(i).getOwed());
136                                 System.out.println(losers.get(j).get
Name() + losers.get(j).getOwed());
137
138                                 break;
139                             }
140
141                             //if loser owes less money than the
winner made
142                             else if(   losers.get(j).getOwed() <
```

```
   Math.abs(winners.get(i).getOwed())  ) {
143                             System.out.println(losers.get(j).get
   Name() + " pay " +
144                                     losers.get(j).getOwed() + "
   to " + winners.get(i).getName());
145
146                             winners.get(i).owed +=
   losers.get(j).getOwed();
147                             losers.get(j).owed -=
   losers.get(j).getOwed();
148
149                             System.out.println(winners.get(i).ge
   tName() + winners.get(i).getOwed());
150                             System.out.println(losers.get(j).get
   Name() + losers.get(j).getOwed());
151                         }
152                     }
153                 }
154             }
155         }
156         System.out.println("CALCULATIONS COMPLETE");
157     }
158     /**
159      * Removes an existing player from the game.
160      */
161     public static void removePlayer() {
162         System.out.println("Enter player to remove (Do not use
   this unless you erroneously entered a player)");
163         String tempname = scnr1.next();
164         for (int i = 0; i < Game.size(); i++) {
165             if (Game.get(i).getName().contentEquals(tempname)) {
166                 Game.remove(i);
167                 System.out.println(tempname + " has been
   removed.");
168             }
169             else {
170                 System.out.println("Player not found");
171             }
172         }
173     }
174     /**
175      * Opens a text-based interface to operate the program.
```

```
176      */
177     public static void menu() {
178
179         System.out.println("MENU");
180         System.out.println(
181                   "Add a new player -------- n\n"
182                 + "Buy-in existing player -- b\n"
183                 + "Show players ------------ s\n"
184                 + "Pay-out players --------- p\n"
185                 + "Remove player ------------r\n"
186                 + "Quit -------------------- q\n");
187
188         char choice = scnr1.next().charAt(0);
189
190         if (choice == 'n') {
191             PokerPayout.addPlayer();
192             menu();
193         }
194         else if (choice == 'b') {
195             PokerPayout.buyInPlayer();
196             menu();
197
198         }
199
200         else if (choice == 's') {
201             PokerPayout.printPlayers();
202             menu();
203         }
204         else if (choice == 'p') {
205             PokerPayout.payoutPlayers();
206             menu();
207         }
208         else if (choice == 'r') {
209             PokerPayout.removePlayer();
210
211             menu();
212         }
213         else if (choice == 'q') {
214             return;
215         }
216         else {
217             menu();
```

```
218         }
219     }
220
221
222     public static void main(String[] args) {
223         PokerPayout.menu();
224         scnr1.close();
225     }
226
227
228
229 }
230
```