1import java.util.ArrayList;

4//Create array of players

5

6/\*\*

1. \*
2. \* @author Garrett Van Beek
3. \* Overall object for program to calculate payments after a poker game.
4. \*/

11public class PokerPayout {

1. static ArrayList<player> Game = new ArrayList<player>();
2. static public double initialTotal = 0;

14

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

17

1. //Menu Methods
2. /\*\*
3. \* Creates a player object by reading input from the user.
4. \*/
5. public static void addPlayer() {

23

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

26

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

29

# 30 player Player1 = new player(tempname, buyin); 31 Game.add(Player1);

32

33 initialTotal += Player1.getBalance();

34}

1. /\*\*
2. \* Changes the buy-in of an existing player.
3. \*/
4. public static void buyInPlayer() {
5. System.out.println("Enter player name:");
6. String tempname = scnr1.next();
7. for (int i = 0; i < Game.size(); i++) {
8. if (i == Game.size()) {
9. System.out.println("Player not found");
10. }

46

1. if (Game.get(i).getName().equals(tempname)) {
2. System.out.println("Enter additional buy in (do not enter total money player has spent");
3. double buyin = scnr1.nextInt();
4. Game.get(i).setBalance(buyin +

Game.get(i).getBalance());

1. }
2. }
3. }
4. /\*\*
5. \* Prints a list of players alongside their balances.
6. \*/
7. public static void printPlayers() {
8. for (int i = 0; i < Game.size(); i++) {
9. System.out.println(Game.get(i).getName() + ": $" + Game.get(i).getBalance());
10. }
11. }
12. /\*\*
13. \* Calculates how much players who lost money should pay to players who earned money.
14. \* Prints instructions to pay out players in the game properly.
15. \*/
16. public static void payoutPlayers() {
17. double finalTotal = 0;

68

1. for (int i = 0; i < Game.size(); i++) {
2. System.out.println("Enter " + Game.get(i).getName() +"'s final balance");
3. Game.get(i).finalBalance = scnr1.nextDouble();

72

1. finalTotal += Game.get(i).getFinalBalance();
2. }

75

1. //Makes sure finalTotal is equal to initialTotal
2. System.out.println(finalTotal + "==" + initialTotal);//helper print
3. if (finalTotal != initialTotal) {
4. System.out.println("Initial total money does not match final total money.");
5. menu();
6. }

82

1. //Seperates players into array of players who won money and lost money
2. ArrayList<player> losers = new ArrayList<player>();
3. ArrayList<player> winners = new ArrayList<player>();

86

1. for (int i = 0; i < Game.size(); i++) {
2. Game.get(i).difference = (Game.get(i).finalBalance - Game.get(i).balance);
3. if (Game.get(i).finalBalance > Game.get(i).balance) {

# 90 winners.add(Game.get(i));

1. }
2. else if (Game.get(i).finalBalance <

Game.get(i).balance) {

# 93 losers.add(Game.get(i));

94 }

95

1. }
2. System.out.println("WINNERS");

# 98 for (int i =0; i < winners.size(); i++) {

1. System.out.println(winners.get(i).getName() +
2. " " + winners.get(i).getFinalBalance() +
3. " " + winners.get(i).getDifference());
4. }
5. System.out.println("LOSERS");

# 104 for (int i =0; i < losers.size(); i++) {

1. System.out.println(losers.get(i).getName() +
2. " " + losers.get(i).getFinalBalance() +
3. " " + losers.get(i).getDifference());
4. }

109

1. //Indicates how much each loser should pay each winner.
2. 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());

1. System.out.println(winners.get(i).getName() + " " + winners.get(i).getOwed());
2. }

# 116 for (int i = 0; i < losers.size(); i++) { 117 losers.get(i).owed = (-1) \*

(losers.get(i).getDifference());

1. System.out.println(losers.get(i).getName() + " " + losers.get(i).getOwed());
2. }

120

1. //iterate through winners array and take money from losers array. Edits the owed value
2. for (int i = 0; i < winners.size(); i++) {
3. if (winners.get(i).getOwed() < 0) {

# 124 for(int j = 0; j < losers.size(); j++) {

1. //if losers owe money and winners need money
2. if ( (losers.get(j).getOwed() > 0) &&

(winners.get(i).getOwed() < 0) ) {

1. //if loser owes more money than the winner made
2. if(losers.get(j).getOwed() >=

Math.abs(winners.get(i).getOwed()) ) {

1. System.out.println(losers.get(j).get

Name() + " pay " +

1. 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

1. System.out.println(winners.get(i).ge tName() + winners.get(i).getOwed());
2. System.out.println(losers.get(j).get

Name() + losers.get(j).getOwed());

137

1. break;
2. }

140

1. //if loser owes less money than the winner made
2. else if( losers.get(j).getOwed() < Math.abs(winners.get(i).getOwed()) ) {
3. System.out.println(losers.get(j).get

Name() + " pay " +

1. 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

1. System.out.println(winners.get(i).ge tName() + winners.get(i).getOwed());
2. System.out.println(losers.get(j).get

Name() + losers.get(j).getOwed());

1. }
2. }
3. }
4. }
5. }
6. System.out.println("CALCULATIONS COMPLETE");
7. }
8. /\*\*
9. \* Removes an existing player from the game.
10. \*/
11. public static void removePlayer() {
12. System.out.println("Enter player to remove (Do not use this unless you erroneously entered a player)");
13. String tempname = scnr1.next();
14. for (int i = 0; i < Game.size(); i++) {
15. if (Game.get(i).getName().contentEquals(tempname)) {
16. Game.remove(i);
17. System.out.println(tempname + " has been removed.");
18. }
19. else {
20. System.out.println("Player not found");
21. }
22. }
23. }
24. /\*\*
25. \* Opens a text-based interface to operate the program.
26. \*/
27. public static void menu() {

178

1. System.out.println("MENU");
2. System.out.println(
3. "Add a new player -------- n\n"
4. + "Buy-in existing player -- b\n"
5. + "Show players ------------ s\n"
6. + "Pay-out players --------- p\n"
7. + "Remove player ------------r\n"
8. + "Quit -------------------- q\n");

187

188 char choice = scnr1.next().charAt(0);

189

# 190 if (choice == 'n') {

1. PokerPayout.addPlayer();
2. menu();
3. }
4. else if (choice == 'b') {
5. PokerPayout.buyInPlayer();
6. menu();

197

198 }

199

1. else if (choice == 's') {
2. PokerPayout.printPlayers();
3. menu();
4. }
5. else if (choice == 'p') {
6. PokerPayout.payoutPlayers();
7. menu();
8. }
9. else if (choice == 'r') {
10. PokerPayout.removePlayer();

210

1. menu();
2. }
3. else if (choice == 'q') {
4. return;
5. }
6. else {
7. menu();
8. }
9. }

220

221

1. public static void main(String[] args) {
2. PokerPayout.menu();
3. scnr1.close();
4. }

226

227

228

229}

230