Lab 7: Classes and Objects

Thomas Bischoff

Original

Date: 4 / 16 / 2018

Problem 1: game.cpp

The function of this program is to create a game like the one created in Lab 3, which saw two six-sided dice rolled and their values combined to determine if the player has won or lost the game. This program differs from the one created in Lab 3 as it uses Classes & Objects to function.

Code 1: game.cpp 🡪 dice.h

1. #include <iostream>
2. #include <cassert>
3. #include <cstdlib>
4. #include <ctime>
5. **using** **namespace** std;
7. // to avoid inclusion of multiple definitions of the
8. // DICE class use the compiler commands below
9. #ifndef DICE
10. #define DICE
12. // definition of class Dice
13. **class** Dice
14. {
15. **private**:
17. **int** face; // can only take values between 1 and 6
18. // static int countDice = 0; // keep track of how many Dice objects were created
19. **public**:
21. // default constructor: initializes the face of a new
22. // Dice object to 1
23. Dice();
25. // specific constructor: initializes the face of a new
26. // Dice object to newFace
27. // Pre-condition: newFace is a valid number
28. // call setFace function inside Dice(int newFace)
29. Dice(**int** newFace);
31. // 10/17/14: Added copy constructor = copies the data of an existing object into a new object
32. Dice(**const** Dice &old);
34. // 10/20/14 added assignment operator
35. Dice & operator=(**const** Dice &right); // returns a reference to a dice object

38. // Sets face to the value in otherFace
39. // Pre-condition: otherFace is valid
40. **void** setFace(**int** otherFace);
42. // Changes the value of face to a random value between 1 and 6
43. **void** roll();
45. // returns the face value of a Dice object
46. **int** getFace() **const**;
48. // displays the face value of a Dice object
49. **void** display() **const**;
51. // compares the face value of the calling object with the face value
52. // of other object
53. // Dice d1, d2;
54. //  d1.display();
55. //  d1.isGreater(d2); // call isGreater: d1 = calling object d2 = passed object
57. // bool isGreater(Dice other) // pass a Dice object)
58. // other is passed by value
60. // change this to pass other by reference (passing the memory address where
61. // other is stored
62. // bool isGreater(Dice &other) // other is passed by reference
63. // &  = reference operator
65. // change it such that the object other cannot be changed inside the
66. // function isGreater
68. // bool isGreater(const Dice &other); // other is passed as a reference to a constant object --> other cannot be changed inside isGreater
70. // isGreater should not change the data of the calling object (d1)
72. **bool** isGreater(**const** Dice &other) **const**;
74. };
75. #endif

Code 1: game.cpp 🡪 dice.cpp

1. #include "Dice.h"
3. // compile the Dice.cpp after you add each function
4. // g++ -c Dice.cpp
6. // default constructor: initializes the face of a new
7. // Dice object to 1
9. Dice::Dice()
10. {
11. //cout <<  "Default constructor " << endl;
12. face = 1; // not redeclaring the data member face
13. }

16. // specific constructor: initializes the face of a new
17. // Dice object to newFace
18. // Pre-condition: newFace is a valid number
19. // call setFace function inside Dice(int newFace)
20. Dice::Dice(**int** newFace)
21. {
22. cout << "Specific constructor " << endl;
23. setFace(newFace);
24. }
26. // 10/17/14 Added definition copy constructor
27. Dice::Dice(**const** Dice &old)
28. {
29. cout << "Inside copy constructor " << endl;
30. face = old.face;
31. }
33. Dice & Dice::operator=(**const** Dice &right)
34. {
35. cout << "Inside assignment operator " << endl;
36. face = right.face;
37. **return** \***this**; // this = pointer to the calling object
38. }
40. // Sets face to the value in otherFace
41. // Pre-condition: otherFace is valid
42. **void** Dice::setFace(**int** otherFace)
43. {
44. assert(otherFace >= 1 && otherFace <= 6);
45. face = otherFace;
46. }

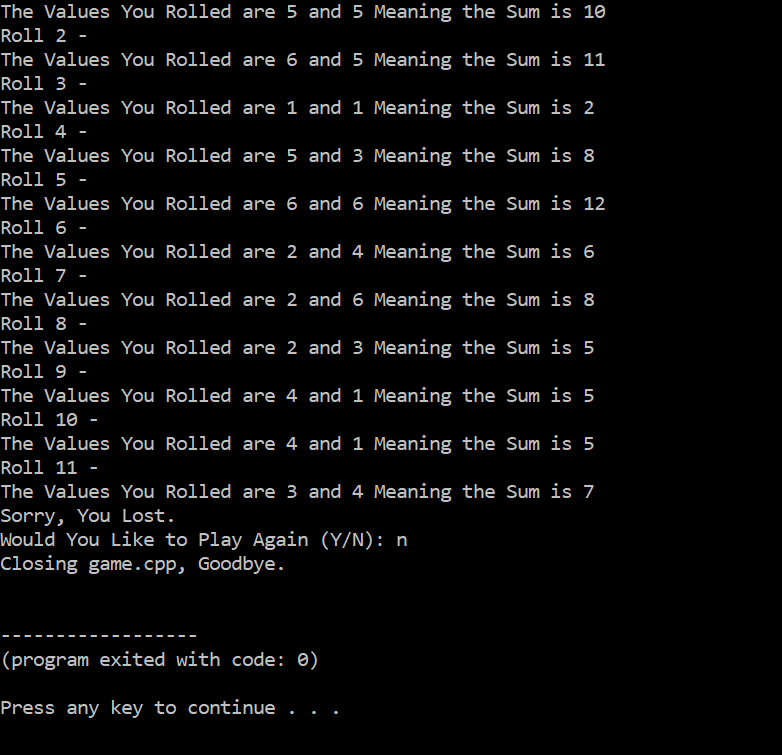
49. // Changes the value of face to a random value between 1 and 6
50. **void** Dice::roll()
51. {
52. face = rand()%6 +1;
53. }
55. // returns the face value of a Dice object
56. **int** Dice::getFace() **const**
57. {
58. **return** face;
59. }
61. // displays the face value of a Dice object
62. **void** Dice::display() **const**
63. {
64. cout << "This dice has " << face << " on top" << endl;
65. }
67. **bool** Dice::isGreater(**const** Dice &other) **const**
68. {
69. // access the face of the calling object
70. // face
71. // access the face of the object other
72. // other.face
73. // call function display() for the calling object
74. // display();
76. // call function display() for the object other
77. // other.display();
79. **bool** result;
81. **if** ( face  >  other.face)
82. result = **true**;
83. **else**
84. result = **false**;
85. **return** result;

88. }

Code 1: game.cpp 🡪 game.cpp

1. //######################################################################
2. // Program Header: game.cpp
3. // The function of this program is to play a dice game where the player
4. // "rolls" two six sided die.
5. // Author: Thomas Bischoff
6. // Date Created: 4/3/2018
7. //######################################################################
8. #include <iostream>
9. #include <cstdlib>
10. #include <string>
11. #include "Dice.h"
13. // Function Declarations
14. **int** dice\_roll();
15. **bool** move\_onto\_second\_roll(**int** value);
16. **void** follow\_up\_rolls(**int** value);
17. **bool** continue\_playing();
19. **int** main()
20. {
21. // Initialize a Statement as True
22. **bool** play\_again = **true**;
23. // Generate a WHILE Loop
24. **while** (play\_again)
25. {
26. // Display the Intro to Roll One
27. cout << "Roll 1 -" << endl;
28. // Initialize a Variable Called roll\_one and set to Hold the Value Returned by dice\_roll()
29. **int** roll\_one = dice\_roll();
30. // Check if the Game Will Continue
31. **if** (move\_onto\_second\_roll(roll\_one))
32. {
33. // Perform follow\_up\_rolls(int value)
34. follow\_up\_rolls(roll\_one);
35. }
36. // Set play\_again to Hold the Value Returned by continue\_playing()
37. play\_again = continue\_playing();
38. }
39. }
41. // Function Definition
42. **int** dice\_roll()
43. {
44. // Get Die One
45. Dice d1;
46. // Get Die Two
47. Dice d2;
48. // "Roll" the First Die
49. d1.roll();
50. // "Roll" the Second Die
51. d2.roll();
52. // Calculate the Sum of the Two Dice
53. **int** sum = d1.getFace() + d2.getFace();
54. // Display the Two Faces of Collected from the Dice and their Sum
55. cout << "The Values You Rolled are " << d1.getFace() << " and " << d2.getFace() << " Meaning the Sum is " << sum << endl;
56. // Resturn the Sum
57. **return** sum;
58. }
60. // Function Defintion
61. **bool** move\_onto\_second\_roll(**int** value)
62. {
63. // Check if the Player Won the Game
64. **if** ((value == 7) || (value == 11))
65. {
66. // Display that the Player has Won
67. cout << "Congradulations, You Win!!!" << endl;
68. // Return False
69. **return** **false**;
70. }
71. // Check if the Player Lost
72. **else** **if** ((value == 2) || (value == 3) || (value == 12))
73. {
74. // Display that the Player Lost
75. cout << "Sorry, You Lost." << endl;
76. // Return False
77. **return** **false**;
78. }
79. // Otherwise
80. **else**
81. {
82. // Return True
83. **return** **true**;
84. }
85. }
87. // Function Definitions
88. **void** follow\_up\_rolls(**int** value)
89. {
90. // Intialize the Current Roll Value as 2
91. **int** current\_roll = 2;
92. // Intialize a Statement as True
93. **bool** keep\_rolling = **true**;
94. // Generate a WHILE Loop
95. **while** (keep\_rolling)
96. {
97. // Display the Current Roll Number
98. cout << "Roll " << current\_roll << " -" << endl;
99. // Initialize a Variable Called followup\_roll and set to Hold the Value Returned by dice\_roll()
100. **int** followup\_roll = dice\_roll();
101. // Check if the Valued Rolled is the Same as the Original Value Rolled
102. **if** (followup\_roll == value)
103. {
104. // Display that the Player has Won
105. cout << "Congradulations, You Win!!!" << endl;
106. // Change keep\_rolling to be False
107. keep\_rolling = **false**;
108. }
109. // Check if the Value Rolled is a Seven
110. **else** **if** (followup\_roll == 7)
111. {
112. // Display that the Player has Lost
113. cout << "Sorry, You Lost." << endl;
114. // Change keep\_rolling to be False
115. keep\_rolling = **false**;
116. }
117. // Otherwise
118. **else**
119. {
120. // Increase current\_roll by One
121. current\_roll += 1;
122. }
123. }
124. }
126. // Function Definition
127. **bool** continue\_playing()
128. {
129. // Initilaize a Statment Called replay
130. **bool** replay;
131. // Initialize a Statement as True
132. **bool** keep\_asking = **true**;
133. // Generate a WHILE Loop
134. **while** (keep\_asking)
135. {
136. // Initialize a Variable to Take the User's Response
137. **char** users\_answer;
138. // Ask the User if they Would Like to Play Again
139. cout << "Would You Like to Play Again (Y/N): ";
140. // Take the User's Response and Set it be Held by users\_anwser
141. cin >> users\_answer;
142. // Convert User's Answer to be Uppercase
143. users\_answer = toupper(users\_answer);
144. // Check if the User Wants to Play Again
145. **if** (users\_answer == 'Y')
146. {
147. // Display that the Game is Going to Play Again
148. cout << "game.cpp Reloading..." << endl;
149. // Make keep\_asking False
150. keep\_asking = **false**;
151. // Set a replay as True
152. replay = **true**;
153. }
154. // Check if the Player Doesn't Want to Play Again
155. **else** **if** (users\_answer == 'N')
156. {
157. // Display a Goodbye Message
158. cout << "Closing game.cpp, Goodbye." << endl;
159. // Make keep\_asking False
160. keep\_asking = **false**;
161. // Set replay as False
162. replay = **false**;
163. }
164. // Otherwise
165. **else**
166. {
167. // Display that the Input was Not Recognized
168. cout << "The Response You Have Entered is Not Recognized. Please Try Again." << endl;
169. }
170. }
171. // Return keep\_playing
172. **return** replay;
173. }

Sample Output 1: game.cpp



Problem 2: testTeam.cpp

The function of this program is to allow the user to create a Team in which they can set the number of players they have, the number of wins and losses, and the ratio to wins and losses. From their it can determine what Team is statistically better.

Code 2: testTeam.cpp 🡪 Team.h

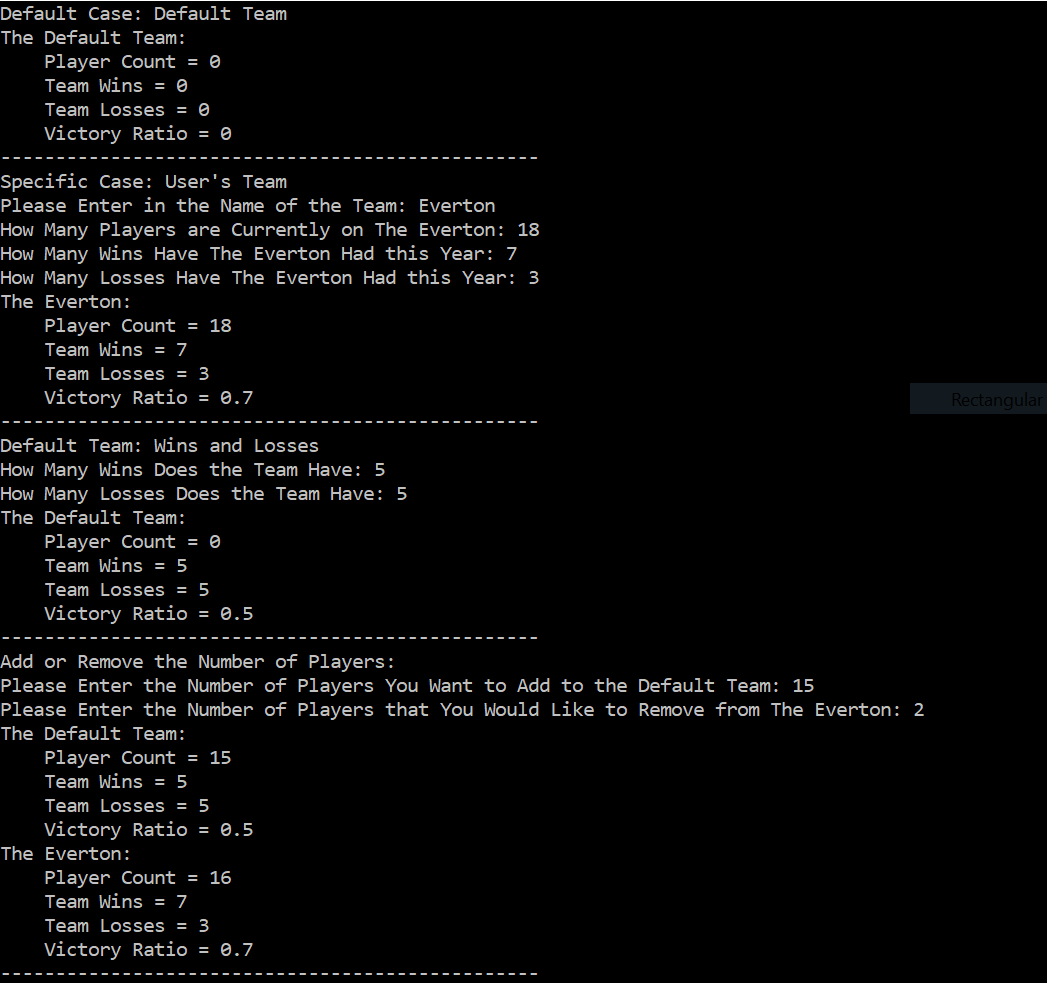
1. //######################################################################
2. // Program Header: Team.h
3. // Author: Thomas Bischoff
4. // Date Created: 4/7/2018
5. //######################################################################
7. #include <iostream>
8. #include <cassert>
9. #include <cstdlib>
10. #include <string>
11. **using** **namespace** std;
13. // In Order to Avoid Inclusion of Multiple Definitions of TEAM Class
14. #ifndef TEAM
15. #define TEAM
17. // Definition of Class Team
18. **class** Team
19. {
20. **private**:
21. // Teams Name
22. string team\_name;
23. // Number of Wins for the Team
24. **double** win\_count;
25. // Number of Losses for the Team
26. **double** loss\_count;
27. // Number of Players on Team
28. **int** player\_count;
29. // Ratio of Wins and Losses
30. **double** points;
31. **public**:
32. // Default Constructor for Teams Name and Number of Players
33. Team();
34. // Specific Constructor for the Teams Name and the Number of Players
35. Team(**const** string new\_name, **int** new\_players);
36. // Displays the Infromation About the Team
37. **void** display();
38. // Increment the Number of Wins
39. **void** add\_wins();
40. // Increment the Number of Wins
41. **void** add\_losses();
42. // Add a Number of Players
43. **void** add\_players(**int** new\_players);
44. // Remove a Number of Players
45. **void** del\_players(**int** new\_players);
46. // Set Up the Teams
47. **void** setTeam(string name, **int** players, **int** wins, **int** losses);
48. // Checks What Team is Better
49. **bool** isBetter(Team &other);
50. // Collects the Number of Wins
51. **double** getWins();
52. // Collect the Number of Losses
53. **double** getLosses();
54. // Collect the Number of Points
55. **double** getPoints();
56. };
57. #endif

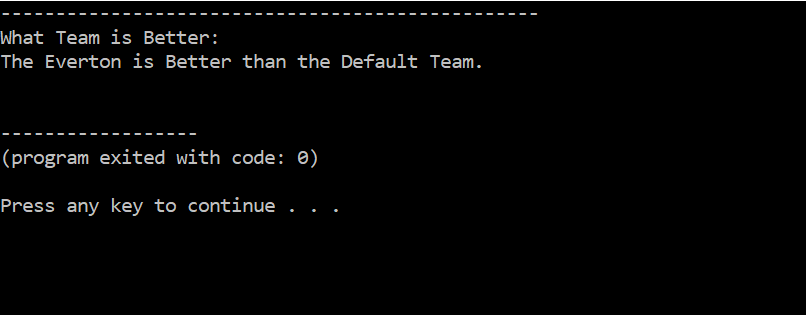
Code 2: testTeam.cpp 🡪 Team.cpp

1. //######################################################################
2. // Program Header: Team.cpp
3. // Author: Thomas Bischoff
4. // Date Created: 4/7/2018
5. //######################################################################
7. #include "Team.h"
9. // Initialize the Default Team
10. Team::Team()
11. {
12. // Place the Default Team Name
13. team\_name = "Default Team";
14. // Set Win Count to Zero
15. win\_count = 0;
16. // Set Loss Count to Zero
17. loss\_count = 0;
18. // Set Player Count to Zero
19. player\_count = 0;
20. // Set Points to Zero
21. points = 0;
22. }
24. // Initialize the Specific Team
25. Team::Team(**const** string newName, **int** newPlayers)
26. {
27. // Ask the User to Enter in the Number of Wins the Team Has
28. cout << "How Many Wins Have The " << newName << " Had this Year: ";
29. // Take the User's Input
30. cin >> win\_count;
31. // Ask the User to Enter the Number of Losses the Team Has
32. cout << "How Many Losses Have The " << newName << " Had this Year: ";
33. // Take the User's Input
34. cin >> loss\_count;
35. // Run Set Team
36. (\***this**).setTeam(newName, newPlayers, win\_count, loss\_count);
37. }
39. // Display the Information About the Team
40. **void** Team::display()
41. {
42. // Display the Team Name
43. cout << "The " << team\_name << ":" << endl;
44. // Display the Number of Players on the Team
45. cout << "    Player Count = " << player\_count << endl;
46. // Display the Number of Wins that the Team Has
47. cout << "    Team Wins = " << (\***this**).getWins() << endl;
48. // Display the Number of Losses that the Team Has
49. cout << "    Team Losses = " << (\***this**).getLosses() << endl;
50. // Display that the Ratio of Wins and Loss
51. cout << "    Victory Ratio = " << (\***this**).getPoints() << endl;
52. }
54. // Increments the Amount of Team Wins
55. **void** Team::add\_wins()
56. {
57. // Increase the Number of Wins by One
58. win\_count += 1;
59. // Adjust the Points
60. points = (win\_count) / (win\_count + loss\_count);
61. }
63. // Increments the Number of Losses by One
64. **void** Team::add\_losses()
65. {
66. // Increase the Number of Losses by One
67. loss\_count += 1;
68. // Adjust the Points
69. points = (win\_count) / (win\_count + loss\_count);
70. }
72. // Adds a Number of Players to the Team
73. **void** Team::add\_players(**int** new\_players)
74. {
75. // Increase the Nmber of Players by the Number Provided
76. player\_count += new\_players;
77. }
79. // Removes a Number of Players from the Team
80. **void** Team::del\_players(**int** new\_players)
81. {
82. // Check to See if new\_players is Greater than the
83. **if** (new\_players > player\_count)
84. {
85. // Display that the That Amount of Player Cannot be Removed
86. cout << "Sorry, but the Number of Players You are Looking to Remove Exceeds the Number of Players on the Team." << endl;
87. }
88. // Otherwise
89. **else**
90. {
91. // Remove the Number of Players from the Team
92. player\_count -= new\_players;
93. }
94. }
96. // Set up the Teams
97. **void** Team::setTeam(string str, **int** players, **int** wins, **int** losses)
98. {
99. // Set the String to be the Teams Name
100. team\_name = str;
101. // Set Number Held by Players to be Amount of Players on the Team
102. player\_count = players;
103. // Set the Number of Wins
104. win\_count = wins;
105. // Set the Number of Losses
106. loss\_count = losses;
107. // Set the Number of Points
108. points = (win\_count) / (win\_count + loss\_count);
109. }
111. // Check What Team is Better
112. **bool** Team::isBetter(Team &other)
113. {
114. // Check to See if this's Points are Greater than other's Points
115. **if** ((\***this**).getPoints() > other.getPoints())
116. {
117. // Return True
118. **return** **true**;
119. }
120. // Otherwise
121. **else**
122. {
123. // Return False
124. **return** **false**;
125. }
126. }
128. // Returns the Numbers of Wins from the Team
129. **double** Team::getWins()
130. {
131. // Return the Wins
132. **return** win\_count;
133. }
135. // Return the Number of Losses from the Team
136. **double** Team::getLosses()
137. {
138. // Return the Losses
139. **return** loss\_count;
140. }
142. // Return the Amount of Points
143. **double** Team::getPoints()
144. {
145. // Return the Points
146. **return** points;
147. }

Code 2: testTeam.cpp 🡪 testTeam.cpp

1. //######################################################################
2. // Program Header: testTeam.cpp
3. // Author: Thomas Bischoff
4. // Date Created: 4/7/2018
5. //######################################################################
7. #include "Team.h"
9. **int** main()
10. {
11. // Display this as the Default Case
12. cout << "Default Case: Default Team" << endl;
13. // Initialize the Default Team
14. Team default\_team;
15. // Display the Current Information About the Team
16. default\_team.display();
17. // Mark off the Section
18. cout << "-------------------------------------------------" << endl;
19. // Display this as the Specific Case
20. cout << "Specific Case: User's Team" << endl;
21. // Initialize the Variable to Hold the Team Name
22. string team;
23. // Ask the User to Enter in the Name of the Team
24. cout << "Please Enter in the Name of the Team: ";
25. // Take the User's Input
26. cin >> team;
27. // Initialize the Variable to Hold the Number of Members on the Team
28. **int** team\_members;
29. // Ask the User to Enter in the Number of People on the Team
30. cout << "How Many Players are Currently on The " << team << ": ";
31. // Take the User's Input
32. cin >> team\_members;
33. // Initialize the New Team
34. Team new\_team(team, team\_members);
35. // Display the Current Infromation About the Team
36. new\_team.display();
37. // Mark off the Section
38. cout << "-------------------------------------------------" << endl;
39. // Display this as the Area that Increments the Numbers of Wins and Losses for the Team
40. cout << "Default Team: Wins and Losses" << endl;
41. // Initialize a Variable for the Number of Wins
42. **int** wins;
43. // Ask the User for the Number of Wins
44. cout << "How Many Wins Does the Team Have: ";
45. // Take the User's Input
46. cin >> wins;
47. // Add the Wins to the Team
48. **for** (**int** i = 0; i < wins; i++)
49. {
50. // Increment the Number of Wins
51. default\_team.add\_wins();
52. }
53. // Initialize a Variable for the Number of Losses
54. **int** losses;
55. // Ask the User for the Number of Losses
56. cout << "How Many Losses Does the Team Have: ";
57. // Take the User's Input
58. cin >> losses;
59. // Add the Losses to the Team
60. **for** (**int** i = 0; i < losses; i++)
61. {
62. // Increment the Number of Wins
63. default\_team.add\_losses();
64. }
65. // Display the Updated Team Information
66. default\_team.display();
67. // Mark off this Section
68. cout << "-------------------------------------------------" << endl;
69. // Display this as the Section Where Team Members are Added and Removed
70. cout << "Add or Remove the Number of Players:" << endl;
71. // Initialize the Number of Players Added and Removed
72. **int** players\_added, players\_removed;
73. // Ask the User to Enter the Number of Players that they Want to Add to the Default Team
74. cout << "Please Enter the Number of Players You Want to Add to the Default Team: ";
75. // Take the User's Input
76. cin >> players\_added;
77. // Add the Players to Default Team
78. default\_team.add\_players(players\_added);
79. // Ask the User to Enter the Number of Players they Want to Remove from the New Team
80. cout << "Please Enter the Number of Players that You Would Like to Remove from The " << team << ": ";
81. // Take the User's
82. cin >> players\_removed;
83. // Remove Players from the New Team
84. new\_team.del\_players(players\_removed);
85. // Display the Information for Both Teams
86. default\_team.display();
87. new\_team.display();
88. // Mark off this Section
89. cout << "-------------------------------------------------" << endl;
90. // Display this Section as the One that Will Determine What Team is Better
91. cout << "What Team is Better:" << endl;
92. // Check if the Default Team is Better than the New Team
93. **if**(default\_team.isBetter(new\_team))
94. {
95. // Display that the Default Team is the Better Team
96. cout << "The Default Team is Better than The " << team << "." << endl;
97. }
98. // Check if the New Team is Better than the Default Team
99. **else** **if** (new\_team.isBetter(default\_team))
100. {
101. // Display that the New Team is the Better Team
102. cout << "The " << team << " is Better than the Default Team." << endl;
103. }
104. // Otherwise
105. **else**
106. {
107. // Display that the Teams are Equivalent
108. cout << "The Two Teams are Equals." << endl;
109. }
110. }

Sample Output 2: testTeam.cpp



Problem 3: testMoney.cpp

The function of this program is to allow a user to add and subtract an entered amount of from a previously initialized amount of money.

Code 3: testMoney.cpp 🡪 Money.h

1. //######################################################################
2. // Program Header: Money.h
3. // Author: Thomas Bischoff
4. // Date Created: 4/7/2018
5. //######################################################################
7. #include <iostream>
8. #include <cassert>
9. #include <cstdlib>
10. #include <string>
11. **using** **namespace** std;
13. // In Order to Avoid Inclusion of Multiple Definitions of MONEY Class
14. #ifndef MONEY
15. #define MONEY
17. // Definitions of Class Money
18. **class** Money
19. {
20. **private**:
21. // Number of Dollar Bills: >= 0
22. **int** dollars;
23. // Number of Cents: >= 0 and < 100
24. **int** cents;
25. // Total Amount of Money in Cents
26. **double** total;
27. // Second Total Amount in Cents
28. **int** second\_total;
29. **public**:
30. // Default Constructor: Sets the New Object to be $0.00
31. Money();
32. // Specific Constructor:Set the New Object to be $new\_dollars.new\_cents
33. Money(**int** new\_dollars, **int** new\_cents);
34. // Add an Amount of Money of Money Object
35. **void** add(**int** add\_dollars, **int** add\_cents);
36. **void** add(**const** Money &add\_money);
37. // Subtract an Amount of Money of Money Object
38. **void** subtract(**int** sub\_dollars, **int** sub\_cents);
39. **void** subtract(**const** Money &sub\_money);
40. // Display the Amount of Money in $dollars.cents
41. **void** display() **const**;
42. // Return the String with the Objects Data
43. string to\_string() **const**;
44. // Returns the Dollar Amount
45. **int** get\_dollars() **const**;
46. // Returns the Cent Amount
47. **int** get\_cents() **const**;
48. // Returns True if the Amount Stored in the Calling Object is Bigger or Equal to the Amount of the Other Object
49. **bool** isBigger(**const** Money &other);
50. };
51. #endif

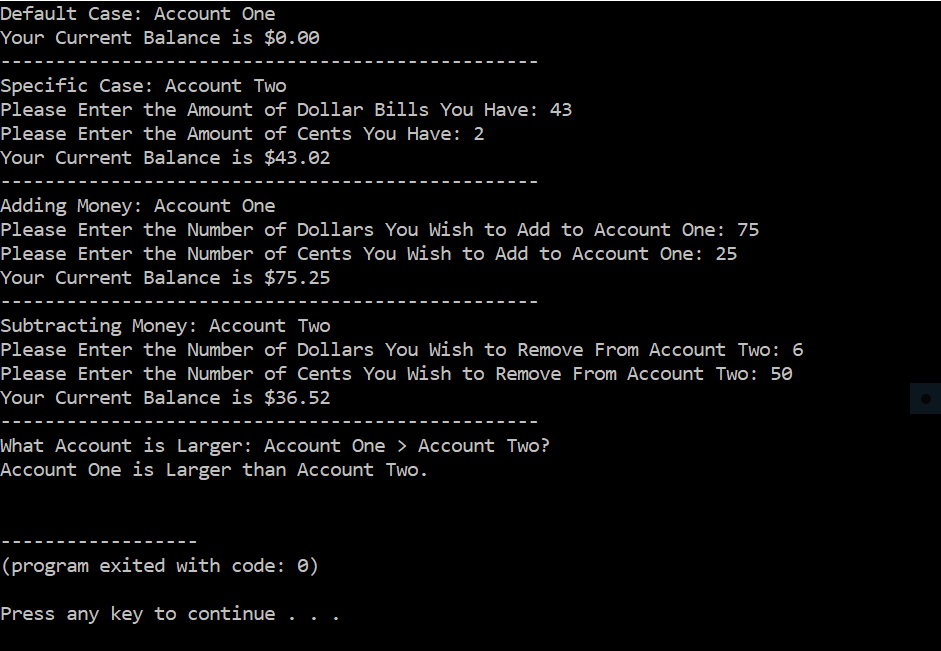
Code 3: testMoney.cpp 🡪 Money.cpp

1. //######################################################################
2. // Program Header: Money.cpp
3. // Author: Thomas Bischoff
4. // Date Created: 4/7/2018
5. //######################################################################
7. #include "Money.h"
9. // Initilaize the Amount of Money to be $0.00
10. Money::Money()
11. {
12. // Set Dollars to be '0'
13. dollars = 0;
14. // Set Cents to be '0'
15. cents = 0;
16. }
18. // Initialize the Amount of Money to be $new\_dollars.new\_cents
19. Money::Money(**int** new\_dollars, **int** new\_cents)
20. {
21. // Set Dollars to be new\_dollars
22. dollars = new\_dollars;
23. // Generate a WHILE Loop that Will Check if new\_cents is Greater Than or Equal to 100
24. **while** (new\_cents >= 100)
25. {
26. // Increase the Number of Dollars by '1'
27. dollars += 1;
28. // Decrease the Value of new\_cents by '100'
29. new\_cents -= 100;
30. }
31. // Set Cents to be new\_cents
32. cents = new\_cents;
33. }
35. **void** Money::add(**int** add\_dollars, **int** add\_cents)
36. {
37. // Set Dollars to be Their Current Value Plus add\_dollars
38. dollars += add\_dollars;
39. // Set Cents to be Their Current Value Plus add\_cents
40. cents += add\_cents;
41. // Generate a WHILE Loop that Will Check if cents is Greater Than or Equal to 100
42. **while** (cents >= 100)
43. {
44. // Increase the Number of Dollars by '1'
45. dollars += 1;
46. // Decrease the Number of Cents by '100'
47. cents -= 100;
48. }
49. // Set the New Total
50. total = dollars + (0.10 \* cents);
51. }
53. **void** Money::subtract(**int** sub\_dollars, **int** sub\_cents)
54. {
55. // Generate a WHILE Loop that Will Check if sub\_cents is Greater Than or Equal to 100
56. **while** (sub\_cents >= 100)
57. {
58. // Increase sub\_dollars by '1'
59. sub\_dollars += 1;
60. // Decrease sub\_cents by '100'
61. sub\_cents -= 100;
62. }
63. // Set the Value of Total to be Sum of Dollars Times 100 and the Number of Cents
64. total = (dollars \* 100) + cents;
65. // Set the Value of Second Total to be the Sum of sub\_dollar Times 100 and the Number of sub\_cents
66. second\_total = (sub\_dollars \* 100) + sub\_cents;
67. // Check to See if the First Total is Greater than or Equal to the Second Total
68. **if** (total >= second\_total)
69. {
70. // Set Total to be Total Minus Second Total
71. total -= second\_total;
72. // Set the Number of Dollars to be the Total Divided by 100
73. dollars = total / 100;
74. // Set the Number of Cents to be the Number of Dollars Times by 100 and Subtract the Total
75. cents = total - (dollars \* 100);
76. total = dollars + (0.10 \* cents);
77. }
78. // Otherwise
79. **else**
80. {
81. // Display that it Can't be Done
82. cout << "Sorry, but the Amount that you Entered Cannot be Removed from Your Current Amount." << endl;
83. }
84. }
86. **void** Money::display() **const**
87. {
88. // Check to See if the Value of Cents is Less than '10'
89. **if** (cents < 10)
90. {
91. // Display the Amount of Money that they Have
92. cout << "Your Current Balance is $" << dollars << ".0" << cents << endl;
93. }
94. // Otherwise
95. **else**
96. {
97. // Display the Amount of Money that they Have
98. cout << "Your Current Balance is $" << dollars << "." << cents << endl;
99. }
100. }
102. **int** Money::get\_dollars() **const**
103. {
104. // Return the Value of Dollars
105. **return** dollars;
106. }
108. **int** Money::get\_cents() **const**
109. {
110. // Return the Value of Cents
111. **return** cents;
112. }
114. **bool** Money::isBigger(**const** Money &other)
115. {
116. // Check the if Dollars in This is Greater than the Dollars in Other
117. **if** ((\***this**).get\_dollars() > other.get\_dollars())
118. {
119. // Return True
120. **return** **true**;
121. }
122. // Check if the Dollars in This is the Same as the Dollars in Other
123. **else** **if** ((\***this**).get\_dollars() == other.get\_dollars())
124. {
125. // Check if the Number of Cents in This is Greater than the Other
126. **if** ((\***this**).get\_cents() > other.get\_cents())
127. {
128. // Return True
129. **return** **true**;
130. }
131. // Otherwise
132. **else**
133. {
134. // Return False
135. **return** **false**;
136. }
137. }
138. // Otherwise
139. **else**
140. {
141. // Return False
142. **return** **false**;
143. }
144. }

Code 3: testMoney.cpp 🡪 testMoney.cpp

1. //######################################################################
2. // Program Header: testMoney.cpp
3. // Author: Thomas Bischoff
4. // Date Created; 4/7/2018
5. //######################################################################
7. #include "Money.h"
9. **int** main()
10. {
11. // Display this as the Default Case
12. cout << "Default Case: Account One" << endl;
13. // Set Balance in Account One
14. Money account\_one;
15. // Display the Current Balance in Account One
16. account\_one.display();
17. // Mark off Next Section
18. cout << "-------------------------------------------------" << endl;
19. // Display this as the Specific Case
20. cout << "Specific Case: Account Two" << endl;
21. // Initilaize the Variables for the Amount of Dollars and Cents
22. **int** amount\_of\_dollars, amount\_of\_cents;
23. // Ask the User to Input the Amount of Dollars they Have
24. cout << "Please Enter the Amount of Dollar Bills You Have: ";
25. // Take the User's Input and Set it to be Held by amount\_of\_dollars
26. cin >> amount\_of\_dollars;
27. // Ask the User to Input the Amount of Cents they Have
28. cout << "Please Enter the Amount of Cents You Have: ";
29. // Take the User's Input and Set it to be Held by amount\_of\_cents
30. cin >> amount\_of\_cents;
31. // Set Balance in Account Two
32. Money account\_two(amount\_of\_dollars, amount\_of\_cents);
33. // Display the Current Balance in Account Two
34. account\_two.display();
35. // Mark off Next Section
36. cout << "-------------------------------------------------" << endl;
37. // Display Adding Money
38. cout << "Adding Money: Account One" << endl;
39. // Ask the User to Input the Amount of Dollars they Want to Add
40. cout << "Please Enter the Number of Dollars You Wish to Add to Account One: ";
41. // Take the User's Input
42. cin >> amount\_of\_dollars;
43. // Ask the User to Input the Amount of Cents they Want to Add
44. cout << "Please Enter the Number of Cents You Wish to Add to Account One: ";
45. // Take the User's Input
46. cin >> amount\_of\_cents;
47. // Add the Money into Account One
48. account\_one.add(amount\_of\_dollars, amount\_of\_cents);
49. // Display the New Balance
50. account\_one.display();
51. // Mark off Next Section
52. cout << "-------------------------------------------------" << endl;
53. // Display Subtracting Money
54. cout << "Subtracting Money: Account Two" << endl;
55. // Ask the User to Input the Amount of Dollars they Want to Remove
56. cout << "Please Enter the Number of Dollars You Wish to Remove From Account Two: ";
57. // Take the User's Input
58. cin >> amount\_of\_dollars;
59. // Ask the User to Input the Amount of Cents they Want to Remove
60. cout << "Please Enter the Number of Cents You Wish to Remove From Account Two: ";
61. // Take the User's Input
62. cin >> amount\_of\_cents;
63. // Subtract the Money From Account Two
64. account\_two.subtract(amount\_of\_dollars, amount\_of\_cents);
65. // Display the New Balance
66. account\_two.display();
67. // Mark off the Next Section
68. cout << "-------------------------------------------------" << endl;
69. // Display Checking What Account is Larger
70. cout << "What Account is Larger: Account One > Account Two?" << endl;
71. // Check if Account One is Larger than Account Two
72. **if** (account\_one.isBigger(account\_two))
73. {
74. // Display that Account One is Larger
75. cout << "Account One is Larger than Account Two." << endl;
76. }
77. // Check if Account Two is Larger than Account One
78. **else** **if** (account\_two.isBigger(account\_one))
79. {
80. // Display that Account Two is Larger
81. cout << "Account Two is Larger than Account One." << endl;
82. }
83. // Otherwise
84. **else**
85. {
86. // Display that the Two Accounts are the Same
87. cout << "Account One and Account Two Have the Same Amount of Money." << endl;
88. }
89. }

Sample Output 3: testMoney.cpp



Problem 4: testAccounts.cpp

The function of this program is to simulate the functions of a bank account allowing the user to transfer, withdraw, and deposit money from their account.

Code 4: testAccounts.cpp 🡪 Accounts.h

1. //######################################################################
2. // Program Header: Accounts.h
3. // Author: Thomas Bischoff
4. // Date Created: 4/10/2018
5. //######################################################################
7. #include <iostream>
8. #include <cassert>
9. #include <cstdlib>
10. #include <ctime>
11. **using** **namespace** std;
13. // To Avoid Inclusion of Multiple Definitions of the Accounts Class
14. #ifndef ACCOUNTS
15. #define ACCOUNTS
17. // Definitions of Class Accounts
18. **class** Accounts
19. {
20. **private**:
21. // The Ten Digit Account String
22. **int** account\_id[10];
23. // The Amount of Dollars in the Account
24. **int** dollars;
25. // The Amount of Cents in the Account
26. **int** cents;
27. // The Total Amount of Money in the Account
28. **double** total;
29. **public**:
30. // Creates a New Account with a Random ID and a Balance of $0.00
31. Accounts();
32. // Creates a New Account with a Random ID and a Balance Provided by the User
33. Accounts(**int** start\_dollars, **int** start\_cents);
34. // Allows the User to Withdraw Money from the Account
35. **void** withdraw(**int** minus\_dollars, **int** minus\_cents);
36. // Allows the User to Deposit Money into the Account
37. **void** deposit(**int** add\_dollars, **int** add\_cents);
38. // Allows the User to Transfer Money into Another Account
39. **void** transfer\_into(Accounts &destination, **int** transfer\_dollars, **int** transfer\_cents);
40. // Allows Money to be Transfered into the User's Account
41. **void** transfer\_from(Accounts &source, **int** transfer\_dollars, **int** transfer\_cents);
42. // Checks to See if an Amount of Money can be Withdrawed or Transferred
43. **bool** is\_valid(**int** sub\_dollars, **int** sub\_cents);
44. // Display the Accounts Information
45. **void** display();
46. // Add Money to an Account
47. **void** add(**int** add\_dollars, **int** add\_cents);
48. // Subtract Money from the Account
49. **void** subtract(**int** sub\_dollars, **int** sub\_cents);
50. };
51. #endif

Code 4: testAccounts.cpp 🡪 Accounts.cpp

1. //######################################################################
2. // Program Header: Accounts.cpp
3. // Author: Thomas Bischoff
4. // Date Created: 4/11/2018
5. //######################################################################
7. #include "Accounts.h"
9. // Creates a New Account with a Random ID and a Balance of $0.00
10. Accounts::Accounts()
11. {
12. // Go Throuhg Each Digit in the ID
13. **for** (**int** i = 0; i < 10; i++)
14. {
15. // Generate a Random Number for the Digit
16. account\_id[i] = rand()% 10;
17. }
18. // Set Dollars to be '0'
19. dollars = 0;
20. // Set Cents to be '0'
21. cents = 0;
22. // Set the Total Amount of Money to be "0.00"
23. total = 0.00;
24. }
26. // Creates a New Account with a Random ID and the Balance that the User Entered
27. Accounts::Accounts(**int** start\_dollars, **int** start\_cents)
28. {
29. // Go Throuhg Each Digit in the String
30. **for** (**int** i = 0; i < 10; i++)
31. {
32. // Generate a Random Number for the Digit
33. account\_id[i] = rand()% 10;
34. }
35. // Generate a WHILE Loop
36. **while** (start\_cents >= 100)
37. {
38. // Increase the Number of Start Dollars by '1'
39. start\_dollars += 1;
40. // Decrease the Number of Start Cents by '100'
41. start\_cents -= 100;
42. }
43. // Set Dollars to be the Starting Dollars
44. dollars = start\_dollars;
45. // Set Cents to be the Starting Cents
46. cents = start\_cents;
47. // Set the Total Amount of Money
48. total = dollars + (0.01 \* cents);
49. }
51. // Allows the User to Withdraw an Amount of Money from the Account
52. **void** Accounts::withdraw(**int** minus\_dollars, **int** minus\_cents)
53. {
54. // Generate a WHILE Loop
55. **while** (minus\_cents >= 100)
56. {
57. // Increase the Number of Dollars by '1'
58. minus\_dollars += 1;
59. // Decrease the Number of Cents by '100'
60. minus\_cents -= 100;
61. }
62. // Check if the Amount Being Withdrawed
63. **if** ((\***this**).is\_valid(minus\_dollars, minus\_cents))
64. {
65. // Set the New Number of Dollars
66. dollars -= minus\_dollars;
67. // Set the New Number of Cents
68. cents -=  minus\_cents;
69. // Check if the Number of Cent is Less than '0'
70. **if** (cents < 0)
71. {
72. // Decrease the Number of Dollars by '1'
73. dollars -= 1;
74. // Increase the Number of Cents by '100'
75. cents += 100;
76. }
77. // Set the New Total
78. total = dollars + (0.01 \* cents);
79. }
80. // Otherwise
81. **else**
82. {
83. // Display that the Amount of Money Entered Exceeds the Current Balance
84. cout << "Sorry, but the Amount You are Looking Remove Exceeds Your Current Balance." << endl;
85. }
86. }
88. // Allows the User to Deposit Money into their Account
89. **void** Accounts::deposit(**int** add\_dollars, **int** add\_cents)
90. {
91. // Set the New Number of Dollars
92. dollars += add\_dollars;
93. // Set the New Number of Cents
94. cents += add\_cents;
95. // Generate a WHILE Loop
96. **while** (cents >= 100)
97. {
98. // Increase the Number of Dollars by '1'
99. dollars += 1;
100. // Decrease the Number of Cents by '100'
101. cents -= 100;
102. }
103. // Set the New Total
104. total = dollars + (0.01 \* cents);
105. }
107. // Allows the User to Transfer Their Money into Another Account
108. **void** Accounts::transfer\_into(Accounts &destination, **int** transfer\_dollars, **int** transfer\_cents)
109. {
110. // Generate a WHILE Loop
111. **while** (transfer\_cents >= 100)
112. {
113. // Increase the Number of Dollars by '1'
114. transfer\_dollars += 1;
115. // Decrease the Number of Cents by '100'
116. transfer\_cents -= 100;
117. }
118. // Check if the Amount Being Tranferred is Valid
119. **if** ((\***this**).is\_valid(transfer\_dollars, transfer\_cents))
120. {
121. // Add the Amount of Money to the Other Account
122. destination.add(transfer\_dollars, transfer\_cents);
123. // Subtract the Amount of Money from the User's Account
124. (\***this**).subtract(transfer\_dollars, transfer\_cents);
125. }
126. // Otherwise
127. **else**
128. {
129. // Display that the Entered Amount of Money Cannot be Transfered
130. cout << "Sorry, but the Amount You are Looking to Transfer Exceeds Your Current Balance." << endl;
131. }
132. }
134. // Allows Money to Transfered from Another Account into the Users
135. **void** Accounts::transfer\_from(Accounts &source, **int** transfer\_dollars, **int** transfer\_cents)
136. {
137. // Generate a WHILE Loop
138. **while** (transfer\_cents >= 100)
139. {
140. // Increase the Number of Dollars by '1'
141. transfer\_dollars += 1;
142. // Decrease the Number of Cents by '100'
143. transfer\_cents -= 100;
144. }
145. // Check if the Amount Being Tranferred is Valid
146. **if** (source.is\_valid(transfer\_dollars, transfer\_cents))
147. {
148. // Add the Amount of Money to the Other Account
149. (\***this**).add(transfer\_dollars, transfer\_cents);
150. // Subtract the Amount of Money from the User's Account
151. source.subtract(transfer\_dollars, transfer\_cents);
152. }
153. // Otherwise
154. **else**
155. {
156. // Display that the Entered Amount of Money Cannot be Transfered
157. cout << "Sorry, but the Amount You are Looking to Transfer Exceeds Your Current Balance." << endl;
158. }
159. }
161. // Check is the Amount Being Transfered is Valid
162. **bool** Accounts::is\_valid(**int** sub\_dollars, **int** sub\_cents)
163. {
164. // Set a Temporary Total
165. **double** temp\_total = sub\_dollars + (0.01 \* sub\_cents);
166. // Check if the Total is Greater than or Equal to the Temporary Total
167. **if** (total >= temp\_total)
168. {
169. // Return True
170. **return** **true**;
171. }
172. // Otherwise
173. **else**
174. {
175. // Return False
176. **return** **false**;
177. }
178. }
180. // Display the Accounts Information
181. **void** Accounts::display()
182. {
183. // Display the Account ID
184. cout << "Account ID: ";
185. **for** (**int** i = 0; i < 10; i++)
186. {
187. cout << account\_id[i];
188. }
189. cout << endl;
190. // Display the Current Balance in the Account
191. cout << "Current Balance: " << "$" << total << endl;
192. }
194. // Add to the Current Balance of the Account
195. **void** Accounts::add(**int** add\_dollars, **int** add\_cents)
196. {
197. // Set the New Amount of Dollars
198. dollars += add\_dollars;
199. // Set the New Amount of Cents
200. cents += add\_cents;
201. // Generate a WHILE Loop
202. **while** (cents >= 100)
203. {
204. // Increase the Number of Dollars by '1'
205. dollars += 1;
206. // Decrease the Number of Cents by '100'
207. cents -= 100;
208. }
209. // Set the New Total
210. total = dollars + (0.01 \* cents);
211. }
213. // Remove Money from the Current Balance
214. **void** Accounts::subtract(**int** sub\_dollars, **int** sub\_cents)
215. {
216. // Generate a WHILE Loop
217. **while** (sub\_cents >= 100)
218. {
219. // Increase the Number of Dollars by '1'
220. dollars += 1;
221. // Decrease the Number of Cents by '100'
222. cents -= 100;
223. }
224. // Set the New Dollars
225. dollars -= sub\_dollars;
226. // Set the New Cents
227. cents -= sub\_cents;
228. // Check if the Amount of Cents is Less than '0'
229. **if** (cents < 0)
230. {
231. // Decrease the Number of Dollars by '1'
232. dollars -= 1;
233. // Increase the Number of Cents by '100'
234. cents += 100;
235. }
236. // Set the New Total
237. total = dollars + (0.01 \* cents);
238. }

Code 4: testAccounts.cpp 🡪 testAccounts.cpp

1. //######################################################################
2. // Program Header: testAccounts.cpp
3. // Author: Thomas Bischoff
4. // Date Created: 4/11/2018
5. //######################################################################
7. #include "Accounts.h"
9. **int** main()
10. {
11. // Set the Time
12. srand(time(0));
13. // Display the Intro to this Section
14. cout << "Account Initialization:" << endl;
15. // Initialize the Number of Dollars and Cents
16. **int** dollar\_count, cent\_count;
17. // Introduce Account One
18. cout << "Account One -" << endl;
19. // Ask the User to Enter in the Amount of Dollars into the First Account
20. cout << "Please Enter the Number of Dollars in Account 1: ";
21. // Take the User's Input and Have it Held by Dollar Count
22. cin >> dollar\_count;
23. // Ask the User to Enter in the Amount of Cents into the First Account
24. cout << "Please Enter the Number of Cents in Account 2: ";
25. // Take the User's Input and Have it Held by Cent Count
26. cin >> cent\_count;
27. // Initilaize the First Account
28. Accounts account\_one(dollar\_count, cent\_count);
29. // Display the Information for the First Account
30. account\_one.display();
31. // Introduce Account Two
32. cout << "Account Two -" << endl;
33. // Ask the User to Enter in the Amount of Dollars into the Second Account
34. cout << "Please Enter the Number of Dollars in Account 2: ";
35. // Take the User's Input and Have it Held by Dollar Count
36. cin >> dollar\_count;
37. // Ask the User to Enter in the Amount of Cents into the Second Account
38. cout << "Please Enter the Number of Cents in Account 2: ";
39. // Take the User's Input and Have it Held by Cent Count
40. cin >> cent\_count;
41. // Initilaize the Second Account
42. Accounts account\_two(dollar\_count, cent\_count);
43. // Display the Information for the Second Account
44. account\_two.display();
45. // Mark off the Next Section
46. cout << "-------------------------------------------------" << endl;
47. // Display the Intro to this Section
48. cout << "Withdrawals and Deposits:" << endl;
49. // Display that Money Will be Removed from Account One
50. cout << "Removing $50.00 from Account One -" << endl;
51. // Withdraw $50.00 from Account One
52. account\_one.withdraw(50,0);
53. // Display the Updated Account
54. account\_one.display();
55. // Display that that Money Will be Added into Account Two
56. cout << "Adding $25.75 to Account Two -" << endl;
57. // Deposit $25.75 to Account Two
58. account\_two.deposit(25,75);
59. // Display the Updated Account
60. account\_two.display();
61. // Mark off the Next Section
62. cout << "-------------------------------------------------" << endl;
63. // Display the Intro to this Section
64. cout << "Transferring Money:" << endl;
65. // Display that $150.38 will be Transferred from Account One to Account Two
66. cout << "Transferring $150.38 from Account One to Account Two -" << endl;
67. // Transfer $150.38 from Account One to Account Two
68. account\_one.transfer\_into(account\_two,150,38);
69. // Display Account One
70. cout << "Account One -" << endl;
71. account\_one.display();
72. // Display Account Two
73. cout << "Account Two -" << endl;
74. account\_two.display();
75. }

Sample Output 4: testAccounts.cpp

