Damian Barrous

Professor Kaur

Data Structures

December 18, 2017

Project Report

Original Project Name:

Bookie Program- It allows a bookie to keep track of how much a customer owes them or they owe a customer by letting them adjust the amount balance of the customer. It will also allow them to add or remove customers.

What I Added to it after the presentation: I was not able to learn how to implement GUI in this current application, forfeiting the bonus points. But I did improve the overall feel of the program by making it more aesthetically pleasing and easier to understand in the command prompt. I did this by adding a little bit more spacing as well as using special characters to create boxes around certain text. I did add exception handling as well, whenever a user enters a string instead of an integer value it repeats the request of an integer as well as warn them they entered an invalid value. I tried to clean of the code as well by creating methods of repeated code such as the scoreboard or the menu. Overall the program works as intended above using arrays as the fundamental data structure.

Big(O) Notation of every method:

BettingGame:

Main – O(n^2) – While loop with nested loops used throughout

errorNoPlayers- O(1)- Error Message

errorInvalidOption- O(1) – Error Message

errorSameIndex- O(1) – Error Message

errorWrongIndex – O(1)- Error Message

errorMorePlayers- O(1)- Error Message

scoreBoard- O(n) – While loop

menuStart- O(n) – While loop

Player:

Player- O(1) – Constructor

Player- O(1) – Overloaded Constructor

getName- O(1) – Access Method

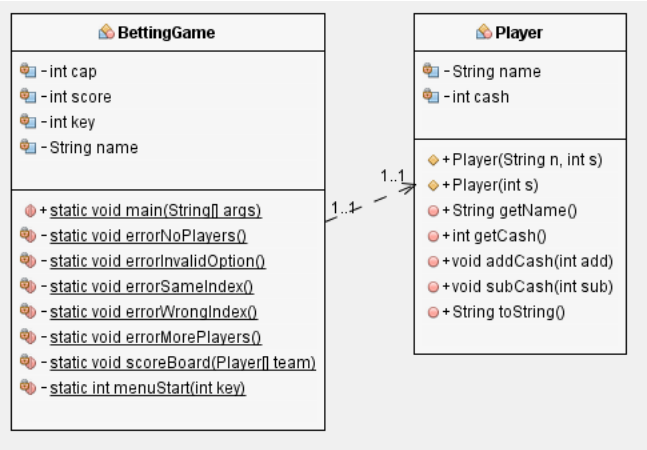
getCash- O(1)- Access Method

addCash- O(1) – Add Cash

subCash- O(1) – Subtract Cash

toString – O(1) – toString Method

UML Diagram:



Source Code:

Player Class:

```
package BettingProgram;


/**
 *
 * @author Damian's PC
 */
public class Player {
    private String name;
    private int cash;


    public Player(String n, int s)
    {
        name = n;
        cash = s;
```

```
    }
    public Player(int s){


    }


    public String getName()
    {
        return name;
    }
    public int getCash()
    {
        return cash;
    }
    public void addCash(int add){
        cash += add;
    }
    public void subCash(int sub){
        cash-= sub;
    }
    @Override
    public String toString()
    {
        return "(" + name + ", Balance: $" + cash + ")";
    }
}
```

Driver Class:

```
/*
Written by : Damian Barrous-Dume
Student ID S0297708
 */
package BettingProgram;
```

```
import java.util.InputMismatchException;
import java.util.Scanner;

/**
 *
 * @author Damian's PC
 */
public class BettingGame {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        int cap = 1000;

        String name;
        int score;
        int count = 0;

        Player[] team = new Player[cap];

        int key = 5;
        while (key != 4) {
            key = menuStart(key);

            switch (key) {

                case 1:
                    while (true) {
                        try {
                            System.out.print("Enter the name of the player: ");
```

```
            Scanner input = new Scanner(System.in);
            name = input.nextLine();
            break;
        } catch (InputMismatchException | NumberFormatException ex) {
            System.out.println("I said a name...");
        }
    }
    while (true) {
        try {
            System.out.print("Enter the Starting cash of the player: ");
            Scanner input = new Scanner(System.in);
            score = input.nextInt();
            Player g = new Player(name, score);
            team[count] = g;
            count++;
            scoreBoard(team);

            break;
        } catch (InputMismatchException | NumberFormatException ex) {
            System.out.println("I said a cash...");
        }
    }
    break;

case 2:
    if (team[1] == null) {

        errorMorePlayers();

    } else {
        int p = 5;
        int p2 = 5;
```

```
int bet = 0;
while (true) {
   try {
      System.out.println("Enter Entrance Wager: ");
      Scanner input = new Scanner(System.in);
      bet = input.nextInt();
      break;
   } catch (InputMismatchException | NumberFormatException ex) {
      System.out.println("I said a wager...");
   }
}
while (true) {
   try {
      System.out.println("Enter First Player index(Team 1) Participating:");
      Scanner input = new Scanner(System.in);
      p = input.nextInt();
      break;
   } catch (InputMismatchException | NumberFormatException ex) {
      System.out.println("I said a index...");
   }
   if (team[p] == null) {
errorWrongIndex();
break;
 } else {
break;

}}
while (true) {
   try {
      System.out.println("Enter Second Player index(Team 2) Participating:");
      Scanner input = new Scanner(System.in);
      p2 = input.nextInt();
```

```
            break;
        } catch (InputMismatchException | NumberFormatException ex) {
            System.out.println("I said a index...");
        }
    }

    if (p == p2) {
        errorSameIndex();

    } else {
        int t = 0;
        while (true) {
            try {
                System.out.println("Which Team Would Won? Enter 1 or 2");
                System.out.print("Team: ");
                Scanner input = new Scanner(System.in);
                t = input.nextInt();
                break;
            } catch (InputMismatchException | NumberFormatException ex) {
                System.out.println("I said a index...");
            }
        }

        if (t == 1) {
            Player temp = team[p];
            Player temp1 = team[p2];
            temp.addCash(bet);
            temp1.subCash(bet);

            team[p] = temp;
            team[p2] = temp1;
        } else if (t == 2) {
```

```
                Player temp = team[p2];
                Player temp1 = team[p];

                temp.addCash(bet);
                temp1.subCash(bet);

                team[p2] = temp;
                team[p] = temp1;
            } else {
                errorInvalidOption();

            }
            System.out.println();
        }
    }
    scoreBoard(team);



    break;

case 3:
    if(team[0]  == null){
        errorNoPlayers();
        break;
    }
    else{
    int c = 1000;
    while (true) {
            try {
    System.out.print("Type in the index of the person you wish to cash out: ");
                Scanner input = new Scanner(System.in);
                c = input.nextInt();
```

```
                    break;
                } catch (InputMismatchException | NumberFormatException ex) {
                    System.out.println("I said a index...");
                }
            }

            if (team[c] == null) {
                errorWrongIndex();

            } else {
                Player temp3 = team[c];
                team[c] = null;

                if (temp3.getCash() >= 0) {
                    System.out.println("~You owe " + temp3.getName() + " $" + temp3.getCash() + "~");
                } else {
                    System.out.println("~" + temp3.getName() + " owes you " + temp3.getCash() + "~");
                }
            }
            scoreBoard(team);

            break;
            }
        default:
            key = 4;
      }
   }
}

private static void errorNoPlayers() {
   System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
   System.out.println("You have no players to cash out!");
```

```
    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
}

private static void errorInvalidOption() {
    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    System.out.println("~Entered Invalid Option, Try Again!~");
    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
}

private static void errorSameIndex() {
    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    System.out.println("~You've entered the same index silly, Try Again!~");
    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
}

private static void errorWrongIndex() {
    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    System.out.println("~You've entered a wrong index, Try Again!~");
    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
}

private static void errorMorePlayers() {
    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    System.out.println("~Add atleast one more player!~");
    System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
}

private static void scoreBoard(Player[] team) {
    System.out.println();
    System.out.println("The Current Player:");
    for (int i = 0; i < team.length; i++) {
        if (team[i] == null) {
```

```
            System.out.print("");
        } else {
            System.out.println("index " + i + ": " + team[i]);
        }
    }
    System.out.println();
}

private static int menuStart(int key) {
    while (true) {
        try {
            System.out.println("*******************************");
            System.out.println("+    Type 1 - Add Player!    +");
            System.out.println("+      Type 2 - Bet!         +");
            System.out.println("+     Type 3 - Cash Out!     +");
            System.out.println("+      Type 4 - Exit!        +");
            System.out.println("*******************************");
            System.out.print("Choice: ");
            Scanner input = new Scanner(System.in);
            key = input.nextInt();
            break;
        } catch (InputMismatchException | NumberFormatException ex) {
            System.out.println("I said a number...");
        }
    }
    return key;
}

}
```

Output:

```
The Current Player:
index 0: (Damian, Balance: $1000)
index 1: (Sadiel, Balance: $1000)

******************************
+    Type 1 - Add Player!    +
+      Type 2 - Bet!         +
+     Type 3 - Cash Out!     +
+      Type 4 - Exit!        +
******************************
Choice: 2
Enter Entrance Wager:
2000
Enter First Player index(Team 1) Participating:
0
Enter Second Player index(Team 2) Participating:
1
Which Team Would Won? Enter 1 or 2
Team: 1


The Current Player:
index 0: (Damian, Balance: $3000)
index 1: (Sadiel, Balance: $-1000)
```

```
The Current Player:
index 0: (Damian, Balance: $3000)
index 1: (Sadiel, Balance: $-1000)

******************************
+    Type 1 - Add Player!    +
+      Type 2 - Bet!         +
+     Type 3 - Cash Out!     +
+      Type 4 - Exit!        +
******************************
Choice: 3
Type in the index of the person you wish to cash out: 1
~Sadiel owes you -1000~

The Current Player:
index 0: (Damian, Balance: $3000)
```

```
The Current Player:
index 0: (Damian, Balance: $3000)

******************************
+    Type 1 - Add Player!    +
+      Type 2 - Bet!         +
+     Type 3 - Cash Out!     +
+      Type 4 - Exit!        +
******************************
Choice: 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~Add atleast one more player!~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
The Current Player:
index 0: (Damian, Balance: $3000)

******************************
+    Type 1 - Add Player!    +
+      Type 2 - Bet!         +
+     Type 3 - Cash Out!     +
+      Type 4 - Exit!        +
******************************
Choice: 1
Enter the name of the player: Sadiel
Enter the Starting cash of the player: String
I said a cash...
Enter the Starting cash of the player:
```

```
run:
******************************
+    Type 1 - Add Player!    +
+      Type 2 - Bet!         +
+     Type 3 - Cash Out!     +
+      Type 4 - Exit!        +
******************************
Choice: lmao
I said a number...
```

```
******************************
+    Type 1 - Add Player!    +
+      Type 2 - Bet!         +
+     Type 3 - Cash Out!     +
+      Type 4 - Exit!        +
******************************
Choice: 4
BUILD SUCCESSFUL (total time: 29 seconds)
```