Mauricio Deguchi

UID: 405582562

CS 31

March 11, 2022

**Project 7**

**Bunco Game**

*Challenges:*

By far the largest obstacle throughout this project was the initial understanding of the classes and how they were intended to communicate across one another. Once I grasped how the classes were structured and understood each of the public accessors, the project came together relatively smoothly. The only major issue I ran into was how to force the player rolls and using that in the Bunco class. The forcing of a roll was difficult for me to understand initially especially since, when I ran the game manually, it seemed to function just fine. Aside from that, the only other obstacle was determining when a game was over. I assumed that two players could not have simultaneously won a round to conclude that the total number of wins in a finished game must add up to 6. From there, I simply checked if the board had a total of 6 wins to check if the game was over.

*Test Data:*

To test the player class, I used the following asserts:

Player p;

assert(p.getScore() == 0);

p.setRound(1);

for (int i = 2; i <= 6; i++)

{

assert(p.roll(i) == i);

assert(p.getScore() == 0);

}

assert(p.roll(1) == 1);

assert(p.getScore() == 1);

assert(p.roll(1) == 1);

assert(p.getScore() == 2);

assert(p.roll(1) == 1);

assert(p.getScore() == 3);

p.setRound(6);

assert(p.getScore() == 0);

assert(p.roll(6) == 6);

assert(p.getScore() == 1);

The initial for loop would ensure that the player’s score doesn’t go up when the die roll doesn’t match the round. Also, it ensured that our forced die rolls produced the correct outcome. The latter asserts ensured that the score counter kept track of the player’s wins properly and reset the player’s score when the round changed.

To test the Board class, I used the following asserts:

Board game;

game.setCurrentRound(1);

assert(game.countUpComputerRoundWins() == 0);

assert(game.countUpHumanRoundWins() == 0);

game.markComputerAsWinner();

assert(game.countUpComputerRoundWins() == 1);

assert(game.countUpHumanRoundWins() == 0);

game.setCurrentRound(2);

game.markHumanAsWinner();

assert(game.countUpComputerRoundWins() == 1);

assert(game.countUpHumanRoundWins() == 1);

These asserts ensured that the win counter and setting a winner in the Board class was functioning properly. Once thing to note was that both players could simultaneously be marked as a winner for a row.

Lastly, for the Bunco class I used the following tests:

Die d1;

d1.setValue(1);

Die d2;

d2.setValue(2);

Die d3;

d3.setValue(3);

Die d4;

d4.setValue(4);

Die d5;

d5.setValue(5);

Die d6;

d6.setValue(6);

Bunco b;

assert(b.determineRoundOutcome() == Bunco::NOTDECIDED);

b.computerPlay(d5);

b.humanPlay(d5);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::NOTDECIDED);

b.computerPlay(d1);

b.humanPlay(d1);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::NOTDECIDED);

b.computerPlay(d1);

b.humanPlay(d2);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::COMPUTERWON);

b.nextRound();

b.computerPlay(d2);

b.humanPlay(d2);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::NOTDECIDED);

b.computerPlay(d2);

b.humanPlay(d2);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::NOTDECIDED);

b.computerPlay(d2);

b.humanPlay(d2);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::NOTDECIDED);

b.computerPlay(d1);

b.humanPlay(d2);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::HUMANWON);

b.nextRound();

assert(b.determineGameOutcome() == Bunco::GAMENOTOVER);

b.computerPlay(d3);

b.humanPlay(d2);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::COMPUTERWON);

b.nextRound();

b.computerPlay(d3);

b.humanPlay(d4);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::HUMANWON);

b.nextRound();

b.computerPlay(d5);

b.humanPlay(d2);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::COMPUTERWON);

b.nextRound();

b.computerPlay(d3);

b.humanPlay(d6);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::HUMANWON);

b.nextRound();

assert(b.determineGameOutcome() == Bunco::TIEDGAME);The first two blocks of code were provided and help ensure that the die rolls are registered, and we can determine the winner of a single round depending on the rolls. The next two blocks help test to ensure we determine the correct winner of the game as well as ensuring that a winner is not arbitrarily predetermined before ending the game.