Duncan Di Mauro

SID: 805163177

March 12, 2021

Report 7

**Obstacles:**

In this project, the main obstacle was understanding how each header file and source file worked together and understanding what the purpose of each class was and how it was used/applied/functioned within the other classes/objects and within the main.cpp file. The main obstacle for me in this project, like most of the previous projects, was getting started. For example, I didn’t understand that the parameters mDie1 and mDie2 had the object type Die and that I could simply use mDie1.roll() in order to change the value. Once I understood that most parameters in this project had object types that were created in other .h and .cpp files within the same project, everything became much easier and I passed all the tests in Codeboard after my first try.

**List of Test Data:**

Test 1: Tests to see if all the functions for a tied-game work properly. Additionally, tests to see that a game is not over in the beginning. Also checks that the .getRoundsWon function is working accurately.

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 );

BeatThat game;

// 5 forced rounds of play...

//tests that a game is not over at the beginning

assert( game.determineGameOutcome( ) == cs31::BeatThat::GAMENOTOVER );

game.humanPlay( d1, d2 );

game.computerPlay( d2, d1 );

game.endTurn( );

assert( game.determineGameOutcome( ) == cs31::BeatThat::GAMENOTOVER );

game.humanPlay( d4, d4 );

game.computerPlay( d6, d1 );

game.endTurn( );

assert( game.getComputer( ).getRoundsWon( ) == 1 );

game.humanPlay( d6, d5 );

game.computerPlay( d6, d2 );

game.endTurn( );

assert( game.getHuman( ).getRoundsWon( ) == 1 );

game.humanPlay( d3, d3 );

game.computerPlay( d3, d3 );

game.endTurn( );

game.humanPlay( d5, d4 );

game.computerPlay( d4, d5 );

game.endTurn( );

assert( game.isGameOver() == true );

assert( game.getHuman( ).getRoundsWon( ) == 1 );

assert( game.getComputer( ).getRoundsWon( ) == 1 );

assert( game.determineGameOutcome( ) == cs31::BeatThat::TIEDGAME );

Test 2: Tests to see if all the functions for a computer-won-game work properly.

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 );

BeatThat game;

game.humanPlay( d1, d2 );

game.computerPlay( d3, d1 );

game.endTurn( );

game.humanPlay( d4, d4 );

game.computerPlay( d1, d6 );

game.endTurn( );

game.humanPlay( d6, d5 );

game.computerPlay( d6, d6 );

game.endTurn( );

game.humanPlay( d3, d3 );

game.computerPlay( d3, d3 );

game.endTurn( );

game.humanPlay( d5, d4 );

game.computerPlay( d4, d5 );

game.endTurn( );

assert( game.isGameOver() == true );

assert( game.determineGameOutcome( ) == cs31::BeatThat::COMPUTERWON );

Test 3: Tests to see if all the functions for a human-won-game work properly.

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 );

BeatThat game;

game.humanPlay( d4, d2 );

game.computerPlay( d3, d1 );

game.endTurn( );

game.humanPlay( d4, d4 );

game.computerPlay( d6, d1 );

game.endTurn( );

game.humanPlay( d6, d5 );

game.computerPlay( d6, d4 );

game.endTurn( );

game.humanPlay( d4, d3 );

game.computerPlay( d3, d3 );

game.endTurn( );

game.humanPlay( d5, d4 );

game.computerPlay( d4, d3 );

game.endTurn( );

assert( game.isGameOver() == true );

assert( game.determineGameOutcome( ) == cs31::BeatThat::HUMANWON );

Test 4: Tests to see if Board is working properly (see the different variations in the comments):

Board b;  
assert( b.getHumanRoundsWon( ) == 0 );  
assert( b.getComputerRoundsWon( ) == 0 );  
assert( b.getTurnsLeft( ) == 0 );  
assert( ! b.didHumanWin( ) );  
assert( ! b.didComputerWin( ) );  
assert( ! b.isGameOver() );  
  
b.setHumanRoundsWon( 1 );

// b.setHumanRoundsWon( 2 ); Variation 2

// b.setHumanRoundsWon( 3 ); Variation 3  
b.setComputerRoundsWon( 2 );  
b.setTurnsLeft( 3 );  
assert( b.getHumanRoundsWon( ) == 1 );

// assert( b.getHumanRoundsWon( ) == 2 ); Variation 2

// assert( b.getHumanRoundsWon( ) == 3 ); Variation 3  
assert( b.getComputerRoundsWon( ) == 2 );  
assert( b.getTurnsLeft( ) == 3 );

b.markComputerAsWinner( );  
// b.markTied( ); Variation 2  
// b.markHumanAsWinner( ); Variation 3  
assert( ! b.didHumanWin( ) ); Included in Variation 2

// assert( b.didHumanWin( ) ); Variation 3  
assert( b.didComputerWin( ) );

// assert( ! b.didComputerWin( ) ); Variation 2 & 3  
assert( b.isGameOver( ) );

Test 5: Tests to see if Player is working properly:

Player p;  
assert( p.getRoundsWon( ) == 0 );   // nothing won yet...  
p.wonARound( );       
assert( p.getRoundsWon( ) == 1 );   // won 1!  
p.roll( );                          // a random toss  
Die d1; d1.setValue( 1 );  
Die d6; d6.setValue( 6 );     
p.roll( d1, d6 );                   // cheating...  
assert( p.largestDie( ).getValue( ) == 6 );  
assert( p.smallestDie( ).getValue( ) == 1 );