Project 7

The main obstacle I overcame was trying to understand how the classes related, and how to implement the methods according to the logic Howard provided. After pouring over the spec and reading the diagram, I started to get an understanding of what I had to do. I figured out that the member variables of the classes linked them up to one another, or to put it in “teacher-speak,” made aggregation possible. Once I realized this, I began to fully understand the project.

Though I knew what was being asked of me, sometimes I did not know how to transfer my intention into code. Most of these questions were answered by using dot operators, and after figuring out how the methods of each class related, I implemented them accordingly. For example, when I would roll the correct amount for the corresponding round, the ‘X’ would not display and move on to the next round. I figured out that I had not yet use any of the mark methods in my Bunco class yet. Once I did, the X’s started appearing, which brought on the next obstacle. The game would not move on until both the human and computer player rolled the amount corresponding to the amount. I figured this had to do with the round outcome, and after going through the debugger, I found that I foolishly forgot to consider if the amounts were equal. Once I coded an if statement that considered this case, the program ran smoothly.

I developed the project incrementally, first testing the Die class, then Player, then the final Bunco class. *Here are the test cases*…

   Die d;

    for (int i = 1; i <= 100; i++)

    {

        d.roll();

        int value = d.getValue();

        assert( value >=1 && value <= 6 );

    }

    ////////////////////////

    //TESTING PLAYER CLASS//

    ////////////////////////

    Player p;

    assert( p.getScore() == 0 );    //test code from spec

    p.setRound( 1 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 0 );

    assert( p.roll( 5 ) == 5 );

    assert( p.getScore() == 0 );

    assert( p.roll( 1 ) == 1);

    assert( p.getScore() == 1 );

    p.setRound( 1 );                //test if round 1 calculates score correctly

    assert( p.getScore() == 0 );

    assert( p.roll( 2 ) == 2 );

    assert( p.getScore() == 0 );

    assert( p.roll( 3 ) == 3 );

    assert( p.getScore() == 0 );

    assert( p.roll( 4 ) == 4 );

    assert( p.getScore() == 0 );

    assert( p.roll( 5 ) == 5 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 0 );

    assert( p.roll( 1 ) == 1);

    assert( p.getScore() == 1 );

    p.setRound( 2 );                //test if round 2 calculates score correctly

    assert( p.getScore() == 0 );

    assert( p.roll( 1 ) == 1 );

    assert( p.getScore() == 0 );

    assert( p.roll( 3 ) == 3 );

    assert( p.getScore() == 0 );

    assert( p.roll( 4 ) == 4 );

    assert( p.getScore() == 0 );

    assert( p.roll( 5 ) == 5 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 0 );

    assert( p.roll( 2 ) == 2);

    assert( p.getScore() == 1 );

    p.setRound( 3 );                //test if round 3 calculates score correctly

    assert( p.getScore() == 0 );

    assert( p.roll( 1 ) == 1 );

    assert( p.getScore() == 0 );

    assert( p.roll( 2 ) == 2 );

    assert( p.getScore() == 0 );

    assert( p.roll( 4 ) == 4 );

    assert( p.getScore() == 0 );

    assert( p.roll( 5 ) == 5 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 0 );

    assert( p.roll( 3 ) == 3);

    assert( p.getScore() == 1 );

    p.setRound( 4 );                //test if round 4 calculates score correctly

    assert( p.getScore() == 0 );

    assert( p.roll( 1 ) == 1 );

    assert( p.getScore() == 0 );

    assert( p.roll( 2 ) == 2 );

    assert( p.getScore() == 0 );

    assert( p.roll( 3 ) == 3 );

    assert( p.getScore() == 0 );

    assert( p.roll( 5 ) == 5 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 0 );

    assert( p.roll( 4 ) == 4);

    assert( p.getScore() == 1 );

    p.setRound( 5 );                //test if round 5 calculates score correctly

    assert( p.getScore() == 0 );

    assert( p.roll( 1 ) == 1 );

    assert( p.getScore() == 0 );

    assert( p.roll( 2 ) == 2 );

    assert( p.getScore() == 0 );

    assert( p.roll( 3 ) == 3 );

    assert( p.getScore() == 0 );

    assert( p.roll( 4 ) == 4 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 0 );

    assert( p.roll( 5 ) == 5);

    assert( p.getScore() == 1 );

    p.setRound( 6 );                ////test if round 6 calculates score correctly

    assert( p.getScore() == 0 );

    assert( p.roll( 1 ) == 1 );

    assert( p.getScore() == 0 );

    assert( p.roll( 2 ) == 2 );

    assert( p.getScore() == 0 );

    assert( p.roll( 3 ) == 3 );

    assert( p.getScore() == 0 );

    assert( p.roll( 4 ) == 4 );

    assert( p.getScore() == 0 );

    assert( p.roll( 5 ) == 5 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6);

    assert( p.getScore() == 1 );

    p.setRound( 6 );                //test code from spec

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 1 );

    p.setRound( 6 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 1 );

    assert( p.roll( 6 ) == 6 ); //a 2nd time

    assert( p.roll( 5 ) == 5 );

    assert( p.getScore() == 2); // can exceed the value 1 and it will in the Bunco game if both the Human and Computer player continually roll the current round value at the same time over and over

    cerr << "die and player class works!" << endl;

//

//

//    ///////////////////////

//    //TESTING BUNCO CLASS//

//    ///////////////////////

    Bunco b;

    b.setRound( 1 );                                            //Round 1

    assert( b.determineRoundOutcome() == Bunco::NOTDECIDED );   //Nothing has been rolled, undecided.

    b.computerPlay( 2 );                                        //General testing:

    b.humanPlay( 2 );

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

    b.computerPlay( 2 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );

    b.humanPlay( 6 );

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

    b.computerPlay( 6 );

    b.humanPlay( 6 );

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

    b.computerPlay( 1 ); //If both human and computer roll correct amount, it does not count for either.

    b.humanPlay( 1 );

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

    b.computerPlay( 1 );                                    //Computer rolls 1 on round 1, computer wins.

    b.humanPlay( 2 );

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

    b.setRound( 2 );                                            //Round 2

    assert( b.determineRoundOutcome() == Bunco::NOTDECIDED );   //Nothing has been rolled, undecided.

    b.computerPlay( 1 );                                        //General testing:

    b.humanPlay( 1 );

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

    b.computerPlay( 1 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );

    b.humanPlay( 6 );

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

    b.computerPlay( 6 );

    b.humanPlay( 6 );

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

    b.computerPlay( 2 ); //If both human and computer roll correct amount, it does not count for either.

    b.humanPlay( 2 );

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

    b.computerPlay( 1 );                        //Human rolls 2 on round 2, human wins.

    b.humanPlay( 2 );

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

    b.setRound( 3 );                                            //Round 3

    assert( b.determineRoundOutcome() == Bunco::NOTDECIDED );   //Nothing has been rolled, undecided.

    b.computerPlay( 1 );                                        //General testing:

    b.humanPlay( 1 );

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

    b.computerPlay( 1 );

    b.humanPlay( 2 );

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

    b.computerPlay( 2 );

    b.humanPlay( 2 );

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

    b.computerPlay( 2 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );

    b.humanPlay( 6 );

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

    b.computerPlay( 6 );

    b.humanPlay( 6 );

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

    b.computerPlay( 3 ); //If both human and computer roll correct amount, it does not count for either.

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );                        //Computer rolls 3 on round 3, computer wins.

    b.humanPlay( 4 );

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

    //round 4

    b.setRound( 4 );                                            //Round 4

    assert( b.determineRoundOutcome() == Bunco::NOTDECIDED );   //Nothing has been rolled, undecided.

    b.computerPlay( 1 );                                        //General testing:

    b.humanPlay( 1 );

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

    b.computerPlay( 1 );

    b.humanPlay( 2 );

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

    b.computerPlay( 2 );

    b.humanPlay( 2 );

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

    b.computerPlay( 2 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );

    b.humanPlay( 6 );

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

    b.computerPlay( 6 );

    b.humanPlay( 6 );

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

    b.computerPlay( 4 ); //If both human and computer roll correct amount, it does not count for either.

    b.humanPlay( 4 );

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

    b.computerPlay( 3 );                        //Human rolls 4 on round 4, human wins.

    b.humanPlay( 4 );

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

    //round 5

    b.setRound( 5 );                                            //Round 5

    assert( b.determineRoundOutcome() == Bunco::NOTDECIDED );   //Nothing has been rolled, undecided.

    b.computerPlay( 1 );                                        //General testing:

    b.humanPlay( 1 );

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

    b.computerPlay( 1 );

    b.humanPlay( 2 );

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

    b.computerPlay( 2 );

    b.humanPlay( 2 );

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

    b.computerPlay( 2 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 6 );

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

    b.computerPlay( 6 );

    b.humanPlay( 6 );

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

    b.computerPlay( 5 ); //If both human and computer roll correct amount, it does not count for either.

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );                        //Computer rolls 5 on round 5, computer wins.

    b.humanPlay( 6 );

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

    //round 6

    b.setRound( 6 );                                            //Round 6

    assert( b.determineRoundOutcome() == Bunco::NOTDECIDED );   //Nothing has been rolled, undecided.

    b.computerPlay( 1 );                                        //General testing:

    b.humanPlay( 1 );

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

    b.computerPlay( 1 );

    b.humanPlay( 2 );

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

    b.computerPlay( 2 );

    b.humanPlay( 2 );

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

    b.computerPlay( 2 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 3 );

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

    b.computerPlay( 3 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 4 );

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

    b.computerPlay( 4 );

    b.humanPlay( 5 );

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

    b.computerPlay( 5 );

    b.humanPlay( 5 );

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

    b.computerPlay( 6 ); //If both human and computer roll correct amount, it does not count for either.

    b.humanPlay( 6 );

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

    b.computerPlay( 5 );                        //Human rolls 6 on round 6, human wins.

    b.humanPlay( 6 );

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

    b.setRound(7);

    assert(b.determineGameOutcome()==Bunco::TIEDGAME);  //Computer score: 3, Human score: 3

    Bunco c;

    c.setRound(1);

    c.computerPlay(1);

    c.humanPlay(6);

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

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

    c.setRound(2);

    c.computerPlay(2);

    c.humanPlay(6);

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

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

    c.setRound(3);

    c.computerPlay(3);

    c.humanPlay(3);

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

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

    c.setRound(3);

    c.computerPlay(3);

    c.humanPlay(6);

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

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

    c.setRound(4);

    c.computerPlay(4);

    c.humanPlay(6);

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

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

    c.setRound(5);

    c.computerPlay(5);

    c.humanPlay(6);

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

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

    c.setRound(6);

    c.computerPlay(6);

    c.humanPlay(1);

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

    //assert(c.determineGameOutcome() == Bunco::GAMENOTOVER);

    assert(c.determineGameOutcome() == Bunco::COMPUTERWONGAME);

    assert(c.determineGameOutcome() == Bunco::COMPUTERWONGAME); //Computer score: 6, Human score: 0

    Bunco e;

    e.setRound(1);

    e.computerPlay(6);

    e.humanPlay(1);

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

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

    e.setRound(2);

    e.computerPlay(6);

    e.humanPlay(2);

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

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

    e.setRound(3);

    e.computerPlay(3);

    e.humanPlay(3);

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

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

    e.setRound(3);

    e.computerPlay(6);

    e.humanPlay(3);

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

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

    e.setRound(4);

    e.computerPlay(6);

    e.humanPlay(4);

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

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

    e.setRound(5);

    e.computerPlay(6);

    e.humanPlay(5);

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

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

    e.setRound(6);

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

    e.computerPlay(1);

    e.humanPlay(6);

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

    assert(e.determineGameOutcome()==Bunco::HUMANWONGAME);  //Computer score: 0, Human score: 6

    assert(e.determineGameOutcome() != Bunco::TIEDGAME);

    assert(e.determineGameOutcome() != Bunco::COMPUTERWONGAME);

    Bunco f;

    f.setRound(1);

    f.humanPlay(1);

    f.computerPlay(2);

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

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

    f.setRound(2);

    f.humanPlay(1);

    f.computerPlay(2);

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

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

    f.setRound(3);

    f.humanPlay(3);

    f.computerPlay(2);

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

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

    f.setRound(4);

    f.humanPlay(3);

    f.computerPlay(4);

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

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

    f.setRound(5);

    f.humanPlay(5);

    f.computerPlay(4);

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

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

    f.setRound(6);

    f.humanPlay(5);

    f.computerPlay(6);

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

    assert(f.determineGameOutcome() == Bunco::TIEDGAME);

    assert(f.determineGameOutcome()!=Bunco::GAMENOTOVER);

    cerr << "bunco works!" << endl;