CS 31

Project 7 Report

Melody Chen

#705120273

One of the obstacles I faced while doing this project was understanding the purpose of each class and their member variables. As this project revolves around main different objects and they each have their own functions, it was a bit confusing at first. But after I read the specs carefully, I understood what to do better.

Test Cases:

*int main()*

*{*

*using namespace std;*

*using namespace cs31;*

*Die d;*

*Player p;*

*Board board, b1;*

*Pig game;*

*// test code for Die*

*for (int i = 1; i <= 100; i++ )*

*{*

*d.roll();*

*int value = d.getValue();*

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

*}*

*// test code for Player*

*assert(p.getScore( ) == 0 ); //test if initial cond is 0*

*assert(p.getTotal( ) == 0 ); //test if initial cond is 0*

*assert( p.roll( 6 ) == 6 ); //test is roll is correct*

*assert( p.roll( 7 ) == 7 ); //test is roll is correct*

*assert( p.getScore() == 13 ); //test is getscore is correct*

*assert( p.roll( 5 )  == 5 ); //test is roll is correct*

*assert( p.getScore() == 18 ); //test is getscore is correct*

*assert( p.getTotal() == 0 ); //total should only be updated after endTurn*

*p.endTurn();*

*assert( p.getScore() == 0 ); //score should be reset after endturn*

*assert( p.roll( 4 ) == 4 );//test is roll is correct*

*assert( p.getScore() == 4 );//test is getscore is correct*

*assert( p.roll( 5 ) == 5 );//test is roll is correct*

*assert( p.getScore() == 9 );//test is getscore is correct*

*assert( p.roll( 1 ) == 1 );//test is roll is correct*

*assert( p.getScore() == 0 ); //test if score gets reset*

*assert( p.getTotal() == 18 ); //total should not be updated*

*// test code for Board*

*assert( board.getComputerTotal() == 0 ); //total should be zero at start*

*assert( board.getHumanTotal() == 0);//total should be zero at start*

*assert( board.getRolledValue() == 0);//rolledvalue should be zero at start*

*assert( board.getComputerScore() == 0 ); //computer score should be zero at start*

*assert( board.getHumanScore() == 0); //human score should be zero at start*

*assert( board.isHumanWinner() == false ); //human has not won yet*

*assert( board.isHumanTurn() == true ); //default human's turn*

*assert( board.isGameOver() == false ); //game is not over*

*board.setComputerTurn();*

*board.setComputerTotal(1); //see if total changes*

*board.setComputerScore(2); //see if score changes*

*board.setHumanScore(30); //should not do anything*

*assert(board.getHumanScore()==0); //human score should still be zero*

*board.setHumanTurn();*

*board.setHumanScore(3);*

*board.setHumanTotal(4);*

*board.setRolledValue(7);*

*board.setComputerScore(20); //should not do anything*

*assert( board.getComputerTotal() == 1 ); //see if total is changed*

*assert( board.getHumanTotal() == 4); //see if total is changed*

*assert( board.getRolledValue() == 7); //see if rolled value is 7*

*assert( board.getComputerScore() == 2 ); //see if computer score is updated*

*assert( board.getHumanScore() == 3); //see if human score is correct*

*assert( board.isHumanWinner() == false ); //no one has won yet*

*assert( board.isHumanTurn() == true ); //still human's turn*

*assert( board.isGameOver() == false ); //no one's score is above 30*

*b1.setComputerTurn();*

*b1.setComputerTotal(21);*

*b1.setComputerScore(24);*

*b1.setComputerScore(22);*

*b1.setHumanTurn();*

*assert( b1.getComputerTotal() == 21 ); //see if total updates correctly*

*assert( b1.getComputerScore() == 22 ); //see if score updates correctly*

*b1.setHumanScore(23);*

*b1.setHumanScore(1);*

*b1.setHumanTotal(24);*

*b1.setRolledValue(27);*

*b1.setRolledValue(250);*

*b1.setGameOver(true);*

*b1.markComputerAsWinner();*

*assert( b1.getComputerTotal() == 21 ); //see if total is correct*

*assert( b1.getHumanTotal() == 24); //see if total is correct*

*assert( b1.getRolledValue() == 250); //see if rolled value is correct*

*assert( b1.getComputerScore() == 22 ); //see if socre is correct*

*assert( b1.getHumanScore() == 1); //see is human socre is correct*

*assert( b1.isHumanWinner() == false ); //human should not be winner*

*assert( b1.isHumanTurn() == true ); //still human's turn*

*assert( b1.isGameOver() == true ); //game is over*

*b1.setComputerTurn();*

*b1.setComputerTotal( 60 );*

*b1.setComputerScore( 50 );*

*b1.setComputerScore(60);*

*b1.setRolledValue( 2 );*

*assert( b1.getComputerTotal( ) == 21 ); //no change made*

*assert( b1.getComputerScore( ) == 22 ); //no change made*

*assert( b1.getRolledValue( ) == 250 ); //no change made*

*// test code for Game*

*assert( game.isGameOver() == false );//game should not be over when it started*

*assert( game.determineGameOutcome() == Pig::GAMEOUTCOME::GAMENOTOVER ); //game should not be over*

*game.humanPlay( 5 );*

*game.humanEndTurn( );*

*game.computerPlay( 5 );*

*game.computerEndTurn( );*

*assert( game.isGameOver() == false ); //game should not be over*

*assert( game.determineGameOutcome() == Pig::GAMEOUTCOME::GAMENOTOVER ); //game should not be over*

*game.humanPlay( 5 );*

*game.humanPlay( 5 );*

*game.humanPlay( 5 );*

*game.humanEndTurn( );*

*assert( game.isGameOver() == false ); //game should still not be over*

*assert( game.determineGameOutcome() == Pig::GAMEOUTCOME::GAMENOTOVER ); //game should not be over*

*game.computerPlay( 5 );*

*game.computerPlay( 5 );*

*game.computerPlay( 5 );*

*game.computerEndTurn( );*

*assert( game.isGameOver() == false ); //game is not over yet*

*assert( game.determineGameOutcome() == Pig::GAMEOUTCOME::GAMENOTOVER ); //game not over*

*game.humanPlay( 5 );*

*game.humanPlay( 6 );*

*assert( game.determineGameOutcome() == Pig::GAMEOUTCOME::GAMENOTOVER ); //game not over becuase human has to end turn*

*game.humanEndTurn( );*

*assert( game.isGameOver() == true ); //game is over*

*assert( game.determineGameOutcome() == Pig::GAMEOUTCOME::HUMANWONGAME ); //human won game*

*// once the game is over, scores can't be changed*

*game.computerPlay( 6 );*

*game.computerPlay( 6 );*

*game.computerPlay( 6 );*

*game.computerEndTurn( );*

*game.humanPlay(10);*

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

*assert( game.determineGameOutcome() == Pig::GAMEOUTCOME::HUMANWONGAME );*

*Pig game2;*

*game2.humanPlay(30);*

*game2.computerPlay(50);*

*game2.humanEndTurn();*

*assert(game2.isGameOver()==true); //game should be over*

*game2.computerPlay(50);*

*assert( game2.determineGameOutcome() == Pig::GAMEOUTCOME::HUMANWONGAME ); //human should win because computer did not end turn*

*cout << "all tests passed!" << endl;*

*return 0;*

*}*