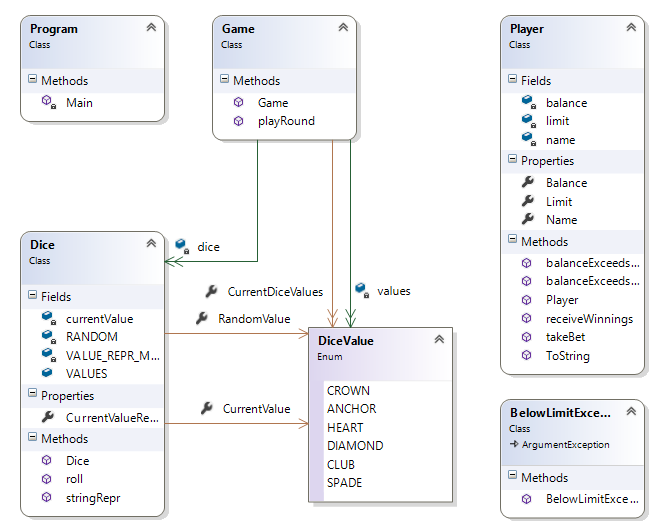
Planned Approach for Assignment 4

# Familiarize myself with the code.

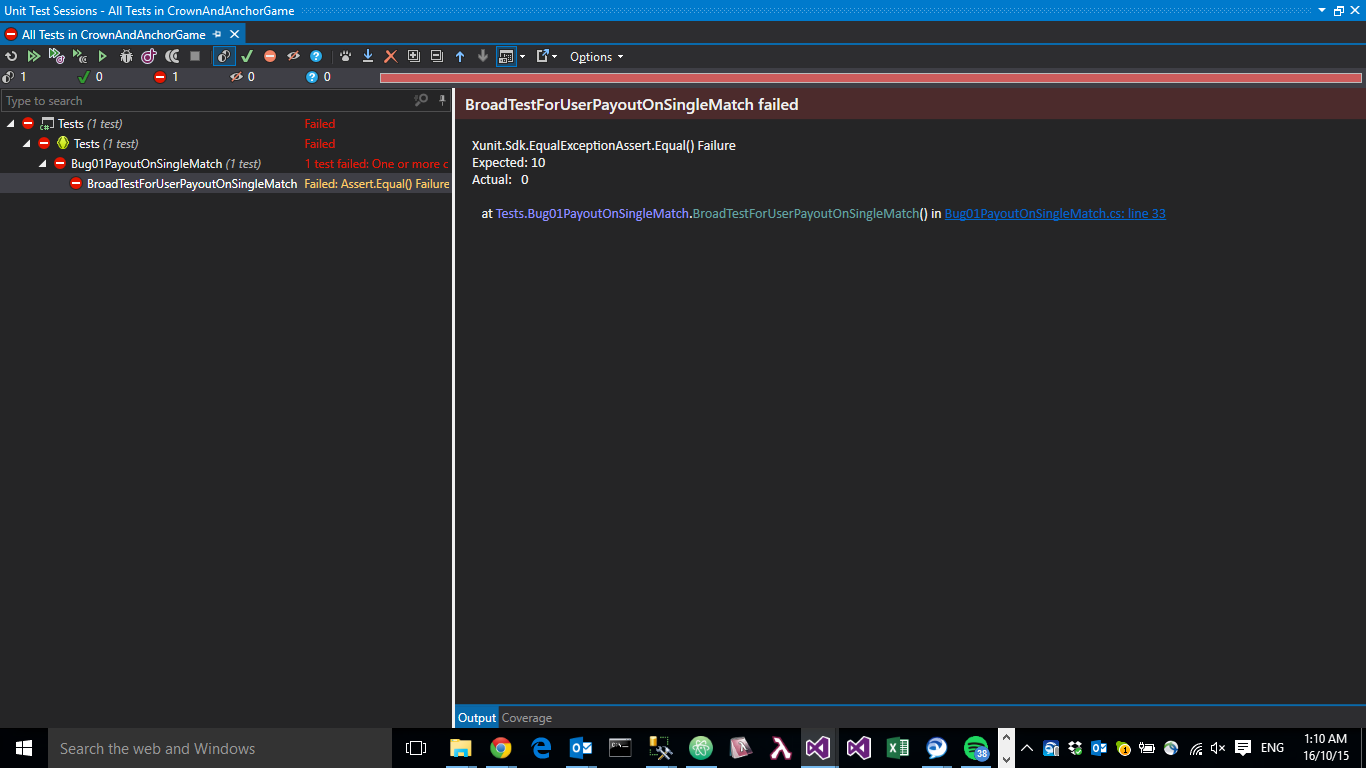
The first step is to do a quick review of the objects, and the structure of the program, see what relates to what and where. I will produce a class diagram so I can visualize what is going on.



# Produce UAT Tests for each Bug

Done

# Produce a broad Unit Test for each bug



Test has been created and is failing because the winning returned is zero and not the bet amount.

Checking in.

# See if there are any obvious additional Tests that I can introduce to cover side-effects

As well as returning correct winnings, I am checking the gambler’s winnings are incremented correctly:

Assert.Equal(bet, winnings);

Assert.Equal(funds + bet, player.Balance);

I have also added tests to make sure the player methods to deduct the bet and increment the player’s funds by the winnings have been called:

[Fact]

public void SideEffectBetIsDeductedDuringGame()

{

var die1 = Substitute.For<Dice>();

var die2 = Substitute.For<Dice>();

var die3 = Substitute.For<Dice>();

DiceValue pick = DiceValue.ANCHOR;

int bet = 10;

int winnings = 0;

int funds = 100;

var player = Substitute.For<Player>("Test", funds);

var game = new Game(die1, die2, die3);

winnings = game.playRound(player, pick, bet);

// Make sure at some point that the bet is deducted by the funds.

player.Received().takeBet(bet);

}

[Fact]

public void SideEffectFundsAreIncrementedByWinningsDuringGame()

{

var die1 = Substitute.For<Dice>();

var die2 = Substitute.For<Dice>();

var die3 = Substitute.For<Dice>();

DiceValue pick = DiceValue.ANCHOR;

int bet = 10;

int winnings = 0;

int funds = 100;

var player = Substitute.For<Player>("Test", funds);

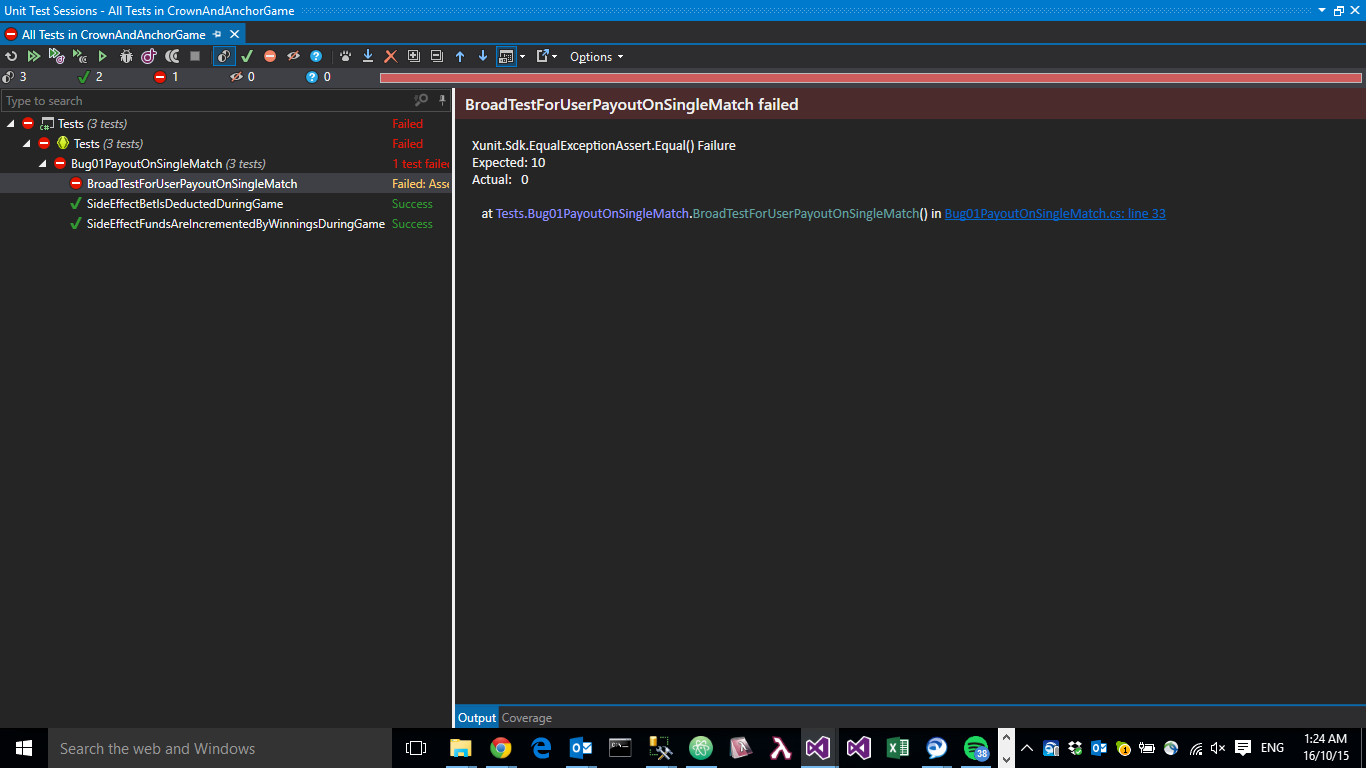
var game = new Game(die1, die2, die3);

winnings = game.playRound(player, pick, bet);

// Make sure at some point that the winnings are added to the funds during the game.

player.Received().receiveWinnings(winnings);

}



# Introduce logging to console and file around the state of object in play

# Run look at where the Unit Tests fail and trace that line of code and check object states at those times

I may wrap tests around what the expected states should be.

# Manually step through the code, the stacktrace, and the object windows to see what is being set and where the bug is occurring

# Write a Unit Test to fix the bug

# Fix the Bug

# Test for Side-Effects