2. To get enough code to scale for refactoring, the Monopoly game, which was coded in C# as a final project for PROG8140 was chosen for refactoring. At first, to get the smell of code, calculate code metrics with Visual studio, and I can get the result as describe at “2.codeMetrics.pdf”.

**1. Extract Method.**

First, LandedOn() has higher Cyclomatic Complexity than other method, which means there are a lot of condition for branch which can make hard to read code and potential to make error. Thus, Using Extract method, extract 2 methods, which are PayRent and BuyProperties. This reduce Cyclomatic Complexity from 12 to 5 and Maintainability Index also increased from 47 to 61.

Please refer to “2.PropertyCell.pdf”.

**2. Extract Constant**

During the project period, our team tried to make clean code. Thus, not so much coded as literal. After reviewing overall code, I found 2 places that use literal code, which are GameMaster.cs and IncomeTaxCell.cs. All literal constant codes are replace with constant variables.

const int MAX\_DICEROLL = 40;

.. } while (!((int.*TryParse*(*Console*.*ReadLine*(), out utilDiceRoll)) && (utilDiceRoll < MAX\_DICEROLL)));

const int MIN\_PROPERTY\_TO\_TAX = 2000;

const double TAX\_RATE = 0.1;

..

tax = (int)*Math*.*Floor*(*Math*.*Min*((double)MIN\_PROPERTY\_TO\_TAX,(double)(curPlayer.getNetWorth() \* TAX\_RATE)));

Other 2 refactoring pattern, which are “Introduce Explaining Variable” and “Replace Constructor Call with Factory Method”, were not used for refactoring because they are not proper to use for our codes.