
# A VARIATION ON MONOPOLY

Laurie Guimont

CSC 17A

Spring 2017

42636

## TABLE OF CONTENTS

## Introduction

Monopoly is a classic board game that has been played and enjoyed by many people over the years. This complex 2-8 player game is a great game to play if you have a considerable amount of time to pass. Since the game takes a while to complete in its entirety, many people have either made up their own rules or refuse to even play the game. Nonetheless, Monopoly is still an extremely popular game, which is the reason why many different thematic versions of the game have been invented since its original creation.

The game's complexity is particularly intriguing to me personally, which is why I decided to code the game. However, although I knew that coding Monopoly would not be an easy task, I did not expect it to be as hard as it proved to be.

## How the Game Works

### Object of the Game

To gain control or a "monopoly" of the entire board by becoming the richest player through buying, renting, and selling/trading property.

### Elements of the Game

The game contains many elements that contribute to the complexity of the classic board game, such as:

- A Board
- Two Dice
- Playing Pieces/Tokens
- Chance & Community Chest cards
- Title Deed Cards
- Houses and Hotels
- Trademarked Money

On the playing board are names and prices of all properties available for players to purchase. However, not all spaces on the board are properties. There are also special spaces on the board including:

- Go
- Income & Luxury Tax
- Chance
- Community Chest
- Jail

- Go to Jail
- Free Parking

Each of these spaces have their own rules and costs related to them.

**Game Play**

Before players can start the game, the banker gives all players $1500 and players choose their playing piece that will move around the board. Also, the Chance and Community Chest cards must be placed in their designated areas. Once this is completed and all players have their money and playing token, the players take turns rolling the dice to see who goes first. Whoever rolls the higher sum of the two die is the player who gets to start the game.

The winner of the roll rolls the dice again and moves their playing piece according to the total number that is displayed on the dice. If the player lands on a property, they choose to either purchase the property landed on, or to visit. (In some versions of the game, if a player chooses not to purchase the property, the property immediately goes into auction and is won by the highest bidder.) If the player decides to make the purchase, they are given a title deed card that contains detailed information about the property, including rent values, house and hotel purchase prices, and the mortgage price. Once the player makes a decision, it is then another players' turn to follow suit.

Players continue to take turns moving around the board and purchasing property. If a player lands on a square that is not property-related however, they must do what is required of them. Here are the rules for such spaces:

- If a player lands on "Chance" or "Community Chest", that player must pick up a card from either stack on the board, read the card, and follow the instructions given to them.
- If a player lands on the "Go to Jail" square, they must relocate their playing piece to the space on the board labeled "Jail". To get out of jail, players have three options:
    - Immediately pay the $50 fine to the Bank
    - Roll the dice a maximum of three times with the hope of rolling a double (or else pay the fine if doubles are not achieved by the third roll)
    - Use a "Get Out of Jail Free" card that is only obtained from acquiring it from the Chance or Community Chest cards (Players can also bargain with another player who already has a "Get Out of Jail Free" card)
- If a player passes "Go", they must collect $200
- If a player lands on a tax square, such as Income Tax or Luxury Tax, they must pay that tax to the bank
- If a player lands on "Free Parking", they can rest until it is their turn again

Houses can be purchased/built once a player owns all the properties within a certain color group. Hotels are built only on specific properties that already have four houses.

If a player is running out of money at any point in the game, they can mortgage any property that they possess. However, if another player lands on a mortgaged property, they do not owe the owner rent. Once the owner has acquired enough money once again, they can unmortgage the property for the price of the mortgage, plus 10%, and start collecting rent again.

The game concludes once all other players become bankrupt.

## My Approach to the Game

### The Race Against Time

As I mentioned earlier, coding Monopoly proved to be extremely hard and extensive. Due to this reason, I did not completely incorporate all the elements of the game nor did I code the game exactly as the board game is played. While I was programming the game, I constantly said to myself, "Man, I wish I had a couple more weeks to finish this!" As I was creating the flowchart that reflected my finished code, I kept saying "I really should have done this a different way." Unfortunately, I had a deadline to keep in mind. Making certain efficient adjustments was out of the question as the deadline approached. However, although I may have overcomplicated the coding, I am satisfied that I created a working product and implemented all the topics that were required.

### Similarities to the Board Game

For the most part, gameplay is the same. I begin the game by having the player and the opponent, which is the computer, roll to see who gets to go first. Once the winner is determined, the game begins and players take turns rolling the dice and advancing around the "board".

Additionally, I maintained all the property information, including the cost of rent pertaining to house and hotel quantity, as well as the positioning of the property as seen on the playing board of the actual game. Special properties, such as all railroads and utilities, also maintain their original characteristics.

I also coded all "special square" procedures, including Chance and Community Chest, Go to Jail, Go, and Tax. These characteristics add to the realness of my game.

### Differences from the Board Game

One major difference between the actual game and my version of the game is that my version is completely textual. I would have liked to have made the game more visual, however I did not have enough time to do so. (Also, I'm not well versed in that yet, unfortunately).

Secondly, I manage houses and hotels differently than the game. As you recall, the typical monopoly rules say that houses can be built on properties in a specific color group once all

properties of that color are possessed by a player. Then once houses are built, rent goes up on the properties that contain houses. Like the game, my version allows the player to purchase houses once they possess all properties in a specific color. In contrast, however, once houses are purchased, there is no specification as to which property owns the houses. This means that rent doesn't just go up for a couple properties owned by the player, but instead rent goes up for ALL properties owned by the player. So, if an opponent (in this instance, the computer) lands on ANY property owned by the player after houses have been purchased, they must pay rent according to the total number of houses owned. Crazy right? It is as if the board represents a gated community where all property owners within the community are forced to pay the same fees/penalties. Also, to further restrict the computer, only the user can purchase properties. The computer does not have that ability.

The final difference is that I do not allow for properties to be mortgaged and unmortgaged. All money in my game must be obtained purely through collecting rent from purchased property and from all special features of the board (passing Go, collecting money from a Community Chest card, etc). This means that the first person to have a negative balance loses the game, regardless of how many properties they own. Lose your money, lose the game!

The differences mentioned above give the user somewhat of an advantage. However, having the advantage does not mean that the user will always win. It is very easy to get caught up in buying as many properties as you can and lose track of your money, even though user progress is displayed at the end of each turn. Greed is a terrible thing!

**Reminder:** Although the computer asks the user to enter their opponent's name, this is not a two-player game. The user's opponent is the computer, but I allow the user to give the computer a name. Therefore, the computer is pre-programmed to automatically purchase all properties landed on, if available. The user has no say in the matter.

## The Logic of it All

### Flowchart & Pseudocode

Since my flowchart is extremely long, I will break it up into smaller pieces and accompany it with pseudocode here. To view my complete flowchart, please visit:

1. https://www.gliffy.com/go/html5/11933300
2. https://www.gliffy.com/go/html5/11933303
3. https://www.gliffy.com/go/html5/11933304
4. https://www.gliffy.com/go/html5/11933308
5. https://www.gliffy.com/go/html5/11933310
6. https://www.gliffy.com/go/html5/11933305

## Board Class

*Include string library*

*Protected member variables declared*

*Declare constructor and all public member functions, including getters*

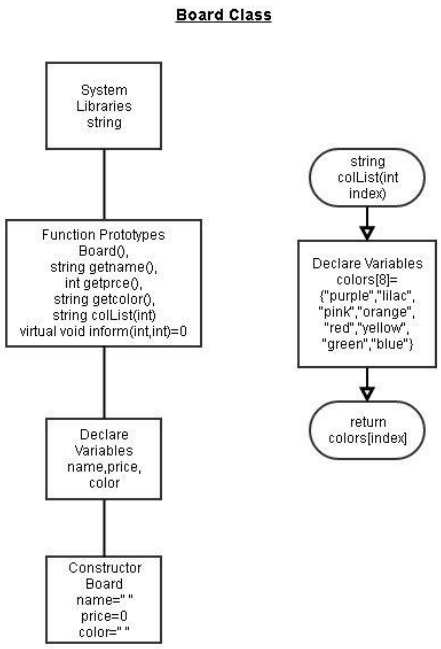
*Initialize pure virtual function called "inform"*

*Board Class is now abstract*

*Initialize class constructor*

*Define "colList" function by declaring array containing eight colors for color-groups*

*Return a string of a color of the specified index when called*




## Die Class

*Include cstdlib library*

*Private member variable value declared*

*Declare constructor and all public member functions, including getters*

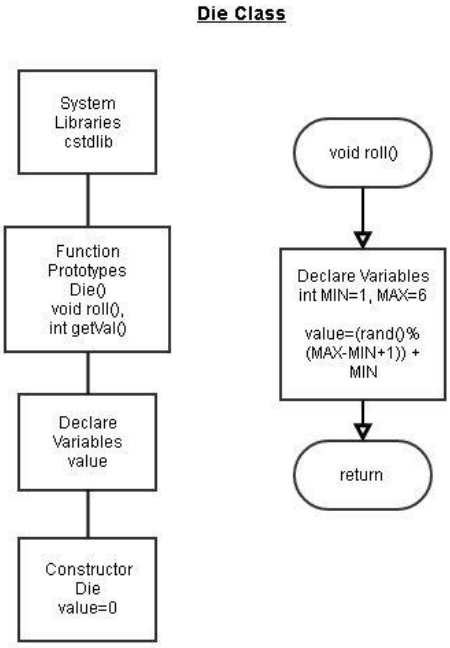
*Initialize constructor by setting value to 0*

*Define roll() function*

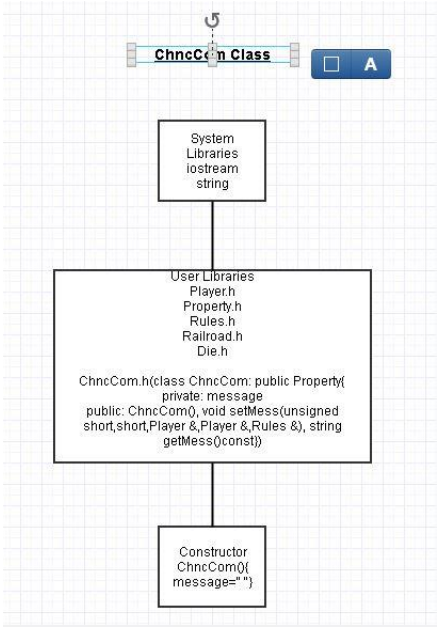
*Declare local function variables*

*Set value equal to a random number between MIN and MAX*

*Return*



**ChncCom Class**




*Include iostream and string libraries*

*Bring in user libraries of other classes to*

*implement and call public functions pertaining*

*to that class*

*Define the ChncCom Class*

*Declare private member variable message*

*Declare constructor and all public getter functions*

*Declare friend function "setMess", who is friends with*

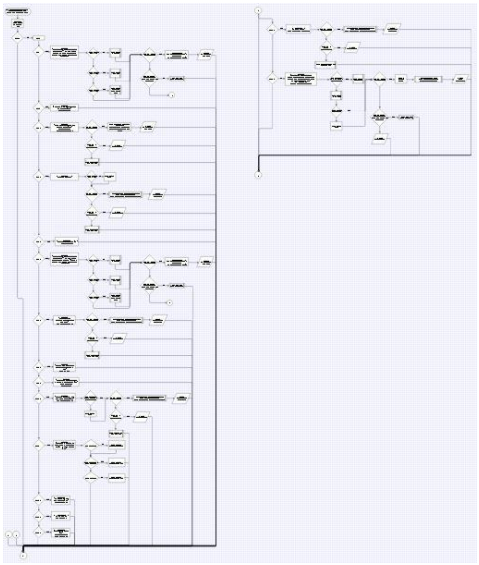
*Player and Rules classes*

*Initialize constructor by defaulting message, a string*

*Define function setMess, whose purpose is to determine*

*the Chance/Community Chest card that is*

*"picked"*



*Instantiate Property, Railroad and Die class objects*

*Begin with if/else statement to determine whether*

*Player has landed on Chance or*

*Community Chest*

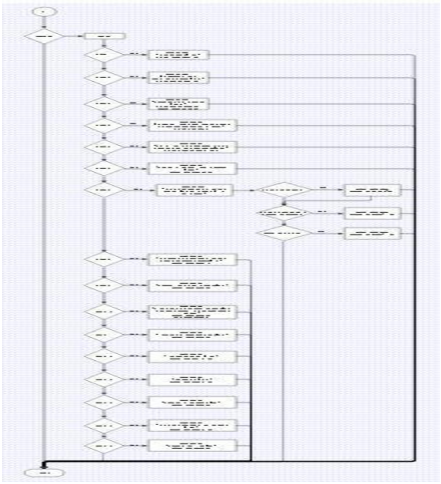
*If Chance, go through first switch statement*

*Else if Community Chest, go through second switch*

*Set message variable to index value that was read in*

*Perform any procedures required based upon the*

*message text*



*ChncCom has direct access to private member functions of*

*Player and Rules*

**Player Class**

*Include iostream, iomanip, cctype, and string libraries*

*Include Property header file to access private member functions for use*

*Define Player class*

*Declare private member variables name, nProps, proprty (a pointer), spot, outJail, nHouses, nHotels*

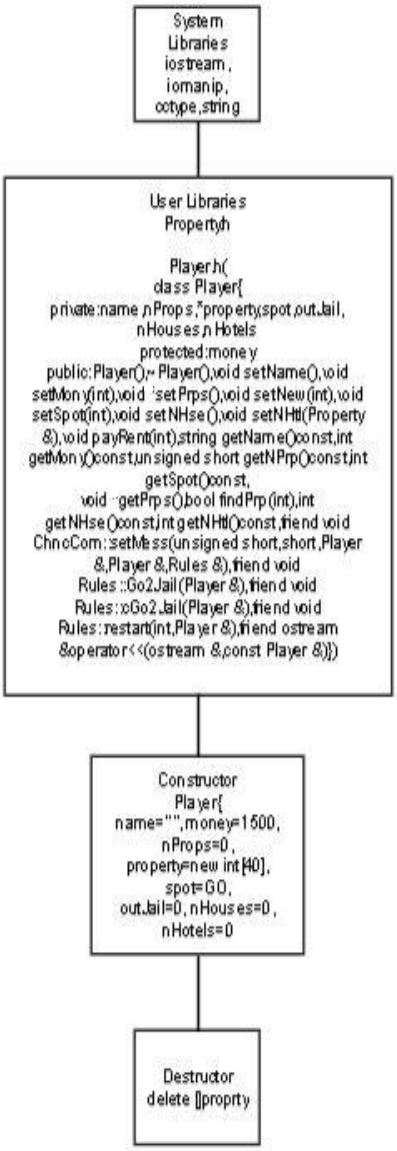
*Declare protected member variable money*

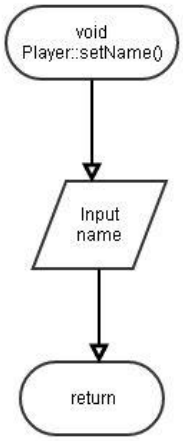
*Declare constructor, destructor, setters, getters, and friend functions that have access to private members*

*Define constructor by setting money to 1500 (representing the amount of money players are to start out with) and set all other variables to 0*

*Spot is set to GO, an enum found in Property Class*

*Define destructor the deletes pointer variable when program ends.*




*Define setName() function*

*Function takes in names from the user and sets it to the indicated player*

*Return*

*Define setPrps() function*

*If player purchases property they are currently on*

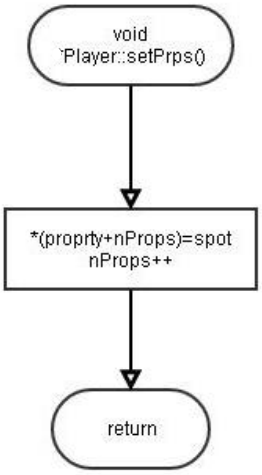
*property is filled individually*

*The index, which is nProps variable then increments*

*the pointer in preparation of the next property player*

*wants to purchase*

*Return*



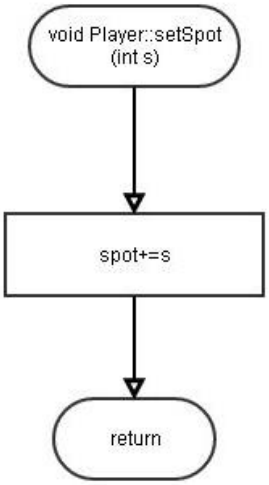

*Define setSpot function*

*Player's new spot is equal to previous spot plus*

*Whatever is rolled by the dice*

*Dice sum is taken in as an int*

*Return*




*Define setNHse function*

*Instantiate Property class object*

*Declare local variables and arrays*

*Go through array of player properties to see if*

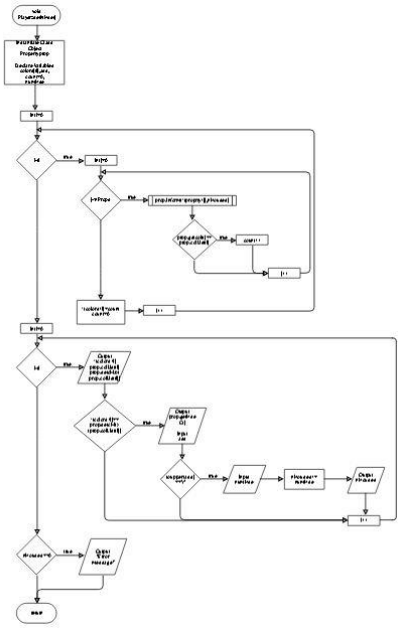
*they have purchased all properties in color-group*

*If yes, then ask how many houses they want*

*Accept number and adjust nHouses variable*

*according to input*

*Else, display error message and Return*



*Define setNHtl function, which accepts a Property Object*

*Declare local variables*

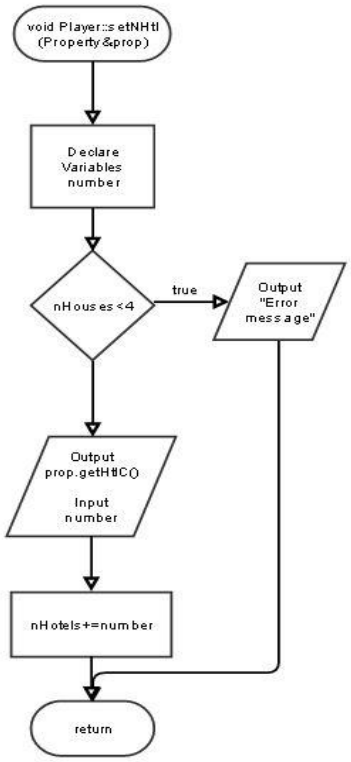
*Check if number of houses bought is less than 4*

*If so, output error message*

*Else, ask how many hotels they would like to purchase and output the cost*

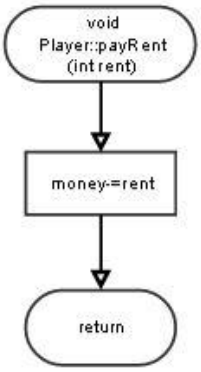
*Take in number and adjust nHotels according to input*

*Return*




*Define payRent function which determines amount of the player's money after rent has been taken out/paid*

*Return*

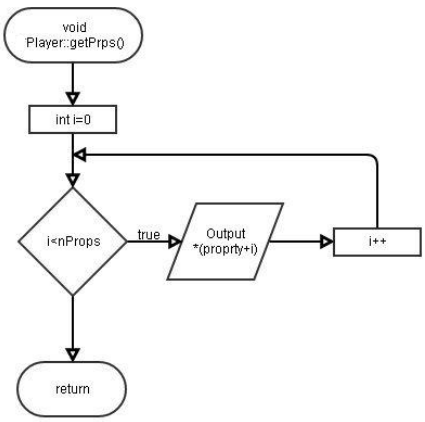



*Define getPrps() function*

*Displays player's properties one by one through a for loop*

*Loops according to how many properties are owned*

*Return*



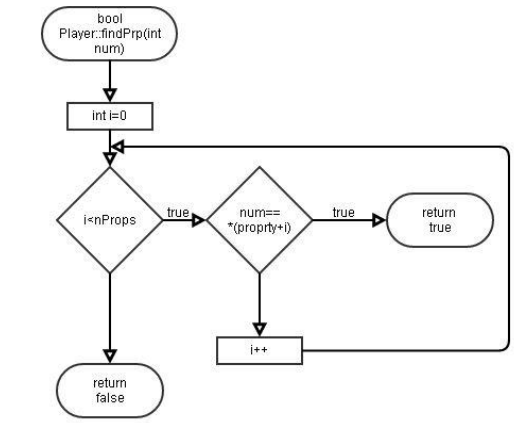
*Define findPrp function*

*Enter for loop to find a particular property owned by*

*specified player*

*If value is found return a Boolean value of true*

*Else return Boolean value of false*




*Declare operator<< function which overloads*

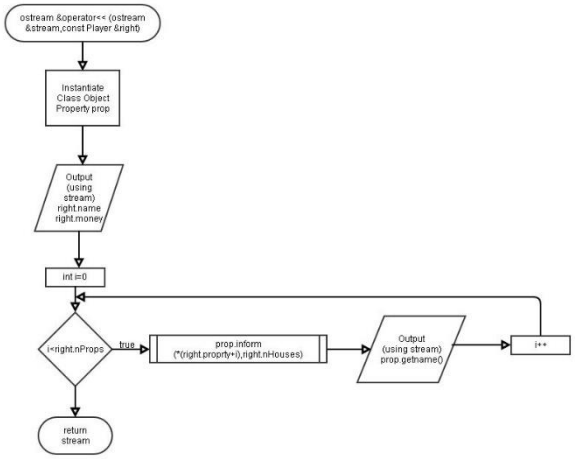
*the << operator when a Player object*

*is to the right of it*

*Output player name, money and properties at the end*

*of each turn*

*Return*




**Property Class**

*Include string Class*

*Declare enum name and initialize all names on Board*

*Property inherits from Board class (its parent class)*

*Declare private and protected member variables*

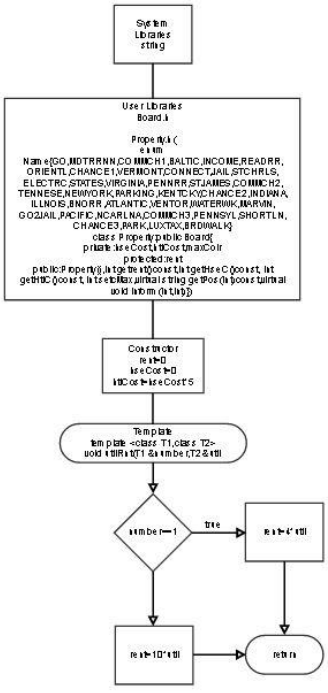
*Declare constructor, public and virtual member functions*

*Define constructor by setting all values to 0*

*Declare and define a template called utilRent*

*If number equals 1, rent is 4 times roll value*

*Else rent is 10 times roll value*



*Define setcMax function which takes in a string*

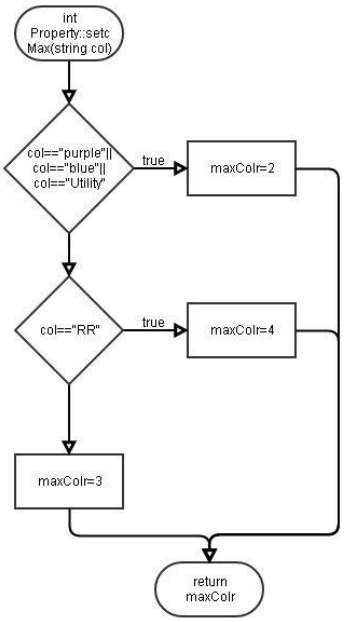
*If col is purple, blue, or labeled Utility,*

*maxColr is 2*

*Else if col is labeled RR maxColr is 4*

*Else maxColr is 3*

*Return maxColr, an integer*




*Define the virtual function inform*

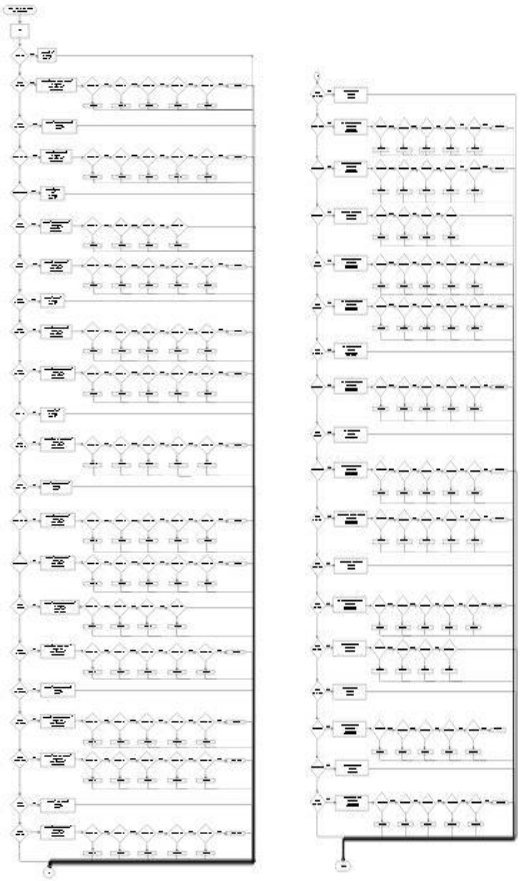
*Function takes in two integers and sets the information of each property using a switch*

*Name, price, and color are all set according to specifications*

*Name, price and color were all inherited from Board*

*Rent and house cost, private variables to the class, are all initialized as well according to specifications*

*Return from function after all is set*



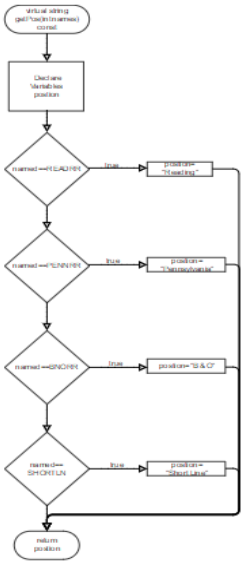
*Define getPos, a virtual function that returns a string*

*Declare local variable postion*

*If int read in is one of the railroads, output the*

*actual railroad name*

*Return postion*



## Railroad Class

*Include iostream and string library*

*Include Player and Property header files*

*Railroad is the child of Property*

*Inheritance is public*

*Declare setrent function*

*Instantiate a local Railroad object*

*Set rent according to the number of railroads*

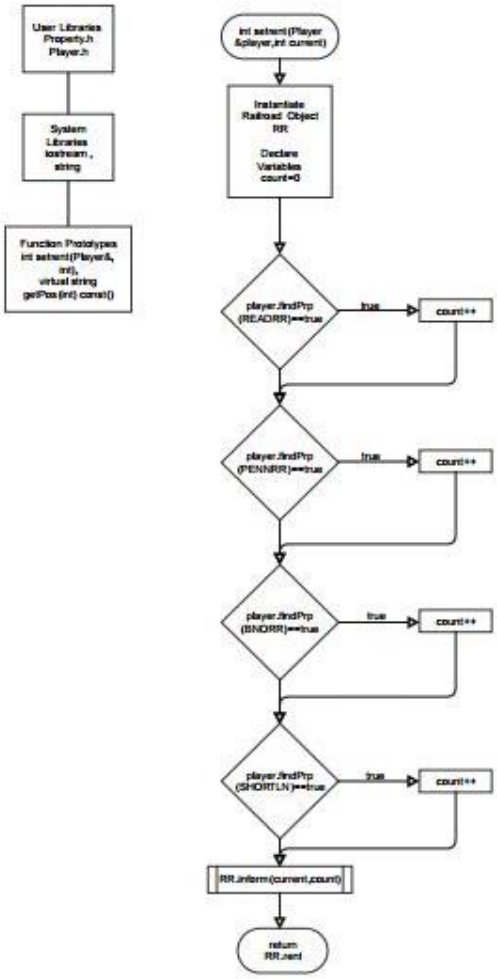
*an opposing player has to determine*

*total rent owed to current player*

*The inform function is called from Property class*

*to get the rent rate*

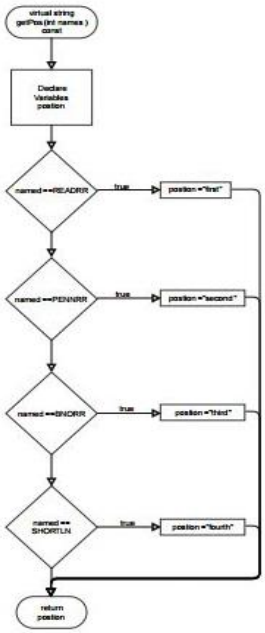
*Rent is then returned as an integer*



*Redefine virtual function getPos from Property class to demonstrate polymorphism*

*This time, postion, another local variable, is set to "first, second, third, or fourth" instead of the actual name*

*Whatever postion is set to, return postion, a string variable*




**Rules Class**

*Include iostream and cctype libraries*

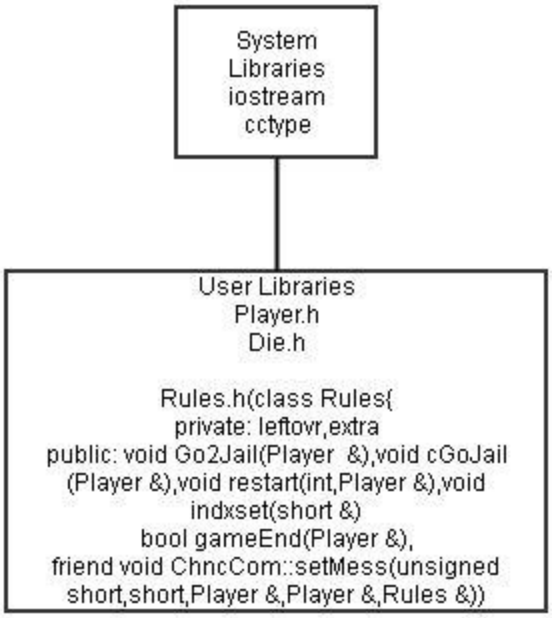
*Include Player and Die header files*

*Define Rules class*

*Declare private member variables*

*Declare public member functions*

*Permit ChncCom to have a friend function*



*Define Go2Jail function that excepts a Player object*

*Instantiate local Die objects and local variables*

*If player has "Get Out of Jail Free" card, ask if they want to use it*

*If yes, use it*

*Else ask to pay fine*

*If yes, pay fine*

*Else Roll Dice a maximum of 3 times*

*If no doubles, pay fine*

*Else pay nothing*

*Else if player does not have card, ask if they want to pay fine*

*If yes, pay*

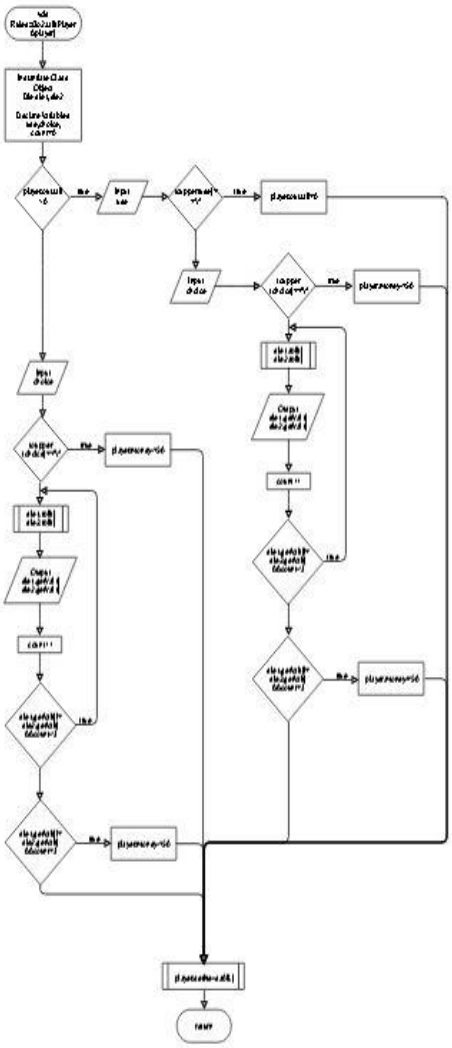
*Else roll a max of 3 times*

*If no doubles, pay fine*

*Else pay nothing*

*Set new spot to "Jail"*

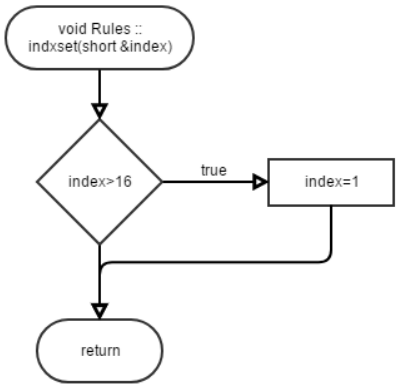
*Return*



*Define indxset function which accepts a short reference variable*

*If index is greater than 16, reset to 1*

*Else, simply return*



*Define cGoJail function*

*Instantiate Die class objects*

*Call roll() function in die to roll both die a maximum*

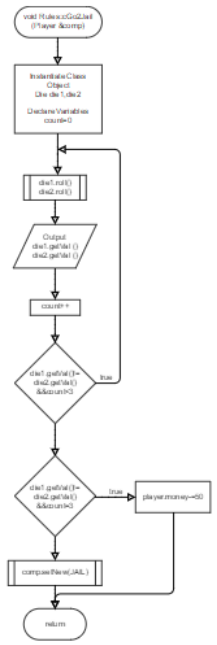
*Of three times*

*If doubles are not rolled, computer must pay fine*

*Else, don't pay*

*Set new spot to Jail*

*Return*



*Define restart function*

*If spot is greater than BRDWALK*

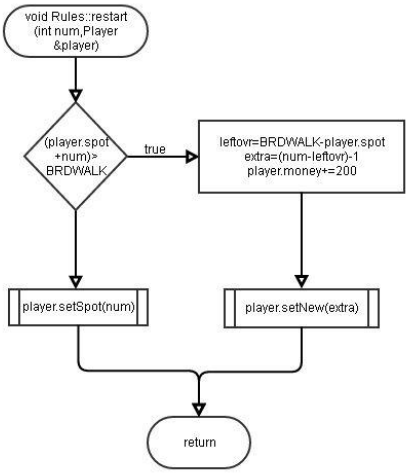
*Pass Go, increase player money by 200*

*Continue past Go for calculated extra spots*

*Set spot to new*

*Else simply set new spot regularly*

*Return*




*Define gameEnd function*

*Declare Boolean local variable and set to*

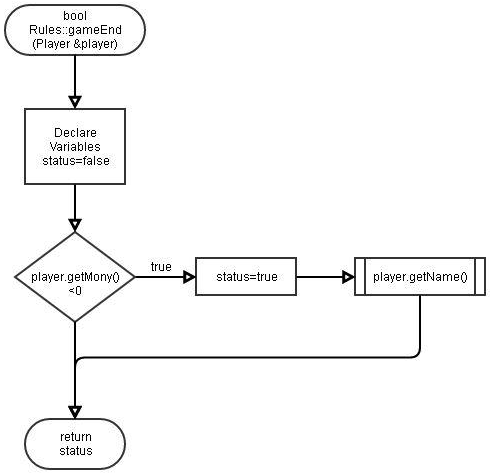
*false*

*If player's money is less than 0*

*Variable is true - return*

*Get player's name*

*Else return status as false*



**Main**

*Include iostream, cstdlib, ctime, cctype, fstream and string libraries*

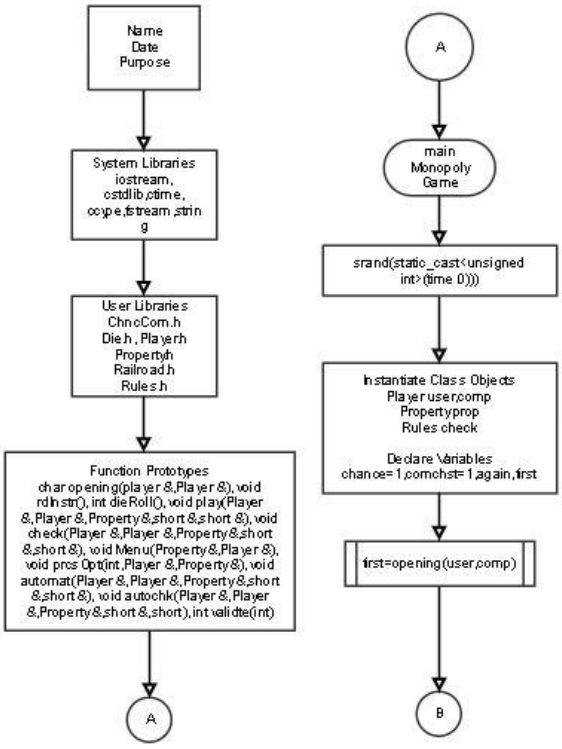
*Include ChncCom, Die, Player, Property, Railroad and Rules header files to access public member functions*

*Declare all function prototypes*

*Set the Random number seed*

*Instantiate class objects and local variables*

*Call Opening function to see who goes first*




*In Opening function, declare local variables*

*Set user name*

*Set opponent name*

*Ask if user wants instructions*

*If yes, call reInstr() to read in file*

*Else, proceed*

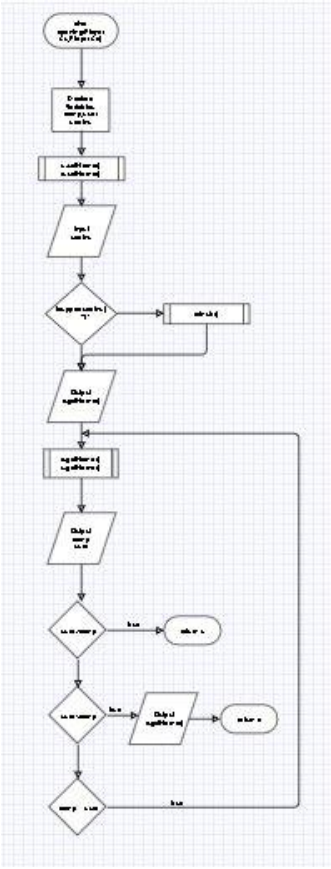
*Enter do/while loop where both players roll to*

*See who is first*

*If comp, return 'c"*

*Else if use, return 'u'*

*Else loop until one of those characters is returned*



*If user needed instruction,*

*Local variables of fstream type are declared*

*Open the file "Instructions.txt"*

*If file is found, read in and output the first line*

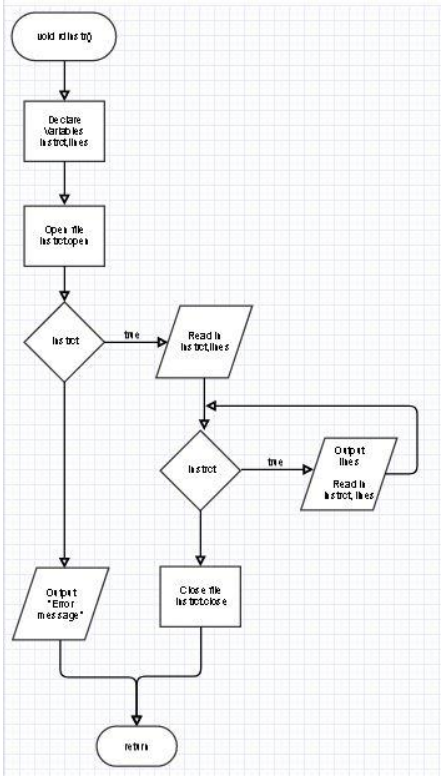
*While there are still lines in the file*

*Read them in and output them*

*Close the file once finished*

*Else display error message that file not found*

*Return*




*Return to main*

*If 'c' was return from Opening, call the automat function*

*Else call the play function for the user*

*When player has returned, automat is called so computer*

*can have a turn*

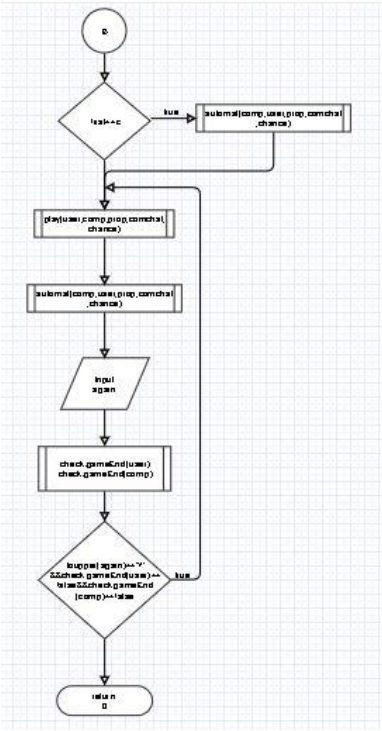
*Once comp returns, ask if user wants to continue playing*

*If yes, check to make sure no one has negative money*

*If result is false, continue game*

*Else, end game*

*Return 0*



*Enter play function if it is player's turn*

*Instantiate Rules class object*

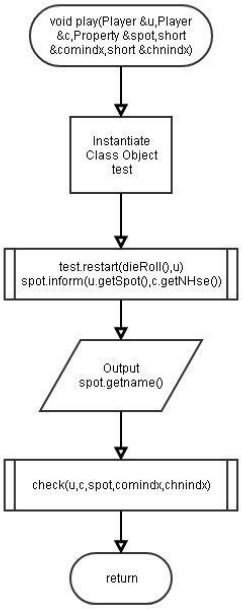
*Always use the class object to test if restart procedure is*

*Necessary*

*Output where player landed*

*Call check function*

*Return to main afterwards*




*Enter check function*

*Instantiate Rules, ChncCom, and Railroad class objects*

*If landed on Chance*

*Get message and output, Increment index and*

*check index, Make sure new spot, if moved, is not*

*"Go To Jail"*

*Else if landed on Community Chest*

*Same instructions as Chance*

*Else if Go to Jail Run procedure*

*Else if color is blank, output message*

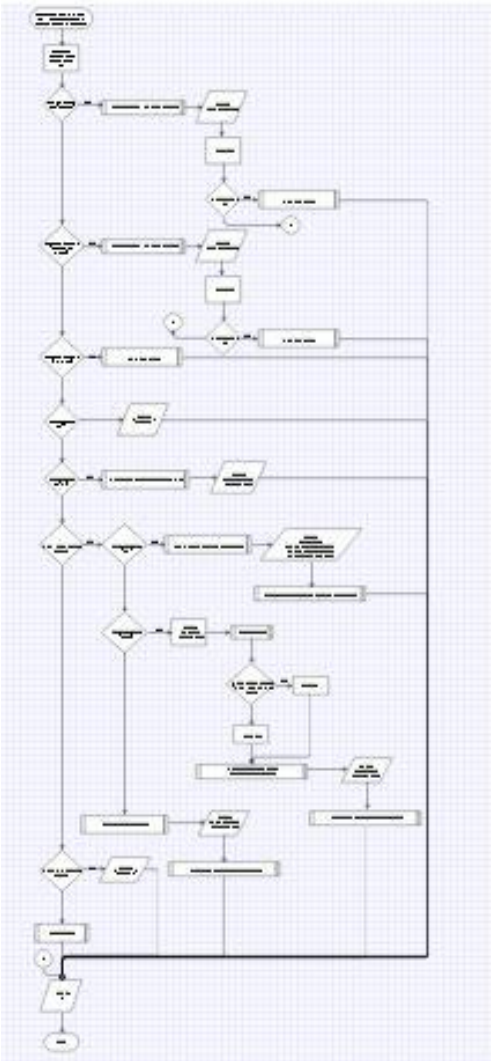
*Else if color is labeled as tax, pay tax*

*Else see if other player or user already owns property*

*If so, pay rent*

*Else, call Menu function*

*Once returned, output player information then return*



*When Menu is called, declare local variables*

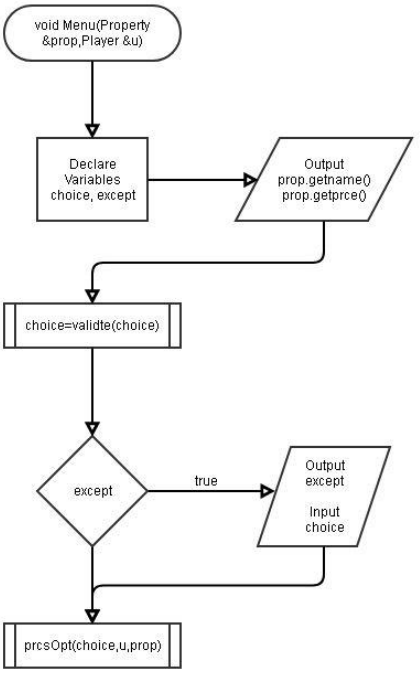
*Output the property name and price*

*Ask if user wants to purchase, visit, build houses,*

*or build hotels*

*Validate with try/catch block*

*Process the choice by calling the function prcsOpt*

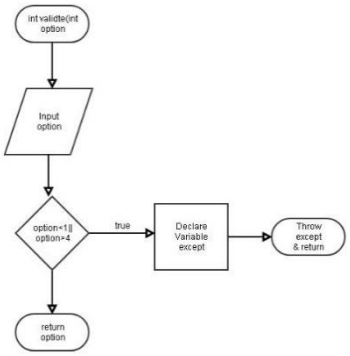



*Enter validte function*

*If the option is not in the range of choices,*

*Throw exception message*

*Else, return the option*




*Enter prcsOpt function to process decision from the Menu*

*If Choice was 1, output the price and deduct from player's*

*money, Set new money total and Add property to player's*

*proprty array*

*Else if choice was 2 output message*

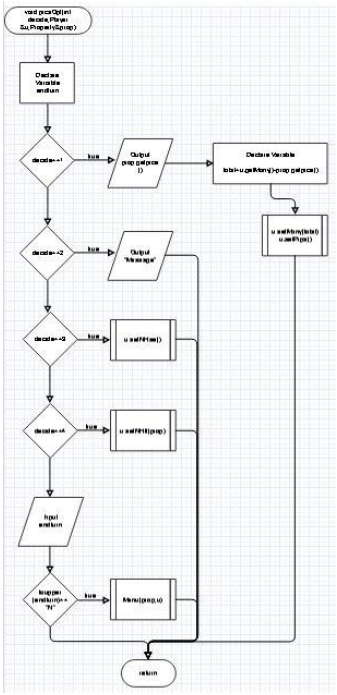
*Else if choice was 3 call setNHse() procedure*

*Else if choice was 4 call setNHtl() procedure*

*Ask if user would like to end their turn*

*If yes, return*

*Else call Menu again*



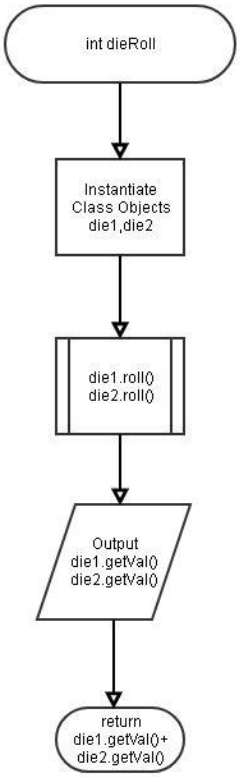
*Enter dieRoll whenever a player needs to roll dice*

*Instantiate two Die objects*

*Roll both die*

*Output the value of each die*

*Return the sum of the die*




*Enter automat function if it is comp's turn*

*Instantiate Rules class object*

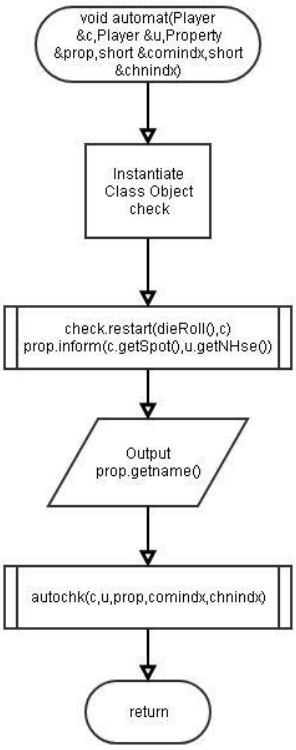
*Always use the class object to test if restart procedure is*

*Necessary*

*Output where player landed*

*Call autochk function*

*Return to main afterwards*



*Enter autochk function*

*Instantiate Rules, ChncCom, and Railroad class objects*

*If landed on Chance*

*Get message and output, Increment index and check index, Make sure new spot, if moved, is not "Go To Jail"*

*Else if landed on Community Chest*

*Same instructions as Chance*

*Else if Go to Jail Run procedure*

*Else if color is blank, output message*

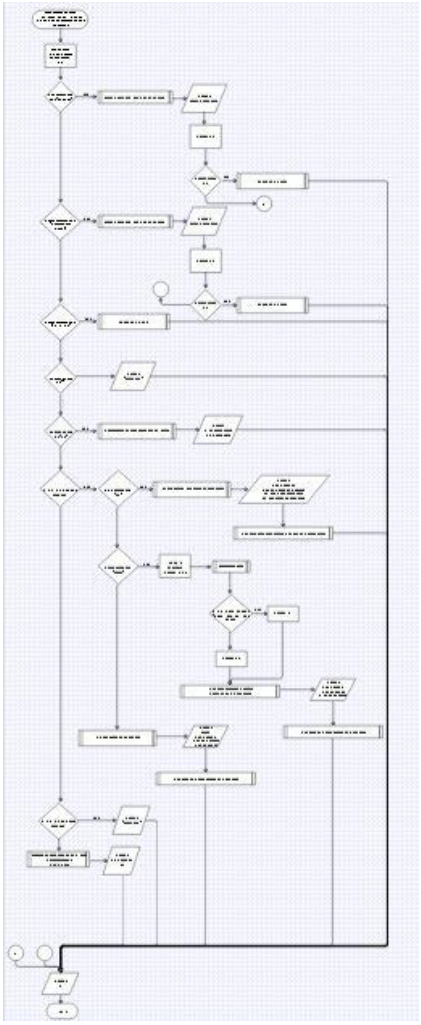
*Else if color is labeled as tax, pay tax*

*Else see if user or comp already owns property*

*If so, pay rent*

*Else, purchase property*

*Once returned, output player information then return*



## Constructs & Concepts Utilized

**iostream Library**

| Name | Frequency | Class | Location |
|---|---|---|---|
| static_cast | 2 | Player<br>main | Line 153<br>Line 40 |
| cout | 155 | ChncCom<br>Player, Rules<br>main | Throughout |
| cin | 12 | Player<br>Rules<br>main | Line 32,75,78,100<br>Line 23,30,48<br>Line 62,181,187,213,348 |

| | | | |
|---|---|---|---|
| try | 1 | main | Line 206 |
| catch | 1 | main | Line 210 |
| throw | 1 | main | Line 190 |

**cstdlib Library**

| Name | Frequency | Class | Location |
|---|---|---|---|
| srand() | 1 | main | Line 40 |
| rand() | 1 | Die | Line 23 |

**ctime Library**

| Name | Frequency | Class | Location |
|---|---|---|---|
| time | 1 | main | Line 40 |

**iomanip Library**

| Name | Frequency | Class | Location |
|---|---|---|---|
| fixed | 1 | Player | Line 152 |
| setprecision() | 1 | Player | Line 152 |
| showpoint | 1 | Player | Line 152 |

**string Library**

| Name | Frequency | Class | Location |
|---|---|---|---|
| string | 41 | All except Die and Rules | throughout |
| getline() | 3 | Player<br>main | Line 32<br>Line 321,324 |

**cctype Library**

| Name | Frequency | Class | Location |
|---|---|---|---|
| toupper() | 7 | Player<br>Rules<br>main | Line 76<br>Lines 24,31,49<br>Lines 64,182,349 |

**fstream Library**

| Name | Frequency | Class | Location |
|---|---|---|---|
| instrct.open() | 1 | main | Line 318 |
| instrct.close() | 1 | main | Line 329 |
| instrct | 5 | main | Line 314, 320-324 |
| fstream | 1 | main | Line 314 |

**Data Types, Conditional Statements & Loops:**

| Data Types | Conditional Statement | Loops |
|---|---|---|
| int | if | for |
| unsigned int | if/else | while |
| unsigned short | if/else if | do/while |
| short | switch | |
| char | | |
| string | | |
| float | | |
| fstream | | |
| bool | | |

**Classes:**

| Class | Features | Members/Access |
|---|---|---|
| Board | Abstract Base Class | Protected, Public |
| ChncCom | Constructor | Private, Public |
| Die | Constructor<br>Inline Member Func. | Private, Public |
| Player | Friend Functions<br>Constructor/Destructor | Private, Protected, Public |
| Property | Derived from Board<br>Constructor | Private, Protected, Public |
| Railroad | Derived from Property<br>Inline Member Func. | Public |
| Rules | Friend Function | Private, Public |

**Friend Functions:**

| Friends | Class |
|---|---|
| friend void ChncCom::setmess(unsigned short,short,Player &,Player &,Rules) | Player/Rules |
| friend void Rules::Go2Jail(Player &) | Player |
| friend void Rules::cGoJail(Player &) | Player |

| | |
|---|---|
| friend void Rules::restart(int, Player &) | Player |
| friend ostream &operator<<(ostream &,const Player &) | Player |

**Virtual Functions**

| Virtual Functions | Concept Demonstrated | Class |
|---|---|---|
| virtual void inform(int,int)=0 | Pure Abstract Function | Board |
| virtual void inform(int,int) | Redefining Abstract Function | Property |
| virtual string getPos(int named) const | Polymorphism | Property |
| virtual string getPos(int named) const | Polymorphism | Railroad |

**Operator Overloading**

| Operator Overloading | Class |
|---|---|
| ostream &operator<<(ostream &,const Player &) | Player |

**Template**

| Template | Class |
|---|---|
| Template <class T1, class T2><br>Void utilRnt(T1 &number,T2 &util) | Player |

**Throwing & Handling Exceptions**

| Exception | Purpose | Location |
|---|---|---|
| Try/Catch Block<br>Execption Thrown | Input Validation | main Line 206-214<br>main Line 190 |

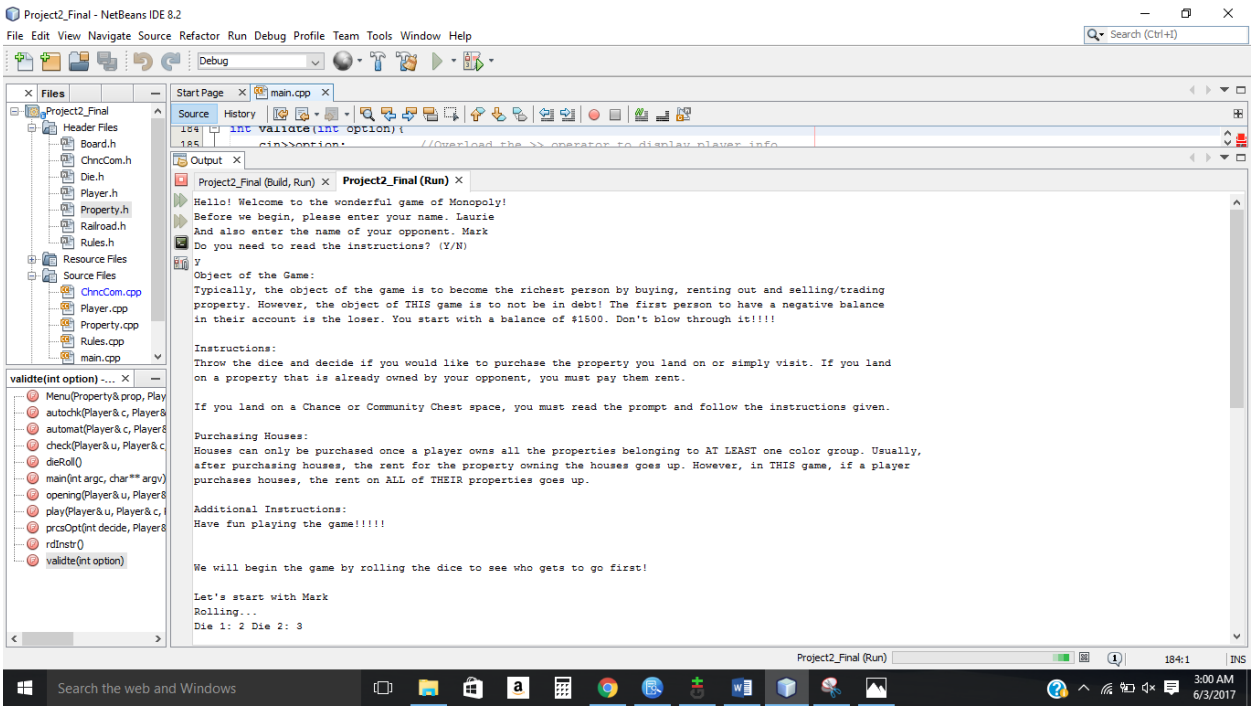
**Enums**

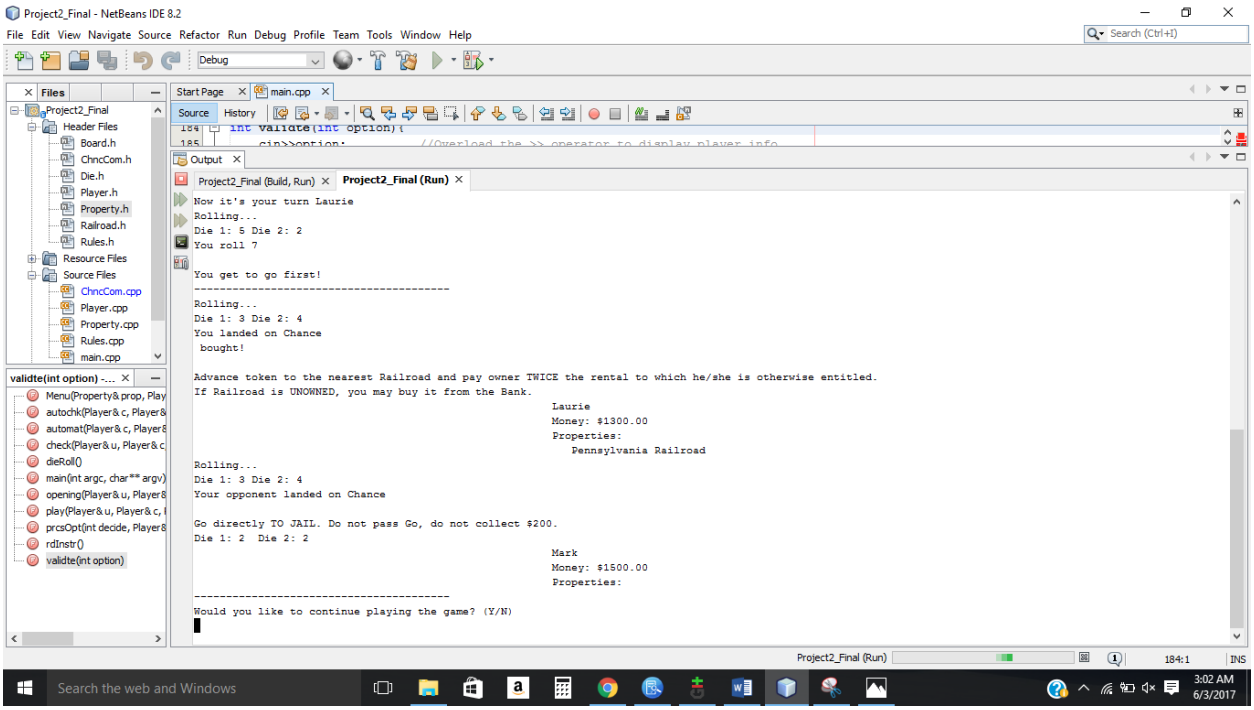
| Enumerator | Class |
|---|---|
| enum Name | Player |

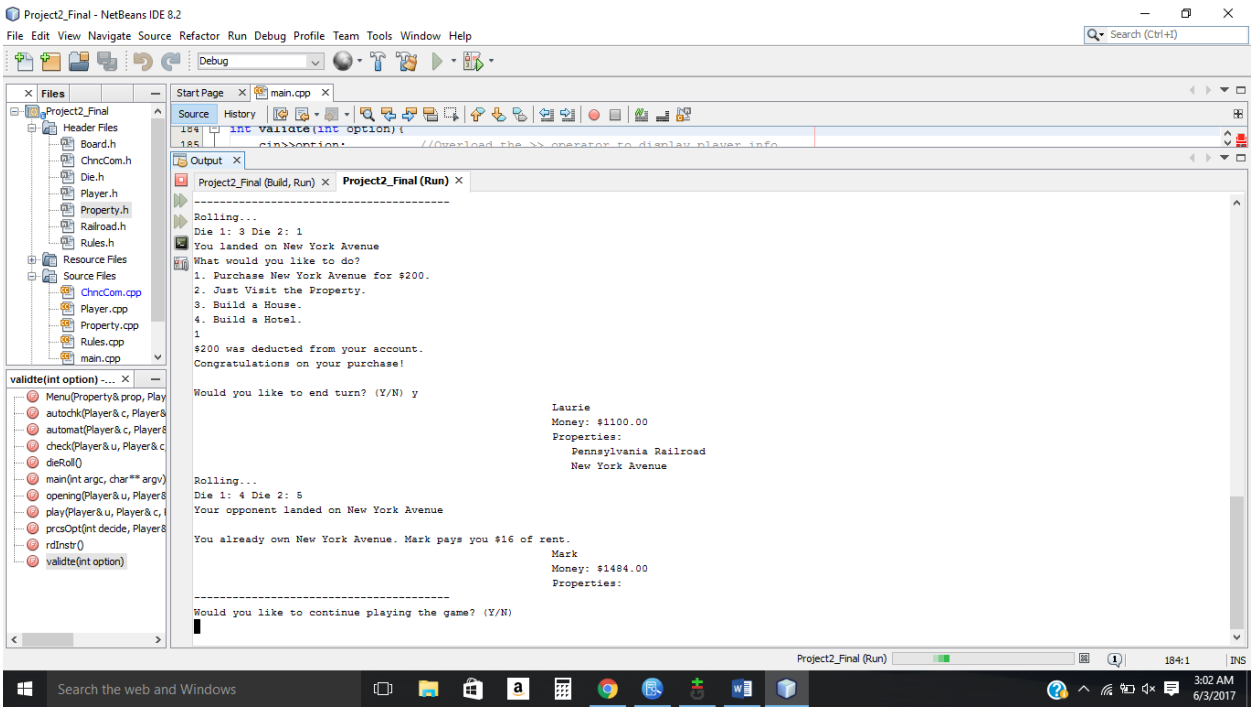
# TOTAL LINES OF CODE: 1,776

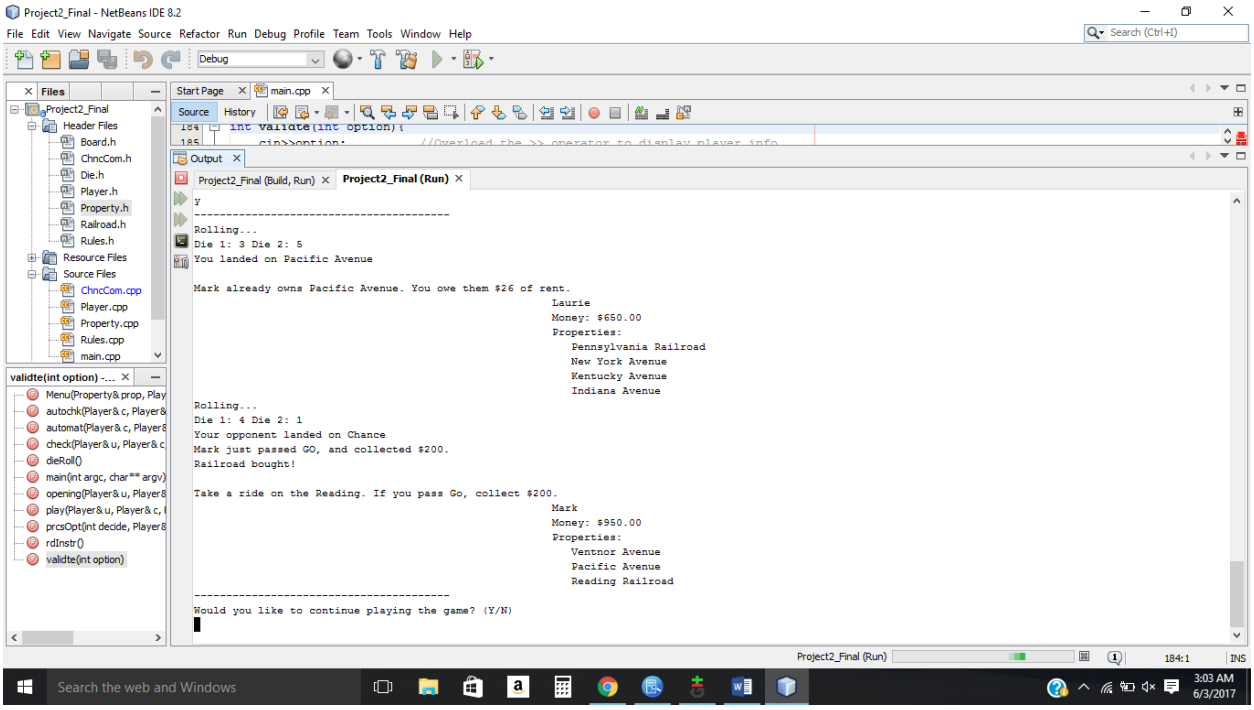
## Proof of a Working Product

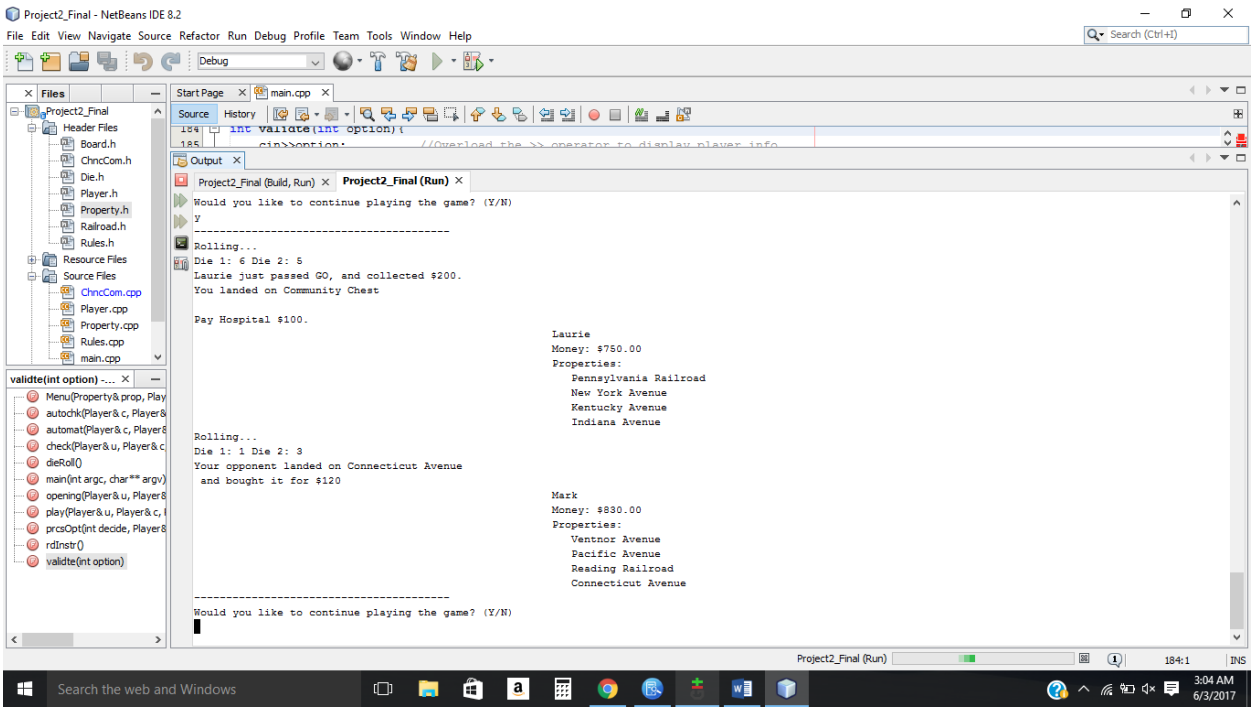
In the event, that my program does not work once it reaches Dr. Lehr, I have provided some screenshots that prove that the program did work at one time on the next few pages.

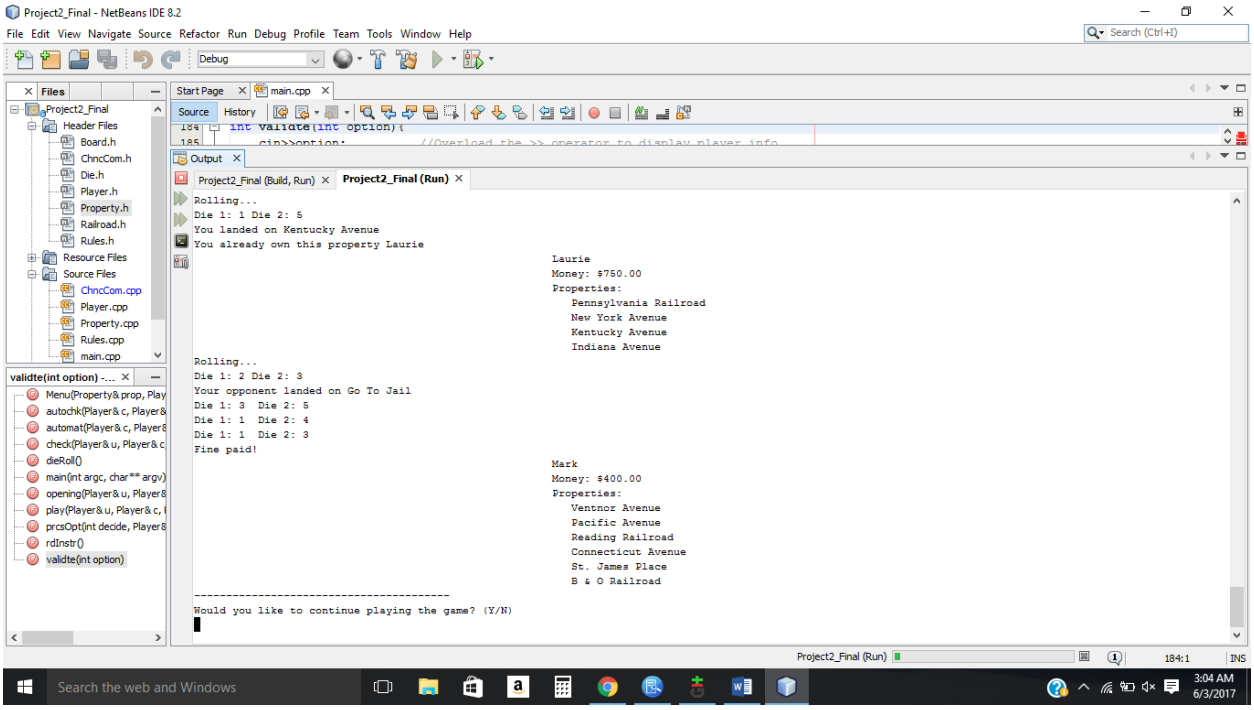


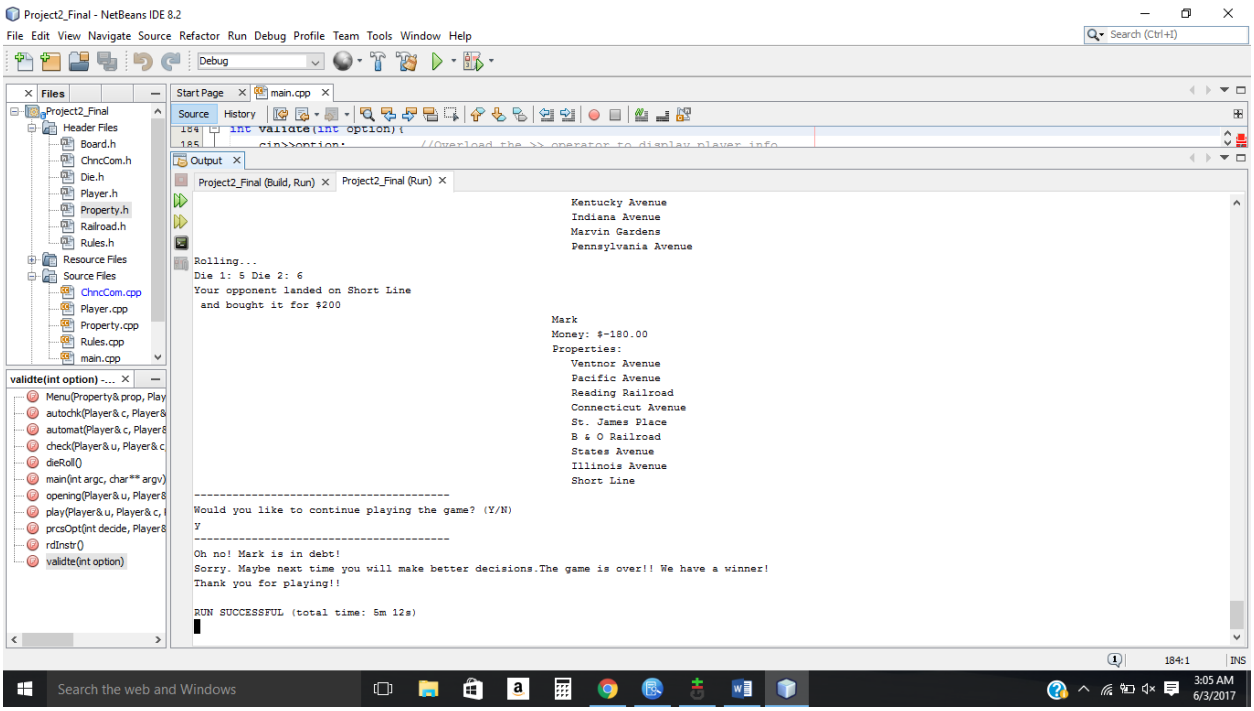



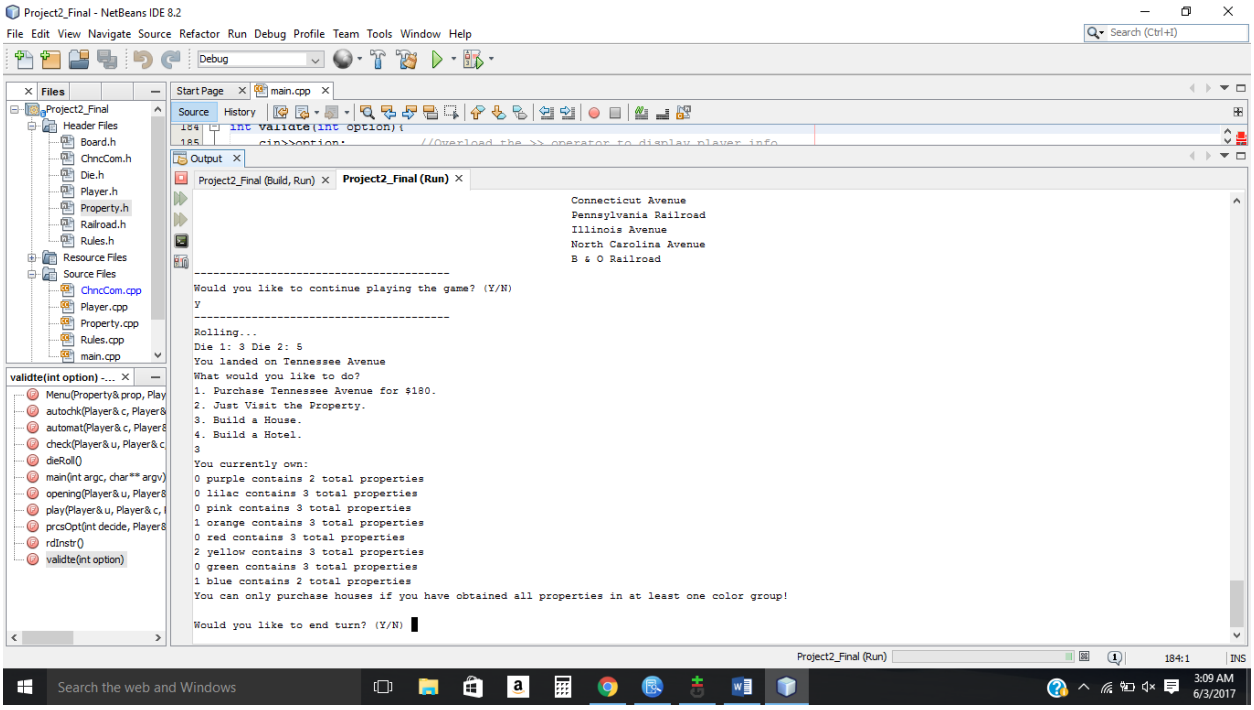

Project2_Final - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Debug
Files
Project2_Final
Header Files
Board.h
ChncCom.h
Die.h
Player.h
Property.h
Railroad.h
Rules.h
Resource Files
Source Files
ChncCom.cpp
Player.cpp
Property.cpp
Rules.cpp
main.cpp
validte(int option) -...
Menu(Property& prop, Play
autochk(Player& c, Player&
automat(Player& c, Player&
check(Player& u, Player& c
dieRoll()
main(int argc, char** argv)
opening(Player& u, Player&
play(Player& u, Player& c,
prcsOpt(int decide, Player&
rdInstr()
validte(int option)
Start Page
main.cpp
Source
History
int validate(int option){
cin>>option;
//Overload the >> operator to display player info
Output
Project2_Final (Build, Run)
Project2_Final (Run)
Rolling...
Die 1: 3 Die 2: 1
You landed on New York Avenue
What would you like to do?
1. Purchase New York Avenue for $200.
2. Just Visit the Property.
3. Build a House.
4. Build a Hotel.
1
$200 was deducted from your account.
Congratulations on your purchase!
Would you like to end turn? (Y/N) y
Laurie
Money: $1100.00
Properties:
Pennsylvania Railroad
New York Avenue
Rolling...
Die 1: 4 Die 2: 5
Your opponent landed on New York Avenue
You already own New York Avenue. Mark pays you $16 of rent.
Mark
Money: $1484.00
Properties:
Would you like to continue playing the game? (Y/N)
Project2_Final (Run)
184:1
INS
Search the web and Windows
3:02 AM
6/3/2017


Project2_Final - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Debug
Files
Project2_Final
Header Files
Board.h
ChncCom.h
Die.h
Player.h
Property.h
Railroad.h
Rules.h
Resource Files
Source Files
ChncCom.cpp
Player.cpp
Property.cpp
Rules.cpp
main.cpp
validte(int option) -...
Menu(Property& prop, Play
autochk(Player& c, Player&
automat(Player& c, Player&
check(Player& u, Player& c
dieRoll()
main(int argc, char** argv)
opening(Player& u, Player&
play(Player& u, Player& c,
prcsOpt(int decide, Player&
rdInstr()
validte(int option)
Start Page
main.cpp
Source
History
int validate(int option){
cin>>option;
//Overload the >> operator to display player info
Output
Project2_Final (Build, Run)
Project2_Final (Run)
y
Rolling...
Die 1: 3 Die 2: 5
You landed on Pacific Avenue
Mark already owns Pacific Avenue. You owe them $26 of rent.
Laurie
Money: $650.00
Properties:
Pennsylvania Railroad
New York Avenue
Kentucky Avenue
Indiana Avenue
Rolling...
Die 1: 4 Die 2: 1
Your opponent landed on Chance
Mark just passed GO, and collected $200.
Railroad bought!
Take a ride on the Reading. If you pass Go, collect $200.
Mark
Money: $950.00
Properties:
Ventnor Avenue
Pacific Avenue
Reading Railroad
Would you like to continue playing the game? (Y/N)
Project2_Final (Run)
184:1
INS
Search the web and Windows
3:03 AM
6/3/2017


Project2_Final - NetBeans IDE 8.2
Would you like to continue playing the game? (Y/N)
y
Rolling...
Die 1: 6 Die 2: 5
Laurie just passed GO, and collected $200.
You landed on Community Chest
Pay Hospital $100.
Laurie
Money: $750.00
Properties:
Pennsylvania Railroad
New York Avenue
Kentucky Avenue
Indiana Avenue
Rolling...
Die 1: 1 Die 2: 3
Your opponent landed on Connecticut Avenue
and bought it for $120
Mark
Money: $830.00
Properties:
Ventnor Avenue
Pacific Avenue
Reading Railroad
Connecticut Avenue
Would you like to continue playing the game? (Y/N)


Project2_Final - NetBeans IDE 8.2
Rolling...
Die 1: 1 Die 2: 5
You landed on Kentucky Avenue
You already own this property Laurie
Laurie
Money: $750.00
Properties:
Pennsylvania Railroad
New York Avenue
Kentucky Avenue
Indiana Avenue
Rolling...
Die 1: 2 Die 2: 3
Your opponent landed on Go To Jail
Die 1: 3 Die 2: 5
Die 1: 1 Die 2: 4
Die 1: 1 Die 2: 3
Fine paid!
Mark
Money: $400.00
Properties:
Ventnor Avenue
Pacific Avenue
Reading Railroad
Connecticut Avenue
St. James Place
B & O Railroad
Would you like to continue playing the game? (Y/N)


Project2_Final - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Debug
Files
Project2_Final
Header Files
Board.h
ChncCom.h
Die.h
Player.h
Property.h
Railroad.h
Rules.h
Resource Files
Source Files
ChncCom.cpp
Player.cpp
Property.cpp
Rules.cpp
main.cpp
validte(int option) -...
Menu(Property& prop, Play
autochk(Player& c, Player&
automat(Player& c, Player&
check(Player& u, Player& c
dieRoll()
main(int argc, char** argv)
opening(Player& u, Player&
play(Player& u, Player& c,
prcsOpt(int decide, Player&
rdInstr()
validte(int option)
Start Page
main.cpp
Source
History
int validate(int option){
cin>>option;
//Overload the >> operator to display player info
Output
Project2_Final (Build, Run)
Project2_Final (Run)
Kentucky Avenue
Indiana Avenue
Marvin Gardens
Pennsylvania Avenue
Rolling...
Die 1: 5 Die 2: 6
Your opponent landed on Short Line
and bought it for $200
Mark
Money: $-180.00
Properties:
Ventnor Avenue
Pacific Avenue
Reading Railroad
Connecticut Avenue
St. James Place
B & O Railroad
States Avenue
Illinois Avenue
Short Line
Would you like to continue playing the game? (Y/N)
y
Oh no! Mark is in debt!
Sorry. Maybe next time you will make better decisions.The game is over!! We have a winner!
Thank you for playing!!
RUN SUCCESSFUL (total time: 5m 12s)
184:1
INS
Search the web and Windows
3:05 AM
6/3/2017


Project2_Final - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Debug
Files
Project2_Final
Header Files
Board.h
ChncCom.h
Die.h
Player.h
Property.h
Railroad.h
Rules.h
Resource Files
Source Files
ChncCom.cpp
Player.cpp
Property.cpp
Rules.cpp
main.cpp
validte(int option) -...
Menu(Property& prop, Play
autochk(Player& c, Player&
automat(Player& c, Player&
check(Player& u, Player& c
dieRoll()
main(int argc, char** argv)
opening(Player& u, Player&
play(Player& u, Player& c,
prcsOpt(int decide, Player&
rdInstr()
validte(int option)
Start Page
main.cpp
Source
History
int validate(int option){
cin>>option;
//Overload the >> operator to display player info
Output
Project2_Final (Build, Run)
Project2_Final (Run)
Connecticut Avenue
Pennsylvania Railroad
Illinois Avenue
North Carolina Avenue
B & O Railroad
Would you like to continue playing the game? (Y/N)
y
Rolling...
Die 1: 3 Die 2: 5
You landed on Tennessee Avenue
What would you like to do?
1. Purchase Tennessee Avenue for $180.
2. Just Visit the Property.
3. Build a House.
4. Build a Hotel.
3
You currently own:
0 purple contains 2 total properties
0 lilac contains 3 total properties
0 pink contains 3 total properties
1 orange contains 3 total properties
0 red contains 3 total properties
2 yellow contains 3 total properties
0 green contains 3 total properties
1 blue contains 2 total properties
You can only purchase houses if you have obtained all properties in at least one color group!
Would you like to end turn? (Y/N)
Project2_Final (Run)
184:1
INS
Search the web and Windows
3:09 AM
6/3/2017

## References

1. Dr. Lehr's Lectures & Slides

2. "Starting Out with C++: From Control Structures through Objects" Gaddis, Tony. 8th Edition. (Textbook)

## Program (Courier New Font 9)

```
/*
 * File:   Board.h
 * Author: Laurie Guimont
 * Created on June 2, 2017, 5:46 PM
 * Purpose: Board Class Specifications
 */

#ifndef BOARD_H
#define BOARD_H

#include <string>
using namespace std;

class Board{
    protected:
        string name;       //Property Names
        int price;         //Purchase Price for Properties
        string color;      //Colors for Property Color Groups
    public:
        Board(){
            name="";
            price=0;
            color="";
        }
        string getname()const{
            return name;
        }
        int getprce()const{
            return price;
        }
        string getcolr()const{
            return color;
        }
        string colList(int index){
            string colors[8]={"purple","lilac","pink","orange",
                              "red","yellow","green","blue"};

            return colors[index];
        }
        virtual void inform(int,int)=0;
};

#endif /* BOARD_H */

#ifndef CHNCCOM_H
#define CHNCCOM_H

#include <string>
using namespace std;
```

```
class Player;
class Rules;

class ChncCom{
    private:
        string message;
    public:
        ChncCom();
        void setMess(unsigned short,short,Player &,Player &,Rules &);  //Friend
        string getMess()const;
};

#endif /* CHNCCOM_H */

#ifndef DIE_H
#define DIE_H

#include <cstdlib>
using namespace std;

class Die{
    private:
        int value;
    public:
        Die(){
            value=0;
        }
        void roll(){
            int MIN=1, MAX=6;
            value=(rand()%(MAX-MIN+1))+MIN;
        }
        int getVal(){
            return value;
        }
};

#endif /* DIE_H */

#ifndef PLAYER_H
#define PLAYER_H
#include "ChncCom.h"
#include "Rules.h"
#include "Property.h"

#include <iostream>
using namespace std;

class Player;
ostream &operator<<(ostream &,const Player &);

class Player{
    private:
        string name;             //Player Name
        unsigned short nProps;  //Number of properties owned by player
        int *proprty;           //Array of property names owned by player
        int spot;               //Property Player has landed on
        int outJail;            //Get Out of Jail Free Card
        int nHouses;            //Number of houses owned by player
        int nHotels;            //Number of hotels owned by player
    protected:
        int money;              //Amount of money player has
    public:
```

```
        Player();
        ~Player(){delete []proprty;};
        void setName();
        void setMony(int);
        void setPrps();
        void setNew(int);
        void setSpot(int);
        void setNHse();
        void setNHtl(Property &);
        void payRent(int);
        string getName()const;
        int getMony()const;
        unsigned short getNPrp()const;
        int getSpot()const;
        void getPrps();
        bool findPrp(int);
        int getNHse()const;
        int getNHtl()const;

        //Friends
        friend void ChncCom::setMess(unsigned short,short,Player &,Player &,Rules &);
        friend void Rules::Go2Jail(Player &);
        friend void Rules::cGoJail(Player &);
        friend void Rules::restart(int,Player &);

        //Overload << Operator
        friend ostream &operator<<(ostream &,const Player &);
};

#endif /* PLAYER_H */

#ifndef PROPERTY_H
#define PROPERTY_H
#include "Board.h"

#include <string>
using namespace std;

enum Name {GO,MDTRRNN,COMMCH1,BALTIC,INCOME,READRR,ORIENTL,CHANCE1,
            VERMONT,CONNECT,JAIL,STCHRLS,ELECTRC,STATES,VIRGNIA,PENNRR,
            STJAMES,COMMCH2,TENNESE,NEWYORK,PARKING,KENTCKY,CHANCE2,INDIANA,
            ILLNOIS,BNORR,ATLANTC,VENTNOR,WATERWK,MARVIN,GO2JAIL,PACIFIC,
            NCARLNA,COMMCH3,PENNSYL,SHORTLN,CHANCE3,PARK,LUXTAX,BRDWALK};

class Property : public Board{
    private:
        int hseCost;       //Cost per house
        int htlCost;       //Cost per hotel
        int maxColr;       //Number of properties in a color group
    protected:
        int rent;          //Amount of rent to charge players
    public:
        Property();
        int getrent()const;
        int getHseC()const;
        int getHtlC()const;
        int setcMax(string);

        template <class T1,class T2>
        void utilRnt(T1 &number,T2 &util){
            if(number==1)rent=4*util;
            else rent=10*util;
        }
```

```
        virtual string getPos(int named)const{
            string postion;
            if(named==READRR)
                postion="Reading";
            else if(named==PENNRR)
                postion="Pennsylvania";
            else if(named==BNORR)
                postion="B & O";
            else if(named==SHORTLN)
                postion="Short Line";

            return postion;
        }

        virtual void inform(int,int);
};

#endif /* PROPERTY_H */

#ifndef RAILROAD_H
#define RAILROAD_H
#include "Property.h"
#include "Player.h"

#include <iostream>
#include <string>
using namespace std;

class Railroad:public Property{
    public:
        int setrent(Player &player,int current){
            Railroad RR;
            int count=0;
            if(player.findPrp(READRR)==true)
                count++;
            if(player.findPrp(PENNRR)==true)
                count++;
            if(player.findPrp(BNORR)==true)
                count++;
            if(player.findPrp(SHORTLN)==true)
                count++;
            RR.inform(current,count);

            return RR.rent;
        };
        virtual string getPos(int named) const{
            string postion;

            if(named==READRR)
                postion="first";
            else if(named==PENNRR)
                postion="second";
            else if(named==BNORR)
                postion="third";
            else if(named==SHORTLN)
                postion="fourth";
            return postion;
        }
};

#endif /* RAILROAD_H */
```

```
#ifndef RULES_H
#define RULES_H
#include "ChncCom.h"

class Player;

class Rules{
    private:
        int leftovr;       //Used in "Pass Go" instructions
        int extra;         //Used in "Pass Go" instructions
    public:
        void Go2Jail(Player &);
        void cGoJail(Player &);
        void restart(int,Player &);  //Friend
        void indxset(short &);
        bool gameEnd(Player &);
        friend void ChncCom::setMess(unsigned short,short,Player &,Player &,
            Rules &);
};

#endif /* RULES_H */

#include "ChncCom.h"
#include "Player.h"
#include "Property.h"
#include "Rules.h"
#include "Railroad.h"
#include "Die.h"

#include <iostream>
#include <string>
using namespace std;

ChncCom::ChncCom(){
    message="";
}

void ChncCom::setMess(unsigned short pick,short index,Player &player,
        Player &opp,Rules &rule){
    Railroad RR;
    Property spot;
    Die die1,die2;
    if(pick==1){
        switch(index){
            case 1:{
                message="Advance token to the nearest Railroad and pay owner "
                        "TWICE the rental to which he/she is otherwise "
                        "entitled. \nIf Railroad is UNOWNED, you may buy it "
                        "from the Bank.";
                if(player.getSpot()==CHANCE1)
                    player.setNew(PENNRR);
                else if(player.getSpot()==CHANCE2)
                    player.setNew(BNORR);
                else if(player.getSpot()==CHANCE3){
                    player.setNew(READRR);
                    player.money+=200;
                    cout<<player.getName()<<" just passed GO, and collected ";
                    cout<<"$200."<<endl;
                }
                if(opp.findPrp(player.getSpot())==true){
                    player.money-=(RR.setrent(opp,player.spot)*2);
                    cout<<"Charged: "<<RR.setrent(opp,player.spot);
                    opp.money+=(RR.setrent(opp,player.spot)*2);
```

```
        }
        else if(opp.findPrp(player.getSpot())!=true&&
                player.findPrp(player.getSpot())!=true){
            player.setPrps();
            player.money-=200;
            cout<<spot.getname()<<" bought!"<<endl;
        }
        break;
    }
    case 2:{
        message="Go directly TO JAIL. Do not pass Go, do not collect "
                "$200.";
        player.setNew(GO2JAIL);
        break;
    }
    case 3:{
        message="Take a ride on the Reading. If you pass Go, collect "
                "$200.";
        player.setNew(READRR);
        player.money+=200;
        cout<<player.getName()<<" just passed GO, and collected ";
        cout<<"$200."<<endl;
        if(opp.findPrp(player.getSpot())==true){
            player.payRent(RR.setrent(opp,player.spot));
            cout<<endl;
            cout<<"Your opponent already owns this railroad. You owe ";
            cout<<"them $"<<RR.setrent(opp,player.spot)<<" of rent.\n";
            opp.setMony(opp.getMony()+RR.setrent(opp,player.spot));
        }
        else if(player.findPrp(player.getSpot())==true)
            cout<<"You already own "<<spot.getname()<<endl;
        else{
            player.money-=200;
            player.setPrps();
            cout<<"Railroad bought!"<<endl;
        }
        break;
    }
    case 4:{
        message="Advance to Illinois Ave.";
        if(player.getSpot()==CHANCE3){
            player.money+=200;
            cout<<player.getName()<<" just passed GO, and collected ";
            cout<<"$200."<<endl;
        }
        player.setNew(ILLNOIS);
        if(opp.findPrp(player.getSpot())==true){
            spot.inform(player.getSpot(),opp.nHouses);
            player.payRent(spot.getrent());
            cout<<endl;
            cout<<"Your opponent already owns "<<spot.getname()<<". You owe ";
            cout<<"them $"<<spot.getrent()<<" of rent."<<endl;
            player.setMony(player.getMony()+spot.getrent());
        }
        else if(player.findPrp(player.getSpot())==true)
            cout<<"You already own "<<spot.getname()<<endl;
        else{
            player.money-=240;
            player.setPrps();
            cout<<"Bought!"<<endl;
        }
        break;
    }
```

```
case 5:{
    message="Bank pays you dividend of $50.";
    player.money+=50;
    break;
}
case 6:{
    message="Advance token to the nearest Railroad and pay owner "
            "TWICE the rental to which he/she is otherwise "
            "entitled. \nIf Railroad is UNOWNED, you may buy it "
            "from the Bank.";
    if(player.getSpot()==CHANCE1)
        player.setNew(PENNRR);
    else if(player.getSpot()==CHANCE2)
        player.setNew(BNORR);
    else if(player.getSpot()==CHANCE3){
        player.setNew(READRR);
        player.money+=200;
        cout<<player.getName()<<" just passed GO, and collected ";
        cout<<"$200."<<endl;
    }
    if(opp.findPrp(player.getSpot())==true){
        player.money-=(RR.setrent(opp,player.spot)*2);
        cout<<"Charged: "<<RR.setrent(opp,player.spot);
        opp.money+=(RR.setrent(opp,player.spot)*2);
    }
    else if(opp.findPrp(player.getSpot())!=true&&
            player.findPrp(player.getSpot())!=true){
        player.setPrps();
        player.money-=200;
        cout<<spot.getname()<<" bought!"<<endl;
    }
    break;
}
case 7:{
    message="Take a walk on the Boardwalk. Advance token to "
            "Boardwalk.";
    player.setNew(BRDWALK);
    if(opp.findPrp(player.getSpot())==true){
        spot.inform(player.getSpot(),opp.nHouses);
        player.payRent(spot.getrent());
        cout<<endl;
        cout<<"Your opponent already owns "<<spot.getname()<<". You owe ";
        cout<<"them $"<<spot.getrent()<<" of rent."<<endl;
        player.setMony(player.getMony()+spot.getrent());
    }
    else if(player.findPrp(player.getSpot())==true)
        cout<<"You already own "<<spot.getname()<<endl;
    else{
        player.money-=400;
        player.setPrps();
        cout<<"Bought!"<<endl;
    }
    break;
}
case 8:{
    message="Get out of Jail Free!! This card may be kept until "
            "needed, or sold.";
    player.outJail++;
    break;
}
case 9:{
    message="You have been elected Chairman of the Board. Pay each "
            "player $50.";
```

```
            player.money-=50;
            opp.money+=50;
            break;
        }
        case 10:{
            message="Advance to St. Charles Place. If you pass Go, collect "
                    "$200.";
            if(player.getSpot()==CHANCE2||player.getSpot()==CHANCE3){
                player.money+=200;
                cout<<player.getName()<<" just passed GO, and collected ";
                cout<<"$200."<<endl;
            }
            player.setNew(STCHRLS);
            if(opp.findPrp(player.getSpot())==true){
                spot.inform(player.getSpot(),opp.nHouses);
                player.payRent(spot.getrent());
                cout<<endl;
                cout<<"Your opponent already owns "<<spot.getname()<<". You owe ";
                cout<<"them $"<<spot.getrent()<<" of rent."<<endl;
                player.setMony(player.getMony()+spot.getrent());
            }
            else if(player.findPrp(player.getSpot())==true)
                cout<<"You already own "<<spot.getname()<<endl;
            else{
                player.money-=140;
                player.setPrps();
                cout<<"Bought!"<<endl;
            }
            break;
        }
        case 11:{
            message="Make general repairs on all your property. For each "
                    "House, pay $25. For each Hotel, pay $100.";
            if(player.nHouses>=1){
                player.money-=(player.nHouses*25);
            }
            if(player.nHotels>=1&&player.nHouses>=1){
                player.money-=(player.nHotels*100);
            }
            else if(player.nHotels>=1){
                player.money-=(player.nHotels*100);
            }
            break;
        }
        case 12:{
            message="Your building and loan matures. Collect $150.";
            player.money+=150;
            break;
        }
        case 13:{
            message="Pay Poor Tax of $15.";
            player.money-=15;
            break;
        }
        case 14:{
            message="Advance to Go. Collect $200.";
            player.setNew(GO);
            player.money+=200;
            cout<<player.getName()<<" just passed GO, and collected ";
            cout<<"$200."<<endl;
            break;
        }
        case 15:{
```

```
            message="Go back 3 spaces.";
            player.setNew(player.getSpot()-3);
            if(opp.findPrp(player.getSpot())==true){
                spot.inform(player.getSpot(),opp.nHouses);
                player.payRent(spot.getrent());
                cout<<endl;
                cout<<"Your opponent already owns "<<spot.getname();
                cout<<". You owe them $"<<spot.getrent()<<" of rent."<<endl;
                player.setMony(player.getMony()+spot.getrent());
            }
            else if(player.findPrp(player.getSpot())==true)
                cout<<"You already own "<<spot.getname()<<endl;
            else{
                player.money-=spot.getprce();
                player.setPrps();
                cout<<"Bought "<<spot.getname()<<"!"<<endl;
            }
            break;
        }
        case 16:{
            message="Advance token to nearest Utility. If UNOWNED you may "
                    "buy it from the Bank. If OWNED, throw dice and pay "
                    "owner a total ten times the amount thrown.";
            //Roll the dice
            die1.roll();
            die2.roll();
            int total=die1.getVal()+die2.getVal();

            //Advance Procedure
            if(player.getSpot()==CHANCE1||player.getSpot()==CHANCE3){
                player.setNew(ELECTRC);
                if(player.getSpot()==CHANCE3){
                    player.money+=200;
                    cout<<player.getName()<<" just passed GO, and ";
                    cout<<"collected $200."<<endl;
                }
            }
            else
                player.setNew(WATERWK);

            //Owned or Not
            if(opp.findPrp(player.getSpot())==true){
                int number=2;
                spot.utilRnt(number,total);
                player.payRent(spot.getrent());
                cout<<"Rolled: "<<total<<endl;
                cout<<"Charged: $"<<spot.getrent();
                opp.money+=spot.getrent();
            }
            else if(opp.findPrp(player.getSpot())!=true&&
                player.findPrp(player.getSpot())!=true){
                player.setPrps();
                player.money-=150;
                cout<<"Bought "<<spot.getname()<<"!"<<endl;
            }
            else
               cout<<"You already own "<<spot.getname()<<endl;
            break;
        }
    }
}
else if(pick==2){
    switch(index){
```

```
case 1:{
    message="Pay Hospital $100.";
    player.money-=100;
    break;
}
case 2:{
    message="Christmas Fund matures. Collect $100.";
    player.money+=100;
    break;
}
case 3:{
    message="Advance to Go. Collect $200.";
    player.setNew(GO);
    player.money+=200;
    cout<<player.getName()<<" just passed GO, and collected ";
    cout<<"$200."<<endl;
    break;
}
case 4:{
    message="Get out of Jail free!!! This card may be kept until "
            "needed, or sold.";
    player.outJail++;
    break;
}
case 5:{
    message="Go TO JAIL. Go directly to Jail. Do not pass Go, do "
            "not collect $200.";
    player.setNew(GO2JAIL);
    break;
}
case 6:{
    message="Bank error in your favor. Collect $200.";
    player.money+=200;
    break;
}
case 7:{
    message="You are assessed for street repairs. $40 per House, "
            "$115 per Hotel.";
    if(player.nHouses>=1){
        player.money-=(player.nHouses*40);
    }
    if(player.nHotels>=1&&player.nHouses>=1){
        player.money-=(player.nHotels*115);
    }
    else if(player.nHotels>=1){
        player.money-=(player.nHotels*115);
    }
    break;
}
case 8:{
    message="You have won second prize in a beauty contest. "
            "Collect $10.";
    player.money+=10;
    break;
}
case 9:{
    message="Income Tax refund. Collect $20.";
    player.money+=20;
    break;
}
case 10:{
    message="Grand Opera Opening. Collect $50 from every player "
            "for opening night seats.";
```

```
                player.money+=50;
                opp.money-=50;
                break;
            }
            case 11:{
                message="From sale of stock you get $45.";
                player.money+=45;
                break;
            }
            case 12:{
                message="Pay school tax of $150.";
                player.money-=150;
                break;
            }
            case 13:{
                message="You inherit $100.";
                player.money+=100;
                break;
            }
            case 14:{
                message="Receive for services $25.";
                player.money+=25;
                break;
            }
            case 15:{
                message="Life insurance matures. Collect $100.";
                player.money+=100;
                break;
            }
            case 16:{
                message="Doctor's fee. Pay $50.";
                player.money-=50;
                break;
            }
        }
    }
}

string ChncCom::getMess() const{
    return message;
}

#include <iostream>
#include <iomanip>
#include <cctype>
#include <string>
using namespace std;

#include "Player.h"
#include "Property.h"


Player::Player(){
    name="";
    money=1500;
    nProps=0;
    proprty=new int[40];
    for(int i=0;i<=40;i++)
        *(proprty+i)=0;
    spot=GO;
    outJail=0;
    nHouses=0;
    nHotels=0;
```

```
}

void Player::setName(){
    getline(cin,name);
}

void Player::setMony(int mon){
    money=mon;
}

void *Player::setPrps(){
    *(proprty+nProps)=spot;
    nProps++;
}

void Player::setNew(int s){
    spot=s;
}

void Player::setSpot(int s){
    spot+=s;
}

void Player::setNHse(){
    Property prop;
    int colors[8];
    int count=0;
    char add;
    int numHse;

    for(int i=0;i<8;i++){
        for(int j=0;j<nProps;j++){
            prop.inform(*(proprty+j),nHouses);
            if(prop.getcolr()==prop.colList(i))
                count++;
        }
        *(colors+i)=count;
        count=0;
    }
    cout<<"You currently own: "<<endl;
    for(int i=0;i<8;i++){
        cout<<*(colors+i)<<" "<<prop.colList(i)<<" contains ";
        cout<<prop.setcMax(prop.colList(i))<<" total properties"<<endl;
        if(*(colors+i)==prop.setcMax(prop.colList(i))){
            cout<<"Would you like to add houses here? The cost per house is $";
            cout<<prop.getHseC()<<endl;
            cin>>add;
            if(toupper(add)=='Y'){
                cout<<"How many would you like to purchase? ";
                cin>>numHse;
                nHouses+=numHse;
                cout<<"You now own "<<nHouses<<" houses. Due to this change, ";
                cout<<"the rent of all of your properties goes up. \n";
            }
        }
    }
    if(nHouses==0){
        cout<<"You can only purchase houses if you have obtained all ";
        cout<<"properties in at least one color group!"<<endl;
    }
}

void Player::setNHtl(Property &prop){
```

```
    int number;

    if(nHouses<4){
        cout<<"You can only purchase hotels if you have first purchased ";
        cout<<"4 houses!"<<endl;
    }
    else{
        cout<<"Hotels cost $"<<prop.getHtlC()<<". How many would you like? \n";
        cin>>number;
        nHotels+=number;
    }
}

void Player::payRent(int rent){
    money-=rent;
}

string Player::getName() const{
    return name;
}

int Player::getMony()const{
    return money;
}

unsigned short Player::getNPrp()const{
    return nProps;
}

void *Player::getPrps(){
    for(int i=0;i<nProps;i++)
        cout<<*(proprty+i)<<" ";
}

bool Player::findPrp(int num){
    for(int i=0;i<nProps;i++){
        if(num==*(proprty+i))
            return true;
    }
    return false;
}

int Player::getSpot() const{
    return spot;
}

int Player::getNHse()const{
    return nHouses;
}

int Player::getNHtl()const{
    return nHotels;
}

ostream &operator << (ostream &stream,const Player &right){
    Property prop;

    cout<<"\t\t\t\t\t\t\t";
    stream<<right.name<<endl;
    cout<<"\t\t\t\t\t\t\tMoney: $";
    cout<<fixed<<setprecision(2)<<showpoint;
    stream<<static_cast<float>(right.money)<<endl;
    cout<<"\t\t\t\t\t\t\tProperties:"<<endl;
```

```
    for(int i=0;i<right.nProps;i++){
        prop.inform(*(right.proprty+i),right.nHouses);
        cout<<"\t\t\t\t\t\t\t   ";
        stream<<prop.getname()<<endl;
    }

    return stream;
}

#include "Property.h"
#include "Board.h"

Property::Property(){
    rent=0;
    hseCost=0;
    htlCost=hseCost*5;
}

int Property::getrent() const{
    return rent;
}

int Property::getHseC()const{
    return hseCost;
}

int Property::getHtlC()const{
    return htlCost;
}

int Property::setcMax(string col){
    if(col=="purple"||col=="blue"||col=="Utility")
        maxColr=2;
    else if(col=="RR")
        maxColr=4;
    else
        maxColr=3;

    return maxColr;
}

void Property::inform(int title,int number){
    switch(title)
    {
        case GO:
            name="Go";
            price=0;
            color="";
            break;
        case MDTRRNN:
            name="Mediterranean Avenue";
            price=60;
            color="purple";
            if(number==0)rent=2;
            else if(number==1)rent=10;
            else if(number==2)rent=30;
            else if(number==3)rent=90;
            else if(number==4)rent=160;
            else rent=250;
            hseCost=50;
            break;
        case COMMCH1:
            name="Community Chest";
```

```
        price=0;
        color="";
        break;
    case BALTIC:
        name="Baltic Avenue";
        price=60;
        color="purple";
        if(number==0)rent=4;
        else if(number==1)rent=20;
        else if(number==2)rent=60;
        else if(number==3)rent=180;
        else if(number==4)rent=320;
        else rent=450;
        hseCost=50;
        break;
    case INCOME:
        name="Income Tax";
        price=200;
        color="tax";
        break;
    case READRR:
        name="Reading Railroad";
        price=200;
        color="RR";
        if(number==1)rent=25;
        else if(number==2)rent=50;
        else if(number==3)rent=100;
        else if(number>=4)rent=200;
        hseCost=0;
        break;
    case ORIENTL:
        name="Oriental Avenue";
        price=100;
        color="lilac";
        if(number==0)rent=6;
        else if(number==1)rent=30;
        else if(number==2)rent=90;
        else if(number==3)rent=270;
        else if(number==4)rent=400;
        else rent=550;
        hseCost=50;
        break;
    case CHANCE1:
        name="Chance";
        price=0;
        color="";
        break;
    case VERMONT:
        name="Vermont Avenue";
        price=100;
        color="lilac";
        if(number==0)rent=6;
        else if(number==1)rent=30;
        else if(number==2)rent=90;
        else if(number==3)rent=270;
        else if(number==4)rent=400;
        else rent=550;
        hseCost=50;
        break;
    case CONNECT:
        name="Connecticut Avenue";
        price=120;
        color="lilac";
```

```
        if(number==0)rent=8;
        else if(number==1)rent=40;
        else if(number==2)rent=100;
        else if(number==3)rent=300;
        else if(number==4)rent=450;
        else rent=600;
        hseCost=50;
        break;
    case JAIL:
        name="Jail";
        price=0;
        color="";
        break;
    case STCHRLS:
        name="St. Charles Avenue";
        price=140;
        color="pink";
        if(number==0)rent=10;
        else if(number==1)rent=50;
        else if(number==2)rent=150;
        else if(number==3)rent=450;
        else if(number==4)rent=625;
        else rent=750;
        hseCost=100;
        break;
    case ELECTRC: //FLAG!
        name="Electric Company";
        price=150;
        color="Utility";
        hseCost=0;
        break;
    case STATES:
        name="States Avenue";
        price=140;
        color="pink";
        if(number==0)rent=10;
        else if(number==1)rent=50;
        else if(number==2)rent=150;
        else if(number==3)rent=450;
        else if(number==4)rent=625;
        else rent=750;
        hseCost=100;
        break;
    case VIRGNIA:
        name="Virginia Avenue";
        price=160;
        color="pink";
        if(number==0)rent=12;
        else if(number==1)rent=60;
        else if(number==2)rent=180;
        else if(number==3)rent=500;
        else if(number==4)rent=700;
        else rent=900;
        hseCost=100;
        break;
    case PENNRR:
        name="Pennsylvania Railroad";
        price=200;
        color="RR";
        if(number==1)rent=25;
        else if(number==2)rent=50;
        else if(number==3)rent=100;
        else if(number>=4)rent=200;
```

```
        hseCost=0;
        break;
    case STJAMES:
        name="St. James Place";
        price=180;
        color="orange";
        if(number==0)rent=14;
        else if(number==1)rent=70;
        else if(number==2)rent=200;
        else if(number==3)rent=550;
        else if(number==4)rent=750;
        else rent=950;
        hseCost=100;
        break;
    case COMMCH2:
        name="Community Chest";
        price=0;
        color="";
        break;
    case TENNESE:
        name="Tennessee Avenue";
        price=180;
        color="orange";
        if(number==0)rent=14;
        else if(number==1)rent=70;
        else if(number==2)rent=200;
        else if(number==3)rent=550;
        else if(number==4)rent=750;
        else rent=950;
        hseCost=100;
        break;
    case NEWYORK:
        name="New York Avenue";
        price=200;
        color="orange";
        if(number==0)rent=16;
        else if(number==1)rent=80;
        else if(number==2)rent=220;
        else if(number==3)rent=600;
        else if(number==4)rent=800;
        else rent=1000;
        hseCost=100;
        break;
    case PARKING:
        name="Free Parking";
        price=0;
        color="";
        break;
    case KENTCKY:
        name="Kentucky Avenue";
        price=220;
        color="red";
        if(number==0)rent=18;
        else if(number==1)rent=90;
        else if(number==2)rent=250;
        else if(number==3)rent=700;
        else if(number==4)rent=875;
        else rent=1050;
        hseCost=150;
        break;
    case CHANCE2:
        name="Chance";
        price=0;
```

```
        color="";
        break;
    case INDIANA:
        name="Indiana Avenue";
        price=220;
        color="red";
        if(number==0)rent=18;
        else if(number==1)rent=90;
        else if(number==2)rent=250;
        else if(number==3)rent=700;
        else if(number==4)rent=875;
        else rent=1050;
        hseCost=150;
        break;
    case ILLNOIS:
        name="Illinois Avenue";
        price=240;
        color="red";
        if(number==0)rent=20;
        else if(number==1)rent=100;
        else if(number==2)rent=300;
        else if(number==3)rent=750;
        else if(number==4)rent=925;
        else rent=1100;
        hseCost=150;
        break;
    case BNORR:
        name="B & O Railroad";
        price=200;
        color="RR";
        if(number==1)rent=25;
        else if(number==2)rent=50;
        else if(number==3)rent=100;
        else if(number>=4)rent=200;
        hseCost=0;
        break;
    case ATLANTC:
        name="Atlantic Avenue";
        price=260;
        color="yellow";
        if(number==0)rent=22;
        else if(number==1)rent=110;
        else if(number==2)rent=330;
        else if(number==3)rent=800;
        else if(number==4)rent=975;
        else rent=1150;
        hseCost=150;
        break;
    case VENTNOR:
        name="Ventnor Avenue";
        price=260;
        color="yellow";
        if(number==0)rent=22;
        else if(number==1)rent=110;
        else if(number==2)rent=330;
        else if(number==3)rent=900;
        else if(number==4)rent=975;
        else rent=1150;
        hseCost=150;
        break;
    case WATERWK:
        name="Water Works";
        price=150;
```

```
        color="Utility";
        hseCost=0;
        break;
    case MARVIN:
        name="Marvin Gardens";
        price=280;
        color="yellow";
        if(number==0)rent=24;
        else if(number==1)rent=120;
        else if(number==2)rent=360;
        else if(number==3)rent=850;
        else if(number==4)rent=1025;
        else rent=1200;
        hseCost=150;
        break;
    case GO2JAIL:
        name="Go To Jail";
        price=0;
        color="";
        break;
    case PACIFIC:
        name="Pacific Avenue";
        price=300;
        color="green";
        if(number==0)rent=26;
        else if(number==1)rent=130;
        else if(number==2)rent=390;
        else if(number==3)rent=900;
        else if(number==4)rent=1100;
        else rent=1275;
        hseCost=200;
        break;
    case NCARLNA:
        name="North Carolina Avenue";
        price=300;
        color="green";
        if(number==0)rent=26;
        else if(number==1)rent=130;
        else if(number==2)rent=390;
        else if(number==3)rent=900;
        else if(number==4)rent=1100;
        else rent=1275;
        hseCost=200;
        break;
    case COMMCH3:
        name="Community Chest";
        price=0;
        color="";
        break;
    case PENNSYL:
        name="Pennsylvania Avenue";
        price=320;
        color="green";
        if(number==0)rent=28;
        else if(number==1)rent=150;
        else if(number==2)rent=450;
        else if(number==3)rent=1000;
        else if(number==4)rent=1200;
        else rent=1400;
        hseCost=200;
        break;
    case SHORTLN:
        name="Short Line";
```

```
            price=200;
            color="RR";
            if(number==1)rent=25;
            else if(number==2)rent=50;
            else if(number==3)rent=100;
            else if(number>=4)rent=200;
            hseCost=0;
            break;
        case CHANCE3:
            name="Chance";
            price=0;
            color="";
            break;
        case PARK:
            name="Park Place";
            price=350;
            color="blue";
            if(number==0)rent=35;
            else if(number==1)rent=175;
            else if(number==2)rent=500;
            else if(number==3)rent=1100;
            else if(number==4)rent=1300;
            else rent=1500;
            hseCost=200;
            break;
        case LUXTAX:
            name="Luxury Tax";
            price=75;
            color="tax";
            break;
        case BRDWALK:
            name="Boardwalk";
            price=400;
            color="blue";
            if(number==0)rent=50;
            else if(number==1)rent=200;
            else if(number==2)rent=600;
            else if(number==3)rent=1400;
            else if(number==4)rent=1700;
            else rent=2000;
            hseCost=200;
            break;
    }
}

#include <iostream>
#include <cctype>
using namespace std;

#include "Rules.h"
#include "Player.h"
#include "Die.h"

void Rules::Go2Jail(Player &player){
    Die die1,die2;
    char use,choice;
    int count=0;

    if(player.outJail>0){
        cout<<"Would you like to use your Get out of Jail Free card? (Y/N)";
        cin>>use;
        if(toupper(use)=='Y'){
            player.outJail=0;
```

```
            cout<<"Get Out of Jail Free Card used!"<<endl;
        }
        else{
            cout<<"Would you like to pay the $50 fine now? (Y/N) ";
            cin>>choice;
            if(toupper(choice)=='Y'){
                player.money-=50;
            }
            else{
                do{
                    die1.roll();
                    die2.roll();
                    cout<<"Die 1: "<<die1.getVal()<<"  Die 2: "<<die2.getVal()<<"\n";
                    count++;
                }while(die1.getVal()!=die2.getVal()&&count<3);
                if(count=3&&die1.getVal()!=die2.getVal())
                    player.money-=50;
            }
        }
    }
    else{
        cout<<"Would you like to pay the $50 fine now? (Y/N) ";
        cin>>choice;
        if(toupper(choice)=='N'){
            do{
                die1.roll();
                die2.roll();
                cout<<"Die 1: "<<die1.getVal()<<"  Die 2: "<<die2.getVal()<<"\n";
                count++;
            }while(die1.getVal()!=die2.getVal()&&count<3);
            if(count=3&&die1.getVal()!=die2.getVal()){
                player.money-=50;
                cout<<"Fine paid!"<<endl;
            }
        }
        else{
            player.money-=50;
            cout<<"Fine paid!"<<endl;
        }
    }
    player.setNew(JAIL);
}

void Rules::cGoJail(Player &comp){
    Die die1,die2;
    int count=0;

    if(comp.outJail>0)
        comp.outJail=0;
    else{
        do{
            die1.roll();
            die2.roll();
            cout<<"Die 1: "<<die1.getVal()<<"  Die 2: "<<die2.getVal()<<"\n";
            count++;
        }while(die1.getVal()!=die2.getVal()&&count<3);
        if(count=3&&die1.getVal()!=die2.getVal()){
            comp.money-=50;
            cout<<"Fine paid!"<<endl;
        }
    }
    comp.setNew(JAIL);
}
```

```
void Rules::restart(int num,Player &player){
    if((player.spot+num)>BRDWALK){
        leftovr=BRDWALK-player.spot;
        extra=(num-leftovr)-1;
        player.setNew(extra);
        player.money+=200;
        cout<<player.getName()<<" just passed GO, and collected $200."<<endl;
    }
    else
        player.setSpot(num);
}

void Rules::indxset(short &index){
    if(index>16)
        index=1;
}

bool Rules::gameEnd(Player &player){
    bool status=false;

    if(player.getMony()<0){
        status=true;
        cout<<"Oh no! "<<player.getName()<<" is in debt!"<<endl;
        cout<<"Sorry. Maybe next time you will make better decisions.";
        cout<<"The game is over!! We have a winner!"<<endl;
    }

    return status;
}

//System Libraries
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string>
#include <cctype>
#include <fstream>
using namespace std;

//User Libraries
#include "ChncCom.h"
#include "Die.h"
#include "Player.h"
#include "Property.h"
#include "Railroad.h"
#include "Rules.h"

//Function Prototypes
char opening(Player &,Player &);
void rdInstr();
int dieRoll();
void play(Player &,Player &,Property &,short &,short &);
void check(Player &,Player &,Property &,short &,short &);
void Menu(Property &,Player &);
void prcsOpt(int,Player &,Property &);
void automat(Player &,Player &,Property &,short &,short &);
void autochk(Player &,Player &,Property &,short &,short &);
int validte(int);

//Execution Begins Here!
int main(int argc, char** argv) {
    //Set the Random Number Seed
```

```
    srand(static_cast<unsigned int>(time(0)));

    //Instantiate Players, Property Object & Local Reference Variables
    Player user, comp;
    Property prop;
    Rules check;
    short chance=1,comchst=1;
    char again;

    //Begin to see who goes first
    char first;
    first=opening(user,comp);
    if(first=='c')
        automat(comp,user,prop,comchst,chance);
    cout<<"----------------------------------------"<<endl;

    //Begin Playing
    do{
        play(user,comp,prop,comchst,chance);
        automat(comp,user,prop,comchst,chance);
        cout<<"----------------------------------------"<<endl;
        cout<<"Would you like to continue playing the game? (Y/N)"<<endl;
        cin>>again;
        cout<<"----------------------------------------"<<endl;
    }while(toupper(again)=='Y'&&check.gameEnd(user)==false&&
            check.gameEnd(comp)==false);

    cout<<"Thank you for playing!!"<<endl;

    //Exit Stage Right!
    return 0;
}

void autochk(Player &c,Player &u,Property &prop,short &comindx,short &chnindx){
    ChncCom temp;
    Rules check;
    Railroad RR;

    if(prop.getname()=="Chance"){
        temp.setMess(1,chnindx,c,u,check);
        cout<<endl<<temp.getMess()<<endl;
        chnindx++;
        check.indxset(chnindx);
        if(c.getSpot()==30)
            check.cGoJail(c);
    }
    else if(prop.getname()=="Community Chest"){
        temp.setMess(2,comindx,c,u,check);
        cout<<endl<<temp.getMess()<<endl;
        comindx++;
        check.indxset(comindx);
        if(c.getSpot()==30)
            check.Go2Jail(c);
    }
    else if(prop.getname()=="Go To Jail")
        check.cGoJail(c);
    else if(prop.getcolr()=="")
        cout<<"Just Visiting!!"<<endl;
    else if(prop.getcolr()=="tax"){
        c.setMony(c.getMony()-prop.getprce());
        cout<<c.getName()<<" just paid "<<prop.getname()<<" of $";
        cout<<prop.getprce()<<endl;
    }
```

```
    else{
        if(u.findPrp(c.getSpot())==true){
            if(prop.getcolr()=="RR"){
                c.payRent(RR.setrent(u,c.getSpot()));
                cout<<endl;
                cout<<"You already own "<<prop.getPos(c.getSpot());
                cout<<" which is the "<<RR.getPos(c.getSpot())<<" railroad. ";
                cout<<c.getName()<<" pays you $"<<RR.setrent(u,c.getSpot());
                cout<<" of rent."<<endl;
                u.setMony(u.getMony()+RR.setrent(u,c.getSpot()));
            }
            else if(prop.getcolr()=="Utility"){
                int number;
                int total=dieRoll();
                if(u.findPrp(WATERWK)^u.findPrp(ELECTRC)==true)
                    number=1;
                else number=2;
                prop.utilRnt(number,total);
                c.payRent(prop.getrent());
                cout<<c.getName()<<"'s new roll is "<<total<<endl;
                cout<<"You already own "<<prop.getname()<<". "<<c.getName();
                cout<<" pays you $"<<prop.getrent()<<" of rent."<<endl;
                u.setMony(u.getMony()+prop.getrent());
            }
            else{
                c.payRent(prop.getrent());
                cout<<endl;
                cout<<"You already own "<<prop.getname()<<". "<<c.getName();
                cout<<" pays you $"<<prop.getrent()<<" of rent."<<endl;
                u.setMony(u.getMony()+prop.getrent());
            }
        }
        else if(c.findPrp(c.getSpot())==true)
            cout<<c.getName()<<" already owns this property."<<endl;
        else{
            int total=c.getMony()-prop.getprce();
            c.setMony(total);
            cout<<" and bought it for $"<<prop.getprce()<<endl;
            c.setPrps();
        }
    }
    cout<<c;
}

void automat(Player &c,Player &u,Property &prop,short &comindx,short &chnindx){
    Rules check;

    check.restart(dieRoll(),c);
    prop.inform(c.getSpot(),u.getNHse());
    cout<<"Your opponent landed on "<<prop.getname()<<endl;

    autochk(c,u,prop,comindx,chnindx);
}

void prcsOpt(int decide,Player &u,Property &prop){
    char endturn;

    //Process the Menu Decision
    if(decide==1){
        if(u.findPrp(u.getSpot())==true)
            cout<<"You already own this property "<<u.getName()<<endl;
        else{
            cout<<"$"<<prop.getprce()<<" was deducted from your account."<<endl;
```

```
            int total=u.getMony()-prop.getprce();
            u.setMony(total);
            u.setPrps();
            cout<<"Congratulations on your purchase!"<<endl;
        }
    else if(decide==2)
        cout<<"Thanks for the visit!"<<endl;
    else if(decide==3){
        u.setNHse();
    }
    else if(decide==4){
        u.setNHtl(prop);
    }
    cout<<endl;

    cout<<"Would you like to end turn? (Y/N) ";
    cin>>endturn;
    if(toupper(endturn)=='N')
        Menu(prop,u);
}

int validte(int option){
    cin>>option;          //Overload the >> operator to display player info
    if(option<1||option>4){
        string except = "ERROR: That is not a valid entry!\n";
        throw except;
    }
    else
        return option;
}

void Menu(Property &prop,Player &u){
    int choice;

    //In-Game Menu
    cout<<"What would you like to do?"<<endl;
    cout<<"1. Purchase "<<prop.getname()<<" for $"<<prop.getprce()<<"."<<endl;
    cout<<"2. Just Visit the Property."<<endl;
    cout<<"3. Build a House."<<endl;
    cout<<"4. Build a Hotel."<<endl;

    try{
        choice=validte(choice);
        prcsOpt(choice,u,prop);
    }
    catch(string except){
        cout<<except;
        cout<<"Please enter a valid Menu option. ";
        cin>>choice;
    }
}

int dieRoll(){
    //Instantiate 2 Dice
    Die die1,die2;

    //Roll the dice
    die1.roll();
    die2.roll();

    cout<<"Rolling..."<<endl;
    cout<<"Die 1: "<<die1.getVal()<<" Die 2: "<<die2.getVal()<<endl;
```

```
    return die1.getVal()+die2.getVal();
}

void check(Player &u,Player &c,Property &spot,short &comindx,short &chnindx){
    Rules check;
    ChncCom temp;
    Railroad RR;

    //Check for Special "Properties"
    if(spot.getname()=="Chance"){
        temp.setMess(1,chnindx,u,c,check);
        cout<<endl<<temp.getMess()<<endl;
        chnindx++;
        check.indxset(chnindx);
        if(u.getSpot()==30)
            check.Go2Jail(u);
    }
    else if(spot.getname()=="Community Chest"){
        temp.setMess(2,comindx,u,c,check);
        cout<<endl<<temp.getMess()<<endl;
        comindx++;
        check.indxset(comindx);
        if(u.getSpot()==30)
            check.Go2Jail(u);
    }
    else if(spot.getname()=="Go To Jail"){
        check.Go2Jail(u);
    }
    else if(spot.getcolr()=="")
        cout<<"Just Visiting!!"<<endl;
    else if(spot.getcolr()=="tax"){
        u.setMony(u.getMony()-spot.getprce());
        cout<<"You just paid "<<spot.getname()<<" of $"<<spot.getprce()<<endl;
    }
    else{
        if(c.findPrp(u.getSpot())==true){
            if(spot.getcolr()=="RR"){
                u.payRent(RR.setrent(c,u.getSpot()));
                cout<<endl;
                cout<<c.getName()<<" already owns "<<spot.getPos(u.getSpot());
                cout<<" which is the "<<RR.getPos(u.getSpot())<<" railroad. ";
                cout<<"You owe them $"<<RR.setrent(c,u.getSpot());
                cout<<" of rent."<<endl;
                c.setMony(c.getMony()+RR.setrent(c,u.getSpot()));
            }
            else if(spot.getcolr()=="Utility"){
                int number;
                int total=dieRoll();
                if(c.findPrp(WATERWK)^c.findPrp(ELECTRC)==true)
                    number=1;
                else number=2;
                spot.utilRnt(number,total);
                u.payRent(spot.getrent());
                cout<<c.getName()<<" already owns "<<spot.getname()<<". ";
                cout<<"You owe them $"<<spot.getrent()<<" of rent."<<endl;
                c.setMony(c.getMony()+spot.getrent());
            }
            else{
                u.payRent(spot.getrent());
                cout<<endl;
                cout<<c.getName()<<" already owns "<<spot.getname()<<". ";
                cout<<"You owe them $"<<spot.getrent()<<" of rent."<<endl;
                c.setMony(c.getMony()+spot.getrent());
```

```
            }
        }
        else if(u.findPrp(u.getSpot())==true)
            cout<<"You already own this property "<<u.getName()<<endl;
        else{
            Menu(spot,u);
        }
    }
    cout<<u;
}

void play(Player &u,Player &c,Property &spot,short &comindx,short &chnindx){
    //Declare Menu Choice Variable
    Rules test;

    test.restart(dieRoll(),u);
    spot.inform(u.getSpot(),c.getNHse());
    cout<<"You landed on "<<spot.getname()<<endl;

    //Check for Special "Properties"
    check(u,c,spot,comindx,chnindx);
}

void rdInstr(){
    fstream instrct;
    string lines;

    //Open the file
    instrct.open("Instructions.txt",ios::in);

    if(instrct){
        getline(instrct,lines);
        while(instrct){
            cout<<lines<<endl;
            getline(instrct,lines);
        }
        cout<<endl;

        //Close the file
        instrct.close();
    }
    else
        cout<<"Error. Cannot open file"<<endl;
}

char opening(Player &u,Player &c){
    //Declare Comparison Variables
    int comp,user;
    char seeIns;

    //Get & Set Names
    cout<<"Hello! Welcome to the wonderful game of Monopoly!"<<endl;
    cout<<"Before we begin, please enter your name. ";
    u.setName();
    cout<<"And also enter the name of your opponent. ";
    c.setName();

    cout<<"Do you need to read the instructions? (Y/N)"<<endl;
    cin>>seeIns;
    if(toupper(seeIns)=='Y'){
        rdInstr();
        cout<<endl;
```

```
    }
    cout<<"We will begin the game by rolling the dice to see who gets to go ";
    cout<<"first!"<<endl<<endl;
    cout<<"Let's start with "<<c.getName()<<endl;

    do{
        comp=dieRoll();
        cout<<c.getName()<<" rolls "<<comp<<endl<<endl;
        cout<<"Now it's your turn "<<u.getName()<<endl;
        user=dieRoll();
        cout<<"You roll "<<user<<endl<<endl;

        //Compare to see who goes first
        if(user>comp){
            cout<<"You get to go first!"<<endl;
            return 'u';
        }
        else if(user<comp){
            cout<<c.getName()<<" goes first."<<endl;
            return 'c';
        }
        else cout<<"It's a Tie! Let's try this again."<<endl;
    }while(comp==user);
}
```