ECE 373 – Team 8 Project Proposal

Abstract

For this project, we have decided to make Monopoly board game. This game will follow exactly same rule as real world Monopoly board game by utilizing object oriented programming concept.

Monopoly Game Description

Monopoly Game Description

The game is playable by up to 4 players. Each Player rolls two dice on their turn and moves forward on the board clockwise a number of spaces equal to the sum of the dice. If the dice each land on the same number, (i.e. the Player rolls doubles) the Player rolls the dice one more time and moves again a number of spaces equal to the sum of the dice. The Player can roll doubles up to 3 times in one turn, in which case the player must go to jail.

The Player can own properties. Players who land on property owned by another player has to pay rent. If the property they land on is not owned then the Player can buy the property for its listed cost. Each Property can be owned by only one player. Each player owns $1,500 at start. If the Player runs out of cash, the Player has to sell properties and/or buildings to make cash. When the Player sells property and buildings to the Bank, they're sold at half their buying price. If the Player runs out of cash and properties, the Player is bankrupted and out of the game. Each Player gets the same money every time they make a full revolution on the game board.

Properties have name, color, price, and rent. Properties can have up to 4 houses or 1 hotel. Properties have a building cost to build houses or hotels. Rent depends on the # of houses or hotel built on the Property. To build houses or a hotel on the Property the Player must own all other properties with the same color, i.e. have a monopoly of that color. Buildings have to be built evenly (no hotel on one property and three or less houses on the other properties in the monopoly). Properties can be sold to the Bank or other players. As mentioned, when sold to the Bank properties and buildings must be sold for half the buying price, i.e. their mortgage value, whereas when they're sold to other players they can be sold for whatever price is agreed upon by the players involved.

If the Player owns one utility, other players who land on the utility pay the owner an amount equal to 4 times the sum of the dice that they rolled. If player owns two utilities, multiplier is 10 instead of 4. Players cannot build houses or hotels on utilities. There are two utility spaces on the game board.

If the Player owns one railroad rent is $25. If two, the rent is $50. If three, the rent is $100, and if four, the rent is $200. Players cannot build houses or hotels on railroads. There are four railroad spaces on the game board.

All the money belongs to either the Bank or the players. The Bank owns all properties and 32 houses and 12 hotels at the game start. The Bank never runs out of money. The Bank can give money to players.

If the Player lands on Go to Jail or draws the chance card saying go to jail, the Player goes to jail and stays there for 3 turns unless the Player rolls doubles. If the Player just lands on jail, player is just visiting. When the Player stays in jail for 3 turns, player has to pay $75 to get out of the jail. If the Player wants to get out of the jail earlier, the Player has to pay $150.

If the Player lands on a Chance cell, the Player draws a Chance card from the top of the Chance deck. Some Chance cards, the "Get out of jail free" cards, can be owned by only one player. All other Chance cards are placed on the bottom of the Chance card deck after the Player immediately obeys them. Chance cards have a command which the Player has to follow. Some examples of chance cards are; Get out of jail, Go to jail, Go to the most expensive city, Go 3 steps forward, Go back 2 steps, Pay $75 to each other Player, Collect $75 from each other Player.

If the Player lands on Community Chest, the Player draws a Community Chest card from the top of the deck. Community Chest cards have a command which the Player has to follow just like the Chance cards. Donated money from Community Chest cards goes to Free Parking. Some examples of Community Chest cards are; Donate $50, Donate $75, Donate $25, Win the prize get $100 from bank.

Free Parking collects the money donated from Community Chest donations. Whoever lands on Free Parking gets all the collected money. If the Player landed on Free Parking and if there is no money accumulated, player doesn't get any money.

There are two tax spaces; Income tax and Luxury tax. If the Player lands on Income tax cell, player pays $200 to the Bank for tax. Else, Luxury tax is $100, paid to the Bank.

UML

There's no superclass or subclasses between any classes. Interface class has not been developed yet. Interface or the driver class will dominate all the other classes and will be calling different classes to make the game work. Rolling the dice will be done in driver or other interface class with just simple random () method.

Bank
- propertyList:ArrayList<Property>
- house :int
- hotel : int
+Bank()
+setPropertyList(prop:Property):void
+addPropertyToList(prop:Property):void
+removefromPropertyList(prop:Property):void
+setHouse(i:int):void
+getHouse():int
+setHotel(i:int):void
+getHotel():int

Property
- name :String
- color : String
- price : int
- rent : int
- house: int
- hotel:int
- owned:boolean
+Property()
+setName(str:String):void
+getName():String
+setColor(str:String):void
+setPrice(i:int):void
+getPrice():int
+setRent(i:int):void
+setNumberOfHouse(i:int):void
+addHouse(i:int):void
+removeHouse(i:int):void
+getNumberOfHouse():int
+setNumberOfHotel(i:int):void
+addHotel(i:int):void
+removeHotel(i:int):void
+getNumberOfHotel():int
+setOwned(own:boolean):void
+getOwned():boolean

Player
- name : String
- propertiesList : ArrayList<Property>
- utilitiesList : ArrayList<Utility>
- railroadList : ArrayList<Railroad>
- chaceCardList : ArrayList<ChanceCard>
- jail:boolean
- locatioin:int
- money: int
- doubleCount:int
- Bankrupted:boolean
+Player()
+setName(str:String):void
+getName():String
+setProperty(prop:Property):void
+addProperty(prop:Property):void
+takeProperty(prop:Property):void
+getProperty():ArrayList<Property>
+setUtility(prop: Utility):void
+addUtility (prop: Utility):void
+takeUtility (prop: Utility):void
+getUtility ():ArrayList<Utility>
+setRailRoad(prop: RailRoad):void
+addRailRoad (prop: RailRoad):void
+takeRailRoad (prop: RailRoad):void
+getRailRoad ():ArrayList<RailRoad>
+setChanceCard(prop: ChanceCard):void
+addChanceCard (prop: ChanceCard):void
+takeChanceCard (prop: ChanceCard):void
+getChanceCard ():ArrayList<ChanceCard>
+setJail(jail:boolean): void
+getJail():boolean
+setLocation(i:int):void
+moveLocation(i:int):void
+getLocation():int
+setMoney(i:int):void
+addMoney(i:int):void
+takeMoney(i:int):void
+getMoney():int
+setDouble(i:int):void

```
+addDouble(i:int):void
+getDouble():void
+setBankrupted(br:boolean):void
+getBankrupted():boolean
```

```
ChanceCard
- command : String
- owned : boolean
- ownable : boolean
+ChanceCard()
+setCommand(str:String):void
+getCommand():String
+setOwned(b:boolean):void
+getOwned():boolean
+setOwnable(b:boolean):void
+getOwnable():boolean
```

```
FreeParking
- money : int
+FreeParking()
+setMoney(i:int):void
+getMoney():void
```

```
Utility
- name : String
- multiplier : int
- rent : int
- owned:boolean
+Utility
+setName(str:String):void
+getName():String
+setMultiplier(i:int):void
+getMultiplier():int
+setRent(multiplier:int, dice:int):void
+getRent():int
+setOwned(own:boolean):void
+getOwned():boolean
```

```
Jail
- turns : int
- payment : int
- payed: boolean
+Jail()
+setTurns(i:int):void
+TurnOver():void
+getTurns():int
+setPayment(i:int):void
+getPayment():int
+IsPayed():boolean
```

```
RailRoad
- name:String
- rent:int
- owned: boolean
- count:int
+RailRoad()
+setName(str:String):void
+getName():String
+setCount(i:int):void
+getCount():int
```

```
CommunityChest
- command : String
+CommunityChest()
+setCommand(str:String):void
+getCommand():String
```

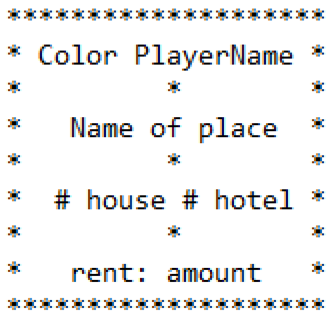
```
Tax
-payment:int
-type:String
+Tax()
+setType(str:String):void
+getType():String
+setPayment(i:int,type:String):void
```

```
+setRent(count:int):void
+getRent():int
+setOwned(own:boolean):void
+getOwned():boolean
```

```
+getPayment(type:String):int
```

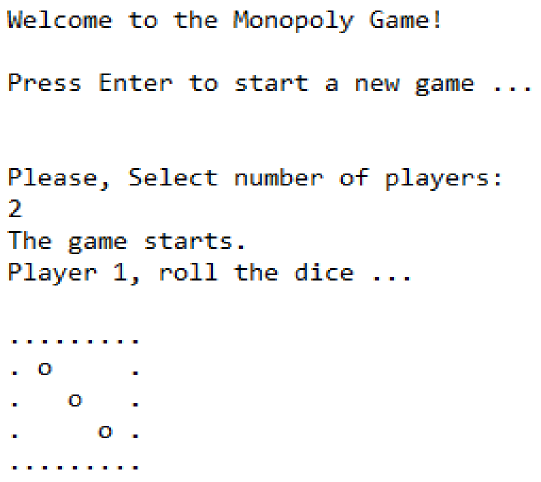
User interface

The user interaction with a game is done through a console. The user input will be scanned from the keyboard. After the player rolls the dice, the board is displayed on the screen, so that the user can keep track of his status in the game. During the game user may see the following information: current property and money user and the other players own, houses and hotels available for sale. This data is brought to a screen when the user enters the specific command. This is a sample view of a card.



```
********************
* Color PlayerName *
*         *        *
*   Name of place  *
*         *        *
*  # house # hotel *
*         *        *
*   rent: amount   *
********************
```

The picture below is a sample representation of the game process:

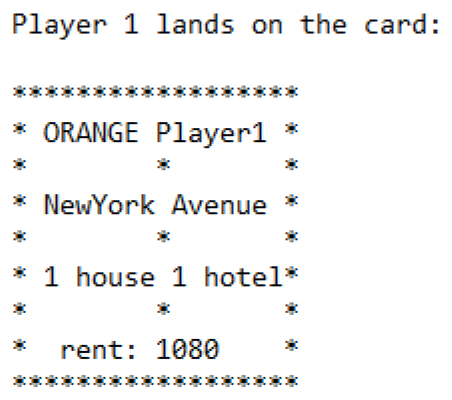


```
Welcome to the Monopoly Game!

Press Enter to start a new game ...


Please, Select number of players:
2
The game starts.
Player 1, roll the dice ...

.........
. o     .
.   o   .
.     o .
.........
```



```
Player 1 lands on the card:

******************
* ORANGE Player1 *
*        *       *
* NewYork Avenue *
*        *       *
* 1 house 1 hotel*
*        *       *
*  rent: 1080    *
******************
```

Division
For the project, there are classes to make, and interface to make. So, we divided our work as following way.

Poe Park - Properties, Utility, and all other remaining classes

Joshua Edward Fijal - Bank, user interface

Aigerim Mukusheva – player class.