Implementation and Design

1. Design Choice

Inheritance:

- AIPlayer extends Player: Because AIPlayer should have behaviors such as buy, sell property and pay tax/rent which have been implemented in the Player class
- PrivateProperty and BankProperty extend Square: Each Square have the name, index of itself in the board, also methods for parsing XML such as toString(), toXML(), readFile()
- Business and Rail extend PrivateProperty because they share the same behaviors such as getOwner(), parsing XML methods such as toString(), toXML(), readFile()

Interface:

- RentableAPI is an interface class which is implemented by Rail and Business. Rail and Business must have the getRentAmount(), getSalePrice() and sell() to do their function.
- PropertyAPI is an interface class which is implemented by BankProperty, Rail and Business. They are all shared the same collectMoney() method
- RoleAPI is implemented by Bank and Player class. Each model share the same methods such as addMoney(), removeMoney()
- ActionListener is implemented by MonopolyController to perform the functionality of the model classes based on the event of the view class

MVC:

- MonopolyController will receive the event when the button in the MonopolyView is trigger and then the controller will tell the MonopolyModel. The MonopolyModel will perform the event based on the component of the view and tell specific model to handle the event. After that it will update the view.

2. XML format instruction

<u>Sample format:</u>

```
<Model>



<Board>
        <Square>name-index</Square>
        <Business>name-index-price-numHouse-numHotel</Business>
        <Rails>name-index-price</Rails>
        <BankProperty>name-index-taxValue</BankProperty>
    </Board>

    <Players>

        <Player>name-playerBalance-inJail-turnsInJail-turnsPlayed-currLocation-
        propertyList</Player>
        <AIPlayer>name-playerBalance-inJail-turnsInJail-turnsPlayed-
        currLocation-propertyList</AIPlayer>
    </Players>

    <currentPlayer>playerIndex</currentPlayer>
    <consecutiveDoubles>#doubles</consecutiveDoubles>

</Model>
```

<u>Tags:</u>

Square: name-index

index is the location of its in the board

Business: name-index-price-numHouse-numHotel

Rail: name-index-price

BankProperty: name-index-taxValue

Player: name-playerBalance-inJail-turnsInJail-turnsPlayed-currLocation-propertyList

currLocation: in the XML should be an integer number < 38 which represents the index of the square on the board. When the program parse the XML, it will get the Square object based on that index.

propertyList:

For example: squareIndex1#squareIndex2

The property that the player owns will be made by the propertyListToString() which will separate each index of the Square by #

When parsing the Player tag, the readFile() will create an Player by using the split method to deal with this format:

name-playerBalance-inJail-turnsInJail-turnsPlayed-currLocation

After that, readFile() will deal with the propertyList by split the # since the format of propertyList is

squareIndex1#squareIndex2

It will get the PrivateProperty based on the index and add that property to the playerPropertyList