# Programming Assignment 3

**Submission Date :** 18.04.2019
**Due Date :** 09.05.2019 (23:59)
**Subject :** Inheritence and Polimorphism
**Advisors :** R.A. Bahar GEZİCİ

## 1 Introduction

In this experiment, you have been commissioned to develop a non graphical Java version of the classic Monopoly game. You are expected to implement monopoly with the given rule. Main focus point of this experiment is to get you familiar with inheritence and polymorphism.

**Inheritance** is one of the core concepts of object-oriented programming (OOP) languages. It is a mechanism where you can to derive a class from another class for a hierarchy of classes that share a set of attributes and methods.

**Polymorphism** is the ability of an object to take on many forms. The most common use of polymorphism in object oriented design occurs when a parent class reference is used to refer to a child class object.

It is expected you to implement the program by using inheritence and polimorphism. Implementing the program without using the benefits of these concepts will be **penalized.**


Figure 1: Monopoly game

## 2 Problem

In this experiment you are expected to implement the monopoly game. The game consists of a sequential set of 40 squares containing 28 properties, 6 action squares (3 ”chance”, 3 ”community chest”), 2 tax squares, ”go”, ”jail”, ”free parking”, and ”go to jail”. Every time a player lands on a square, she takes a different action if this square is one of the properties.

For the other squares, the player takes a different action depend on the type of the square. For example, if she lands on a "tax square", she must pay a certain amount of money to the banker, but if she lands on an "action square" she must draw a card from the appropriate pile and she must follow the instructions on the card.

In this game, properties are classified as Land, Company and Rail Roads. Users are classified as Player and Banker and they have name and money attributes. Your task is to design the game with the given rules by using inheritence and polymorphism. You must use these concepts while designing your program.

You will be given two JSON formatted input files:
a. Properties file

b. Cards file

JSON is a format for storing data in the files for users (see Appendix A). You are supposed to use JSON.simple library, which is a simple Java library for reading and writing JSON formatted files. (with full compliance with JSON specification (RFC4627)).

**a. Properties file**

The properties are taken from a property file. Land, Railroads, Company are numerated as 1,2,3 respectively. This file includes information for all the properties in the game. The following information is provided for each property: property id (the position of this property on board), name of the property, the cost of the property. A sample property file is given as follows:

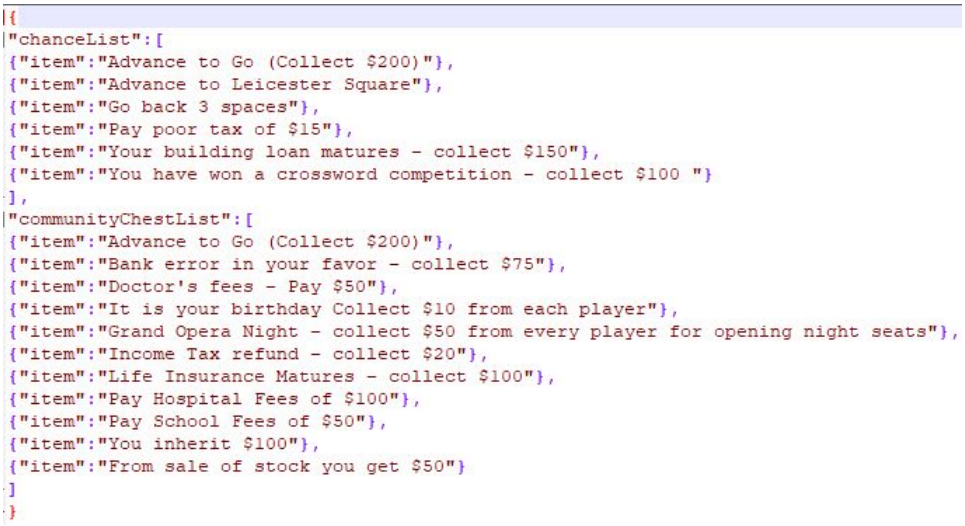

```
{
    "1": [
        {"id": "2","name":"Old Kent Road","cost": "600"},
        {"id": "4","name":"Whitechapel Road", "cost": "600"},
        {"id": "7","name":"The Angel Islington","cost": "1000"},
        {"id": "9","name":"Euston Road", "cost": "1000"},
        {"id": "10","name":"Pentonville Road","cost": "1200"},
        {"id": "12","name":"Pall Mall","cost": "1400"},
        {"id": "14","name":"Whitehall","cost": "1400"},
        {"id": "15","name":"Northumberland Avenue","cost": "1600"},
        {"id": "17","name":"Bow Street","cost": "1800"},
        {"id": "19","name":"Marlborough Street","cost": "1800"},
        {"id": "20","name":"Vine Street","cost": "2000"},
        {"id": "22","name":"The Strand","cost": "2200"},
        {"id": "24","name":"Fleet Street","cost": "2200"},
        {"id": "25","name":"Trafalgar Square", "cost": "2400"},
        {"id": "27","name":"Leicester Square","cost": "2600"},
        {"id": "28","name":"Coventry Street","cost": "2600"},
        {"id": "30","name":"Piccadilly","cost": "2800"},
        {"id": "32","name":"Regent Street","cost": "3000"},
        {"id": "33","name":"Oxford Street","cost": "3000"},
        {"id": "35","name":"Bond Street","cost": "3200"},
        {"id": "38","name":"Park Lane","cost": "3500"},
        {"id": "40","name":"Mayfair","cost": "4000"}
    ],
    "2": [
        {"id": "6","name":"Kings Cross Station","cost": "2000"},
        {"id": "16","name":"Marylebone Station","cost": "2000"},
        {"id": "26","name":"Fenchurch St Station","cost": "2000"},
        { "id": "36","name":"Liverpool Street Station","cost": "2000"}
    ],

    "3": [
        {"id": "13","name":"Electric Company", "cost": "1500"},
        {"id": "29","name":"Water Works","cost": "1500"}
        ]
}
```


Figure 2: A Sample property file

### b. Cards file

This file involves all the information about the cards in the game, which are divided into Chance List and Community Chest List. A sample cards file is given below:



```
{
"chanceList":[
{"item":"Advance to Go (Collect $200)"},
{"item":"Advance to Leicester Square"},
{"item":"Go back 3 spaces"},
{"item":"Pay poor tax of $15"},
{"item":"Your building loan matures - collect $150"},
{"item":"You have won a crossword competition - collect $100 "}
],
"communityChestList":[
{"item":"Advance to Go (Collect $200)"},
{"item":"Bank error in your favor - collect $75"},
{"item":"Doctor's fees - Pay $50"},
{"item":"It is your birthday Collect $10 from each player"},
{"item":"Grand Opera Night - collect $50 from every player for opening night seats"},
{"item":"Income Tax refund - collect $20"},
{"item":"Life Insurance Matures - collect $100"},
{"item":"Pay Hospital Fees of $100"},
{"item":"Pay School Fees of $50"},
{"item":"You inherit $100"},
{"item":"From sale of stock you get $50"}
]
}
```

Figure 3: A Sample cards file

### Commands

This file includes the commands in the game. Each command consists of a player id of the player to be moved, which is followed by an integer that is the number on the rolled dice. Id and the number are delimited by semicolons:

```
[Player name]; [dice]
[Player name]; [dice]
[Player name]; [dice]
show()
.....
```

The game flows as follows: You will read the properties file to have a list of properties in the game. You will read the cards file to have a list of cards to draw in any action square (chance vs community chest). Your players will take the actions written in the commands file and you will write the results of the commands in an output file. In the commands, there will be also show command. You will write the players properties, money and bankers money in an output file whenever you come across show command. At the end, the result of the game will be written in the same output file as if a show command is met.

After reading the input files, your program will start reading the commands file line by line and execute the moves of the players. The program will exit when it reaches the end of the commands file or any players money is zero or under zero.

```
Player 1;12
Player 2;12
show()
Player 1;4
Player 2;4
show()
Player 1;6
Player 2;6
show()
Player 1;5
Player 2;4
Player 1;8
Player 2;9
show()
Player 1;7
Player 2;6
show()
```

Figure 4: An example of command file

# 3 General Rules of the Game:

- There are only two players and a banker in the game. They have attributes like name and money. Players have got 15000 TL and banker has got 100000 TL at the beginning of the game.
- Player moves to new square by rolling a dice. When she lands on a "property" square if any player does not own the property, player will buy this property if she can afford to buy the property.
- If any player owns the property where the current player landed on, the current player must pay for the rent of the property.
- If the property is a land, the rent is calculated based on the cost of the land (for 0-2000 rent is 40%, for 2001-3000 30%, for 3001-4000 35% of the cost). If the property is a company, the rent amount is 4xdice. If the property is a railroad, the rent of the property is 25*(the number of the railroads other players has) TL.
- If the player lands on an "action" square, she draws a card and plays depending on the instructions on the card.
- Cards are selected sequentially from the given card list.
- If the player lands on a "jail" square, she must wait for 3 times without playing her move.
- If the player lands on a "tax square", she must pay for 100 TL to the banker.
- If the player lands on a "free parking" square, she waits for a tour without playing.
- If the player lands on a "go to jail" square, she must go to the jail square.
- If the player passes by a "go" square while "go to jail" , she can't take money for passing a go square.
- If the player lands on or passes by a "go" square, she takes 200 TL from the banker.

- The game will be finished when all the commands in the command file are played or any player goes bankrupt.
- The game board is circle, the go square (the 1st square) follows the 40th square on the board.

Here, the order of squares is given to help you. Each color represents 40 squares that is containing 28 properties, 6 action squares (3 "chance", 3 "community chest"), 2 tax squares, "go", "jail", "free parking", and "go to jail".

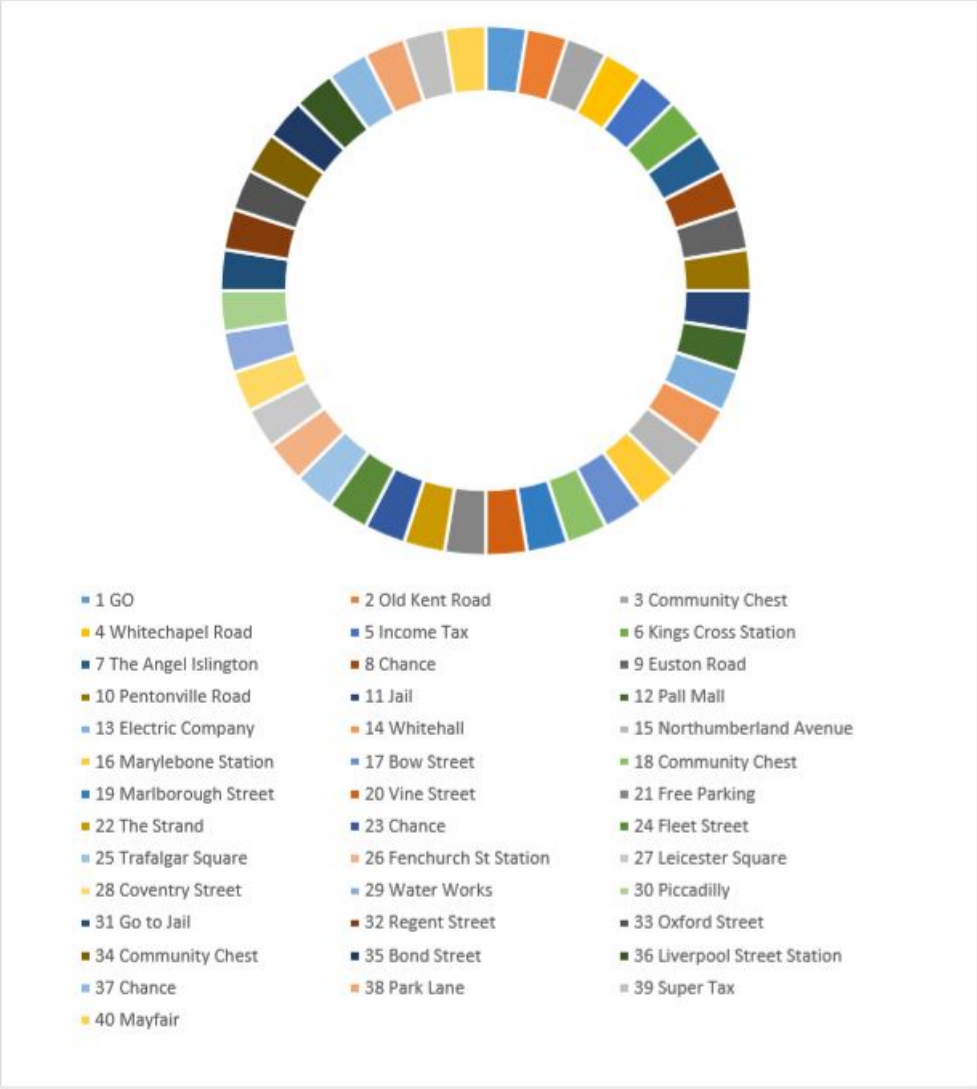



Figure 5: Order of squares

# 4 Output:

Once all the lines in the command file are performed, your program will create and output the current status of the players on the board and players and banker's money as shown in the figure given below: The order of each column is given sequentially :
[**Player**]tab[**Dice**]tab[**New Position**]tab[**Player1 Money**]tab[**Player2 Money**]tab[**Processing**]

```
Player 1    12  13  13500   15000   Player 1 bought Electric Company
Player 2    12  13  13548   14952   Player 2 paid rent for Electric Company
-----------------------------------------------------------------------------------------------------------
Player 1    13548   have: Electric Company
Player 2    14952   have:
Banker  101500
Winner Player 2
-----------------------------------------------------------------------------------------------------------
Player 1    4   17  11748   14952   Player 1 bought Bow Street
Player 2    4   17  12468   14232   Player 2 paid rent for Bow Street
-----------------------------------------------------------------------------------------------------------
Player 1    12468   have: Bow Street,Electric Company
Player 2    14232   have:
Banker  103300
Winner Player 2
-----------------------------------------------------------------------------------------------------------
Player 1    6   1   12668   14232   Player 1 draw Advance to Go (Collect $200)
Player 2    6   27  12668   11632   Player 2 draw Advance to Leicester Square Player 2 bought Leicester Squ
-----------------------------------------------------------------------------------------------------------
Player 1    12668   have: Bow Street,Electric Company
Player 2    11632   have: Leicester Square
Banker  105700
Winner Player 1
-----------------------------------------------------------------------------------------------------------
Player 1    5   6   10668   11632   Player 1 bought Kings Cross Station
Player 2    4   11  10668   11632   Player 2 went to jail
Player 1    8   14  9268    11632   Player 1 bought Whitehall
Player 2    9   11  9268    11632   Player 2 in jail (count=1)
-----------------------------------------------------------------------------------------------------------
Player 1    9268    have: Whitehall,Bow Street,Kings Cross Station,Electric Company
Player 2    11632   have: Leicester Square
Banker  109100
Winner Player 2
-----------------------------------------------------------------------------------------------------------
Player 1    7   21  9268    11632   Player 1 is in Free Parking
Player 2    6   11  9268    11632   Player 2 in jail (count=2)
-----------------------------------------------------------------------------------------------------------
Player 1    9268    have: Whitehall,Bow Street,Kings Cross Station,Electric Company
Player 2    11632   have: Leicester Square
Banker  109100
Winner Player 2
-----------------------------------------------------------------------------------------------------------
Player 1    9268    have: Whitehall,Bow Street,Kings Cross Station,Electric Company
Player 2    11632   have: Leicester Square
Banker  109100
Winner Player 2
```

Figure 6: Sample output of program

**Submit Format** - File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system)

- ⟨*studentid*⟩*.zip*

    - src.zip (Main.java, *.java)

    - Report.pdf

# 5 Notes

- The assignment must be original, individual work. Downloaded or modified source codes will be considered as cheating. Also the students who share their works will be punished in the same way.
- We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism. Our department takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned.
- You can ask your questions through courses piazza group and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes and reports .
- With this assignment which covers the subjects of classes, objects, polimorphism, and inheritance; you are expected to gain practice on the basics of object oriented programming (OOP). Therefore, you will not be graded if any paradigm other than OOP (i.e., structured programming) is developed!
- You are responsible for a correct model design. Your design should be accurate. It is important to **draw a class diagram** to show the whole of the system that you have created.
- **In your REPORT,** it is important giving explanation about problem definition (limited to max. 3 sentences) and steps of algorithms that you followed (limited to max. 5 sentences). Also, you must add class diagram of your code in the report. Do not give so much unnecessary details in your report (totally max 2 pages written with Times New Roman 12).
- Don't forget to write comments of your codes when necessary.
- The names of classes, attributes and methods should obey to Java naming convention.
- Save all work until the assignment is graded.
- Do not miss the deadline. Submission will be end at 09/05/2019 23:59, the system will be open 23:59:59. The problem about submission after 23:59 will not be considered.
- There will be no extension in this project. So, please do not wait and ask for any extension.

**Appendix A.**

JSON (JavaScript Object Notation) is an open-standard format that uses human-readable text to transmit data objects consisting of attributevalue pairs. It is the most common data format used for asynchronous browser/server communication, largely replacing XML which is used by AJAX. There are several APIs for writing or reading JSON formatted objects. [1]

**JSON Example**

```
{"employees":[
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter", "lastName":"Jones"}
]}
```

Figure 7: JSON Example

# 6 References

1. *http://www.mkyong.com/java/json-simple-example-read-and-write-json/*