# Hacettepe University
# Department of Computer Engineering
# BBM104 Introduction to Programming Laboratory II
# Programming Assignment III

| | |
|---|---|
| **Submission Date** | **:07.04.2016** |
| **Due Date** | **:20.04.2016** |
| **Advisors** | **: Assist. Prof. Dr. Burcu CAN, R. A. Necva BÖLÜCÜ** |
| **Programming Language** | **:JAVA** |
| **Subject** | **: Polymorphism and Abstract Classes** |

## 1. INTRODUCTION

In this experiment, you are expected to implement monopoly with the given rule. Main focus point of this experiment is to get you familiar with polymorphism and abstract classes.

Polymorphism is theability of an objecttotake on manyforms. Themostcommonuse of polymorphism in objectorienteddesignoccurswhen a parentclassreference is usedtoreferto a childclassobject.

A Java abstractclass is a classwhichcannot be instantiated, meaningyoucannotcreatenewinstances of an abstractclass. Thepurpose of an abstractclass is tofunction as a baseforthesubclasses.

Implementing the program without using the benefits of polymorphism and abstract classes will be penalized.

## 2. THE 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 into Land, Company and Rail Roads. Users are classified into Player and Banker and they have name and money attributes.

Your task is to design the game with the given rules by using polymorphism and abstract classes. You are supposed to document your program with Javadoc.

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 libraryforreadingandwriting JSON formattedfiles (withfullcompliancewithJSON 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": "60"},
        {"id": "4","name":"Whitechapel Road", "cost": "60"},
        {"id": "7","name":"The Angel Islington","cost": "100"},
        {"id": "9","name":"Euston Road", "cost": "100"},
        {"id": "10","name":"Pentonville Road","cost": "120"},
        {"id": "12","name":"Pall Mall","cost": "140"},
        {"id": "14","name":"Whitehall","cost": "140"},
        {"id": "15","name":"Northumberland Avenue","cost": "160"},
        {"id": "17","name":"Bow Street","cost": "180"},
        {"id": "19","name":"Marlborough Street","cost": "180"},
        {"id": "20","name":"Vine Street","cost": "200"},
        {"id": "22","name":"The Strand","cost": "220"},
        {"id": "22","name":"The Strand","cost": "220"},
        {"id": "24","name":"Fleet Street","cost": "220"},
        {"id": "25","name":"Trafalgar Square", "cost": "240"},
        {"id": "27","name":"Leicester Square","cost": "260"},
        {"id": "28","name":"Coventry Street","cost": "260"},
        {"id": "30","name":"Piccadilly","cost": "280"},
        {"id": "32","name":"Regent Street","cost": "300"},
        {"id": "33","name":"Oxford Street","cost": "300"},
        {"id": "35","name":"Bond Street","cost": "320"},
        {"id": "38","name":"Park Lane","cost": "350"},
        {"id": "40","name":"Mayfair","cost": "400"}
    ],
    "2": [
        {"id": "6","name":"Kings Cross Station","cost": "200"},
        {"id": "16","name":"Marylebone Station","cost": "200"},
        {"id": "26","name":"Fenchurch St Station","cost": "200"},
        { "id": "36","name":"Liverpool Street Station","cost": "200"}
    ],

    "3": [
        {"id": "13","name":"Electric Company", "cost": "150"},
        {"id": "29","name":"Water Works","cost": "150"}
        ]
}
```

### 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"}
]
}
```

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 banker's 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.

## 2.1 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 can 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 0-200 rent is 40%, for 201-300 30%, for 301-400 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 you have) 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 she passes by a "go" square, 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 (see Appendix B).

## 2.2. 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()
.....
```

An example command file is given below:

```
Player 1;9
Player 2;5
show()
Player 1;4
Player 2;4
Player 1;4
Player 2;6
Player 1;10
Player 2;7
Player 1;5
Player 2;3
```

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 player's money is zero or under zero.

## 2.3.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' money as shown in the figure given below:

**[Player]***tab***[Dice]***tab***[New Position]***tab***[Player1 Money]** *tab***[Player2 Money]** *tab***[Processing]**

```
Player 1        9       10      14880   15000   Player 1 bought Pentonville Road
Player 2        5       6       14880   14800   Player 2 bought King Cross Station
-------------------------------------------------------------------------------------------------------------
---
Player 1        14880   have: Pentonville Road
Player 2        14800   have: King Cross Station
Banker 100320
Winner Player 1
-------------------------------------------------------------------------------------------------------------
------
Player 1        4       14      14740   14800   Player 1 bought Whitehall
Player 2        4       10      14788   14752   Player 2 paid rent for Pentonville Road
Player 1        4       1       14988   14752   Player 1 draw Community Chest –advance to go
Player 2        6       16      14988   14552   Player 2 bought Marylebone Station
Player 1        10      11      14988   14552   Player 1 went to jail
Player 2        7       1       14988   14752   Player 2 advance to go (collect 200)
Player 1        5       11      14988   14752   Player 1 in jail (count=1)
Player 2        3       4       14988   14692   Player 2 bought Whitechapel Road
-------------------------------------------------------------------------------------------------------------
---
Player 1        14988   have: Pentonville Road, Whitehall
Player 2        14692   have: King Cross Station, Marylebone Station, Whitechapel Road
Banker 100320
Winner Player 1
-------------------------------------------------------------------------------------------------------------
```

### Submit Format

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

<student id>.zip

- javadoc.zip (See : https://en.wikipedia.org/wiki/Javadoc)

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

- Report.pdf (It will be same format with homework 1&2)

(See for report: ftp://ftp.cs.hacettepe.edu.tr/pub/dersler/genel/FormatForLabReports.doc)

### Late Policy

You may use up to three extension days for the assignment. But each extension day will bring about additional 10% degradation for evaluation of the assignment.

## Notes and Constraints

You should obey the constraints described below. Otherwise your experiment will not be evaluated as well as you expected.

- 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.
- You can ask your questions through course's 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
- 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.
- Every method and class must be documented using Javadoc.
- You are responsible for a correct model design. Your design should be accurate.
- Do not miss the deadline. Submission will be end at 20/04/2016 16:56:59, the system will be open 23:59:59. The problem about submission after 17:00:00 will not be considered.

## Appendix A.

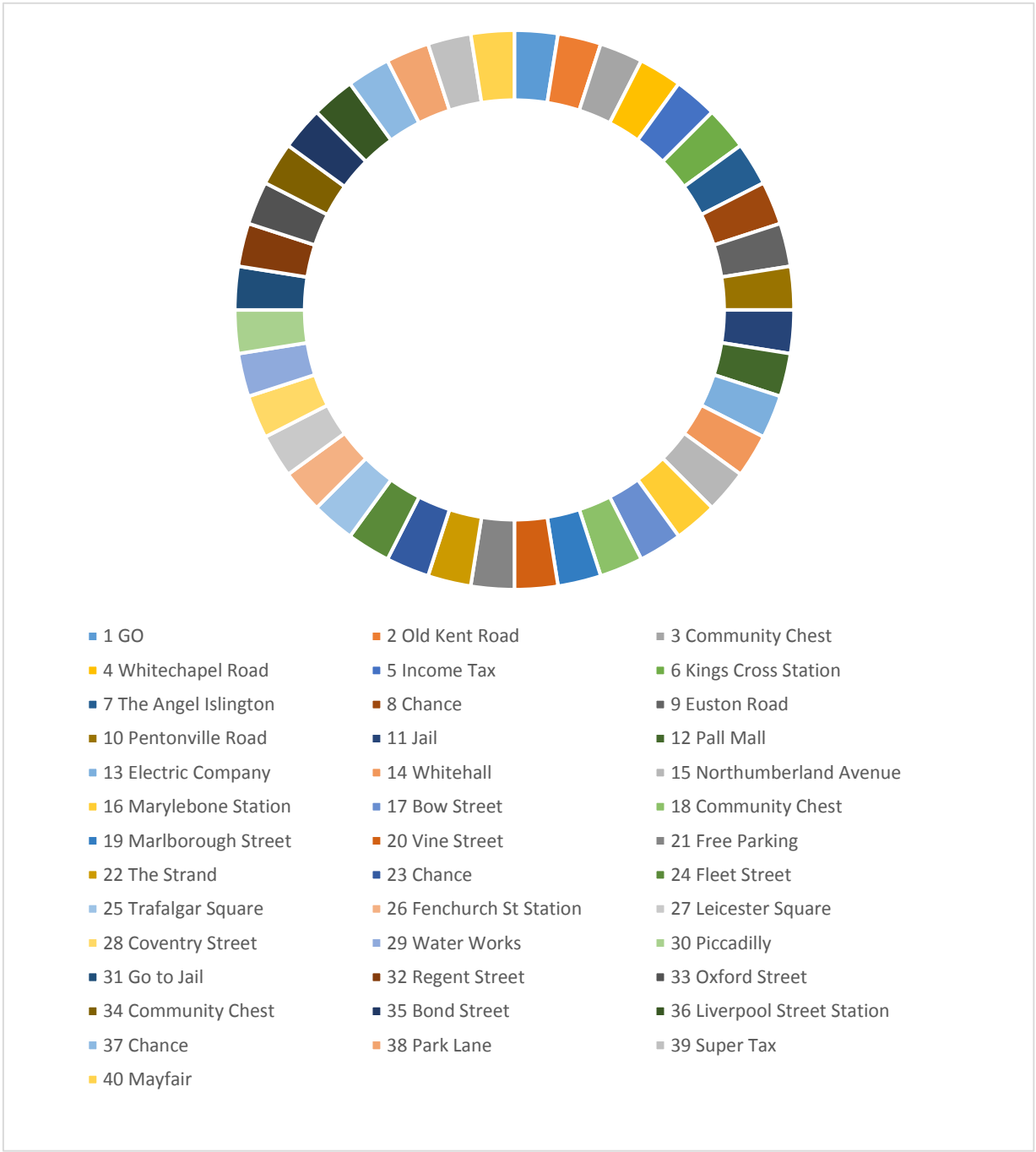
**JSON** (**JavaScript Object Notation**) is an open-standardformat that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is the most common data format used for asynchronous browser/server communication, largely replacing XML which is used by AJAX. There are 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"}
]}
```

## Appendix B.

The order of squares is given to help you.




## References

[1] https://en.wikipedia.org/wiki/JSON

[2] http://tutorials.jenkov.com/java/abstract-classes.html

[3] http://www.mkyong.com/java/json-simple-example-read-and-write-json/