REQUIREMENT SPEFICATION DOCUMENT FOR OOP PROJECT (MONOPOLY)

(FINAL ITERATION)

Alperen BAYAR 150115031
Büşra YAĞCI 150115057
Oğuzhan YİĞİT 150115042

VISION

We designed and implemented a monopoly game with using Java Language which is one of the most popular object oriented programming languages. Java Language allowed us creating working methods and variables, then re-use all or part of them without compromising security. Monopoly is a board game which is played by two to four players. It is played on a board with spaces. Players roll two six-sided dice to move around the game board, buying and trading properties, and developing them with houses and hotels. and/or buying proeprties from other competitors, with the goal being to drive them into bankruptcy.

PROBLEM STATEMENT AND SCOPE

To start the game, our program takes an input from user to select total number of players and names from the user. The game is moving forward with the dice. When it comes to each player, two dice are thrown then the player move around the board according to the sum of a throwed pair of dice.

In the game board there are 40 blocks/locations. Each side of the square board is divided into 10 small rectangles representing specific properties, cities, lucky cards, jails, tax administrations and various other places. At the start of the game, each player has a fixed amount of money which equals to 1500$.

Any player who lands on an unowned property/city may buy it, but if he/she lands on a property/city owned by another player, rent must be paid to owner. If the player doesn't have enough money to pay the rent price and if the player have properties/cities bought before, the player have to sell their lands(foreclose,haciz) till his money be enough to pay the current location's/city's rent price. If the player's money is still not enough despite the sale, the player will go bankrupt.

A player who lands on a goToJail location/property will be sent into the jail, and other round he/dhe wn't be able to play the game(for 1 round) In the other round he will exit out of jail and he will be able to play game again, and his/her new location will be the other corner of the board.

A player who lands on a Tax Administration property have to pay tax. If the player doesn't have enough money and if the player have another lands the player have to sell their lands till his money be enough to pay tax. If the player's money is still not enough despite the sale, the player will go bankrupt.

A player who lands on a LuckyCard location may face various situations. His/her money can be increased or decreased. He can be sent to Jail, tax administration or other cities. The appropriate

method/function of LocationLuckyCard class, which takes player to other cities, checks that it won't take him into another LuckyCard location by default.

A player continues to travel around the board until he/she will become bankrupt. Bankruptcy results in eliminating of player who goes to bankrupt . The last player remaining on the board is the winner.

SYSTEM REQUIREMENTS

We use Intellij because;

- Easily committing and pushing into GIT
- More descriptive warning,errors
- Various dev-friendly features like code autocompletion, lint

The Java version we use in this project is Java 8.

STAKEHOLDERS

- Alperen BAYAR
- Büşra YAĞCI
- Oğuzhan YİĞİT
- Murat Can GANİZ
- Berna ALTINEL

GLOSSARY OF TERMS

- Block: Blocks which pawn travels.
- Dice: Determine how far the proceed.
- LocationCity: City Locations.
- LocationJail: Jail Locations holding jail entrances and exits.
- LocationLuckyCard: LuckyCard Locations which perform various tasks randomly.
- LocationTaxAdmin: Tax Administration Locations, Users on this locations always pay the tax price