CSE3063 Object Oriented Analysis and Design
MONOPOLY GAME REPORT FOR the 2nd ITERATION

Group Members :

Yasin TOY	**150117830**
Mehmethan GÜR	**150116840**
Necmettin ILGIN	**150116838**

VİSİON

This is the last iteration of our project. We were needed to add some more properties to Monopoly Game as our customer requirements. Then according to this we designed our project and then wrote our code and tested it lastly. We always aimed to do the best project we could.

PROBLEM STATEMENT

Our project simulates the monopoly game according to the values which given by the user at the beginning of the game as in the other iterations. In our designing stage, we changed of course some plans and determined the constraints of the monopoly game simply and we defined our attributes and methods in our UML Class Diagram. Then we defined the ways which simulation follows step by step in our Sequence Diagram.

SCOPE

We determined the scope of our project as giving a simulation game for our customer requirements and making a successful project. We determined it as doing the best we could. While we are doing our project, we are always trying to keep it simple. Reaching our aim is important but at the end there should be an understandable and not a complex project ofcourse.

SYSTEM CONSTRAINTS

Addition to the other iterations we included some constraints to the game. This is the final state of all constraints:

1. 2 to 8 players,

2. A board which contains 40 squares,

3. And implemented different types of squares and their actions(they were defined in Class Diagrams) such as,
   - **GoSquare**
   - **Jail**
   - **GoJail**
   - **FreeParkingSquare**
   - **IncomeSquare**
   - **LuxurySquare**
   - **6 OrdinarySquare**
   - **22 LotsSquare**
   - **2 UtilitySquare**
   - **4 railsRoadSquare**

4. Pieces for each player,

5. 2 dice for moving our player to another location,

6. At most 6 + 6= 12 squares a player can move for each playDice(),

7. A “Money” class which keeps the amount of player’s Money.

8. The game will last until a single player remains in the game or it will last until the round number which is given at the beginning of the game.

STAKEHOLDERS

Murat Can Ganiz (Customer)
MehmetHan GÜR and Yasin TOY(Project Manager, Analyst/Programmer)
Necmettin ILGIN (Librarian, Analyst/Programmer)

GLOSSARY OF TERMS

Board: is a part which monopoly game is played on and contains 40 squares in it.

Die: has 6 faces and numbers from 1 to 6 in each face. Two dice gives the total number of moving step on the board.

Monopoly Game: is a game which players sell and buy estate until their money finish.

Piece: is an object which is a shape of good or a shape of an animal. Each piece refers to a player in the game.

Player: plays the monopoly game and tries not to bankrupt along the game.

Square: is a part of the board which gives directions to the player what to do.

GoSquare: is the start square of the game.

Jail: is a square that a player can not continue the game until it Rolls Doubles in next 3 turns or until it pays $50 for punishment.

Gojail: is the square which the player is sent to jail.

FreeParkingSquare: Nothing happens here when a player lands of it. The player goes on the next turn.

IncomeSquare: When a player lands on it he or she pays %10 of the total value of it’s Money.

LuxurySquare: When a player lands on it he or she pays $75 to cash means reduces $75 the amount of her or his Money.

LotsSquare: will be taken from a text file and will be written such as "Square 2" , "Square 6". There will be position, price and rent in the text. There are 22 "LotsSquare" in the game.

UtilitySquare: There are two UtilitySquare whose names are "ElectricUtility" and "WaterUtility" squares. These squares are landed on 13. and 29. squares. The price is for these is 150. Rent amount is 10 times the roll of a dice.

RailsRoadSquare: There are four railsRoadSquare. The price is 200 and the rent is 5 times the roll of a dice plus 25.

OrdinarySquare: is the name of other squares which has no any instances. I mean it refers to the remaining 6 squares.

Money: is given to each player at the beginning of the game at a certain amount and each player tries to have the most amount of Money in the game. According to the special squares, each player pays or receives some amout of it.

USE CASES

First enter the number of players and their names. Second, enter the number of rounds that you want to play. Then first player playDice acoording to the total value and the index of piece moves to a square on the board. At the end of each turn we are printing player's name, cash, current square, dice face values, dice total, landed square and actions associated with landed square.

If a player moves more than 40 squares he or she finishes the first round and passes to the second round.

If a player lands on the "GoSquare" while finisihing a round, he or she receives $200.

If a player lands on 4th square which means "IncomeSquare", he pays %10 of total amount of his Money.

If a player lands on the 10th square which means "go to jail" then player moves to "jail" square or if a player roll doubles three times in succession in one move then he or she goes to jail square.

If a player lands on 20th square which means FreeParkingSquare, nothing happens as punishment or reward he or she continues on the next turn.

If a player lands on the 30th square which means "jail" the states which ı mentioned above happens.

If a player lands on 34th square which means "LuxurySquare" he pays $75 to the bank meanly his Money amount reduces $75.

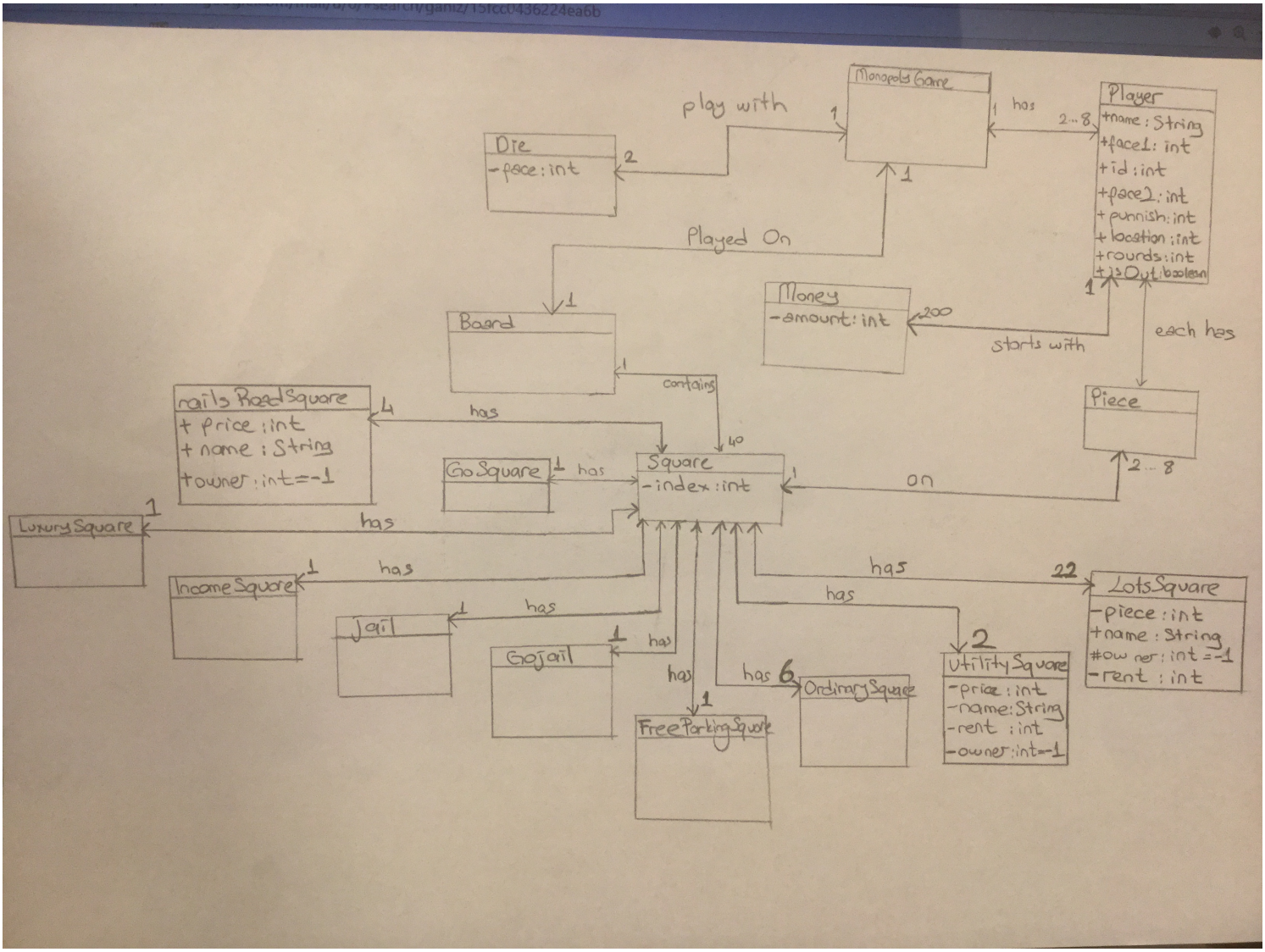
If a player lands on the LotsSquare(the positions of the squares are written in the text) he will pay the rent which will be taken from text file if to the owner, if it is owned by a player.

If a player lands on the 13. and 29. squares which mean "ElectricUtility" and "WaterUtility" squares, he will pay 10 times the roll of a dice to the owner.

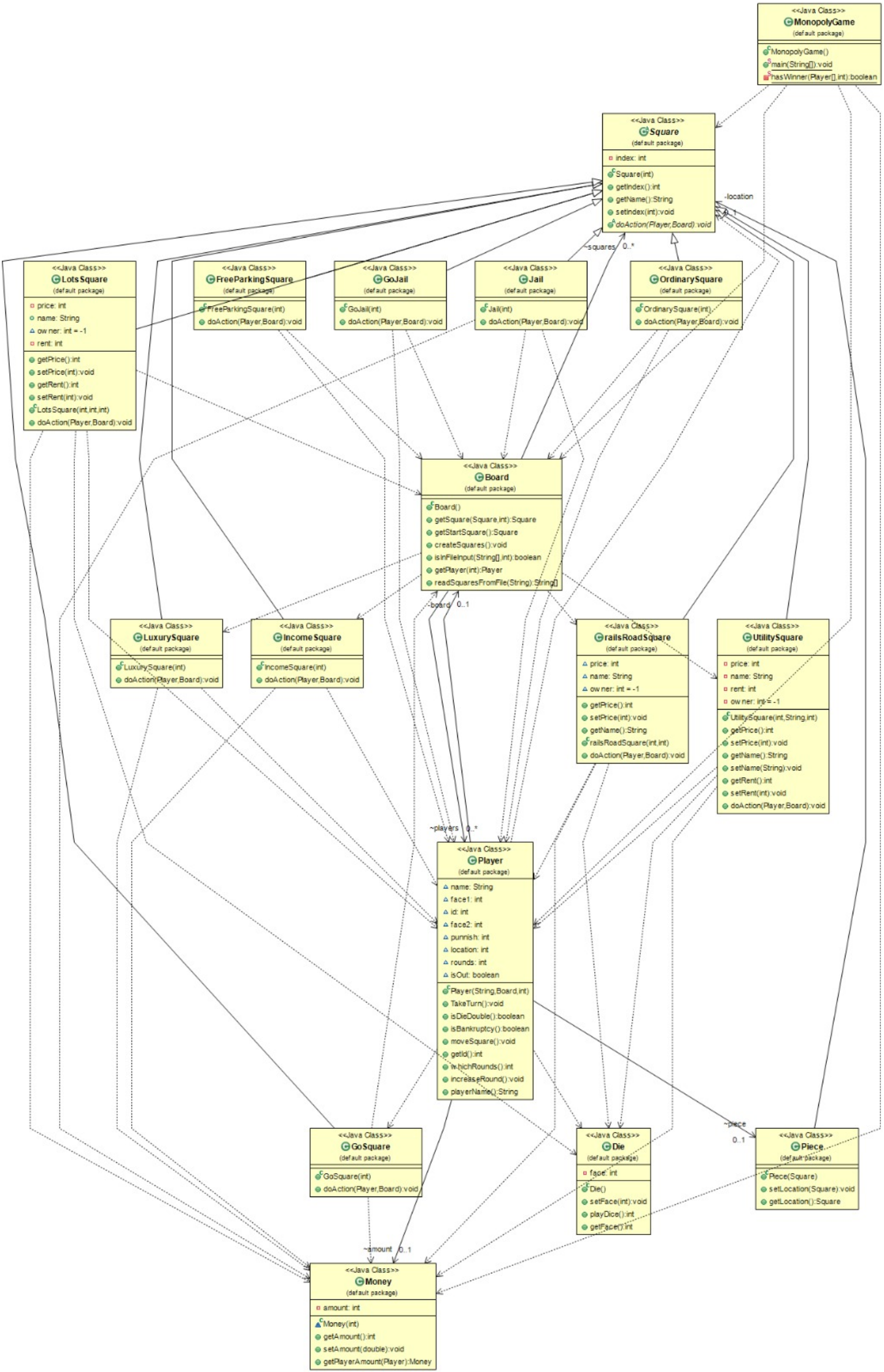
If a player lands on the 6. 16. 26. and 36. squares which mean "railsRoadSquare" he will pay the rent 5 times the roll of a dice plus 25.

And When a player who reaches the final round first, he or she wins the game and the game finishes or if a single player remains from the other players who go to bankrupt the game will finish.

UML DOMAIN MODEL



UML CLASS DIAGRAM


<<Java Class>>
MonopolyGame
(default package)
MonopolyGame()
main(String[]):void
hasWinner(Player[],int):boolean
<<Java Class>>
Square
(default package)
index: int
Square(int)
getIndex():int
getName():String
setIndex(int):void
doAction(Player,Board):void
-location
0..1
-squares
0..*
<<Java Class>>
LotsSquare
(default package)
price: int
name: String
owner: int = -1
rent: int
getPrice():int
setPrice(int):void
getRent():int
setRent(int):void
LotsSquare(int,int,int)
doAction(Player,Board):void
<<Java Class>>
FreeParkingSquare
(default package)
FreeParkingSquare(int)
doAction(Player,Board):void
<<Java Class>>
GoJail
(default package)
GoJail(int)
doAction(Player,Board):void
<<Java Class>>
Jail
(default package)
Jail(int)
doAction(Player,Board):void
<<Java Class>>
OrdinarySquare
(default package)
OrdinarySquare(int)
doAction(Player,Board):void
<<Java Class>>
Board
(default package)
Board()
getSquare(Square,int):Square
getStartSquare():Square
createSquares():void
isInFileInput(String[],int):boolean
getPlayer(int):Player
readSquaresFromFile(String):String[]
-board
0..1
<<Java Class>>
LuxurySquare
(default package)
LuxurySquare(int)
doAction(Player,Board):void
<<Java Class>>
IncomeSquare
(default package)
IncomeSquare(int)
doAction(Player,Board):void
<<Java Class>>
RailsRoadSquare
(default package)
price: int
name: String
owner: int = -1
getPrice():int
setPrice(int):void
getName():String
RailsRoadSquare(int,int)
doAction(Player,Board):void
<<Java Class>>
UtilitySquare
(default package)
price: int
name: String
rent: int
owner: int = -1
UtilitySquare(int,String,int)
getPrice():int
setPrice(int):void
getName():String
setName(String):void
getRent():int
setRent(int):void
doAction(Player,Board):void
-players
0..*
<<Java Class>>
Player
(default package)
name: String
face1: int
id: int
face2: int
punnish: int
location: int
rounds: int
isOut: boolean
Player(String,Board,int)
TakeTurn():void
isDieDouble():boolean
isBankruptcy():boolean
moveSquare():void
getId():int
whichRounds():int
increaseRound():void
playerName():String
<<Java Class>>
GoSquare
(default package)
GoSquare(int)
doAction(Player,Board):void
<<Java Class>>
Die
(default package)
face: int
Die()
setFace(int):void
playDice():int
getFace():int
-piece
0..1
<<Java Class>>
Piece
(default package)
Piece(Square)
setLocation(Square):void
getLocation():Square
-amount
0..1
<<Java Class>>
Money
(default package)
amount: int
Money(int)
getAmount():int
setAmount(double):void
getPlayerAmount(Player):Money

UML SEQUENCE DIAGRAM