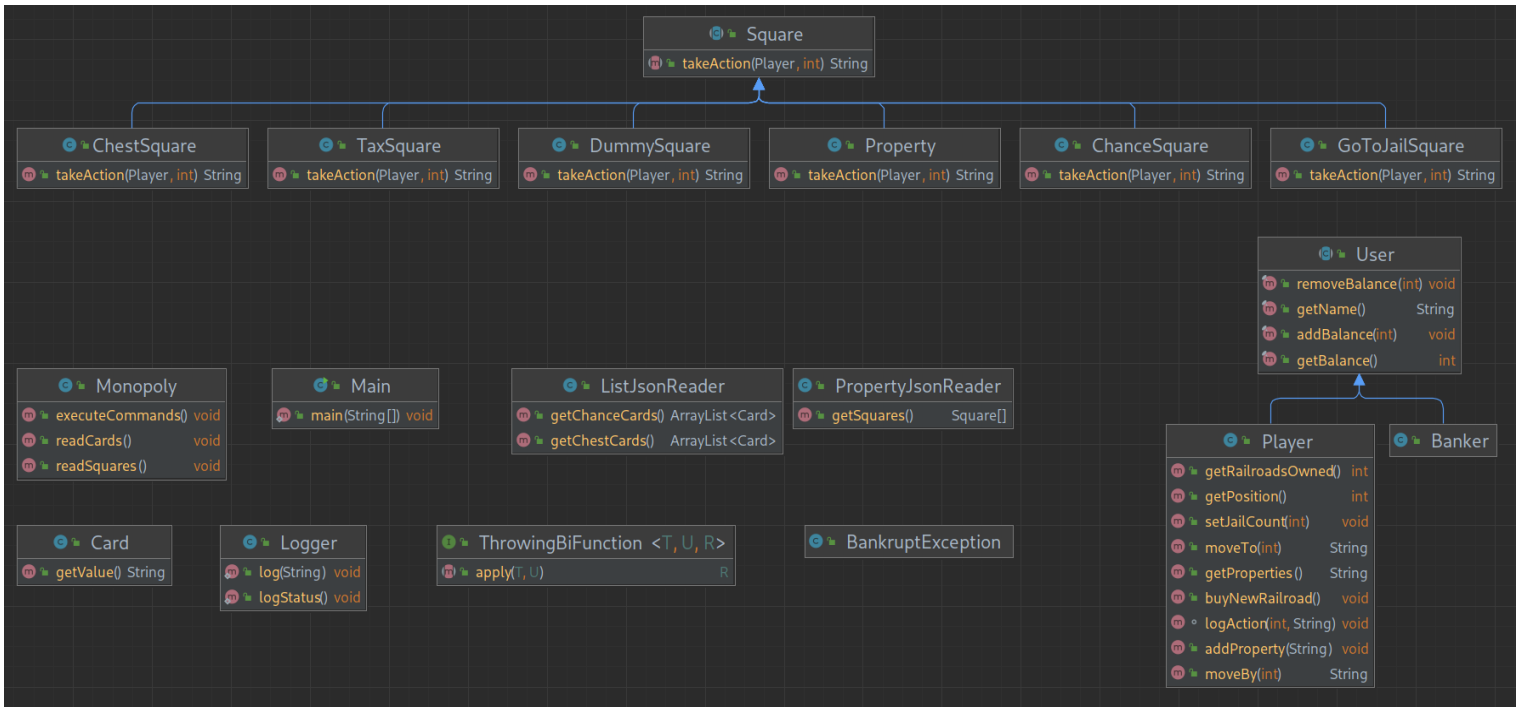
# BBM104 Assignment 2 Report

İlker AVCI 2210356061 01/04/2022

In this assignment, I designed such a UML diagram. Note that this diagram doesn't show private methods.




I created a Square class to make a base class for all the squares in the game. It's an abstract class so it can't be initialized directly. It has an abstract method named takeAction(), takes the player of the action and the value of dice as parameters, and returns a string containing the information that is going to be logged in output.txt file.

There are 6 subclasses belonging to Square base class, namely ChestSquare, ChanceSquare, TaxSquare, GoToJailSquare, Property and DummySquare.
DummySquare is for squares which do not have any action related to them, they're just for logging that the player was there. DummySquare takes the place name as the argument for its constructor. There are only 2 squares that make use of DummySquare: "Go Square" and "Free Parking Square".
The rest of subclasses of Square class are pretty much self-explanatory and neither take any argument for their constructors, nor have a constructor method at all.

There's a User class which is also an abstract class, providing a base for Player and Banker classes. It has 4 methods and they have all self-explanatory names as well.

Banker class has nothing special, just has a predefined balance of 100000TL.

Player class has a bunch of methods:

- getRailroadsOwned() returns the total number of railroads the player owns
- buyNewRailroad() increases the number of railroads the player owns by one
- getPosition() returns the position of the player
- moveTo(int) moves the player to a certain square
- moveBy(int) moves the player by a certain amount
- setJailCount(int) sets the number of rounds player should wait in jail
- addProperty(String) adds the given property to the possessions of the player
- getProperties() returns all the properties the player owns, seperated with a comma (,)
- logAction(String) logs the given action into the output file

ListJsonReader and PropertyJsonReader were given as starter codes, I just added some getters for them.

Card class is meant to be used for both community chest cards and chance cards. They're initialized with the text of the card in the constructor, and have a method of ThrowingBiFunction class which is to be called later. getValue() method simply returns the value (the text) of the card.

Logger is a static class to manage all sorts of logging operations in this game. It has a method called log(String) for logging a line of string and logStatus() for logging the current status to the output file.

ThrowingBiFunction is essentially the same as java.util.function.BiFunction, but allows its method to throw exceptions of type BankruptException.

BankruptException has nothing different than its super class Exception, all it does is to distinguish regular Exceptions from the ones that are thrown because of a player going bankrupt.

Monopoly class is the class where everything is managed to play the game. It has functions with self-explanatory names. These functions are called from the Main class.

Main class doesn't have much in it, it just creates a new instance of Monopoly class and calls its required methods.

I also wrote javadoc-style comments all over my code to make it easier to understand. I only avoided some getter and setter functions when writing these comments, because their purpose is pretty much obvious and don't need any explanation. I also added in-line comments when needed and used legible names for classes and variables.