Helper Document

Project 1: Monopoly

The objective of the project was to implement the game of Monopoly in JAVA language using the concepts of Object Oriented Programming. To improve the user interaction, we have added a graphical user interface(GUI) to the game. It is implemented using JavaFX.
We have also provided updated UML diagram which shows the relations as well as the interactions between various classes.

Execution Instructions

The code was tested on Pop!_OS 21.04 which is an ubuntu-based Linux distro using Visual Studio Code. Here I will outline the instructions to execute the application:

- Install JDK 17 using sudo apt install openjdk-17-jdk
- We have used this Java Extension on VS Code.
- Download SDK from here.
- Unzip it to a desired location
- Add an environment variable pointing to the lib directory of the runtime in .bashrc
- export PATH_TO_FX=path/to/javafx-sdk-17/lib
- In VS Code settings.json file in the project directory, add the path to lib folder to referencedLibraries like this:

```
{
    "java.project.sourcePaths": ["src"],
    "java.project.outputPath": "bin",
    "java.project.referencedLibraries": [
        "lib/**/*.jar",
        "/home/aviralomar/development/sdk/javafx-sdk-17.0.1/lib/*.jar"
```

```
        ]
}
```

- Install following: `sudo apt install libswt-gtk-4-java`
- To compile the code go into the bin folder of the project directory and run:

`javac --module-path $PATH_TO_FX --add-modules javafx.controls ../src/*.java -d .`

- To execute the application, from the bin directory, run:

  `java --module-path $PATH_TO_FX --add-modules javafx.controls Monopoly`

- Getting Started with JavaFX for further reference.

Class Description

We have made use of the following classes:

- Actor
- Bank
- Board
- Card
- CardDeck
- ChanceCard
- ChanceDeck
- ChanceSquare
- CommunityChestCard
- CommunityChestDeck
- CommunityChestSquare
- FreeParkingSquare

- Game
- GoSquare
- GoToJailSquare
- JailSquare
- Monopoly
- Player
- PropertySquare
- RealEstateDeed
- RealEstateSquare
- Square
- StationDeed
- StationSquare
- TaxSquare
- TitleDeed
- UI
- UtilityDeed
- UtilitySquare

The code execution starts from the main() function which is written in the Monopoly class of Monopoly.java file which sends control over to the Game class of Game.java file which calls all other functions which help in simulating the game. The code for GUI is written in the UI.java file.

Documentation of the classes

Actor

The class is an abstract class.

Variables: It has an int variable cash, int variable housesOwned and int variable hotelsOwned all of private access. It also has an Arraylist of TitleDeed type called titledeeds which is of protected type.

Functions: The class has a getter for cash variable. The addCash function increases the value of cash by a specific amount. The deductCash function reduces cash by specified amount. The addDeed and removeDeed functions adds and removes deed in the arraylist.

Bank

The bank class inherits the Actor class.
Variables: It has an arraylist titleDeeds which stores details like property name, rent, property colour etc about all the properties.
The constructor stores all the details in the arraylist. The class also has a searchDeed function which searches for the name of a particular property in the arraylist.

Board

Variables: The board class has an arraylist squares of the type Squares which is used to store the details of all 40 squares of the monopoly game board.
The constructor initialises the arraylist. The function getSquare returns the square at a specified position.

CardDeck

It is an abstract class.
Variable: Deck which is a linked list.
Functions: shuffleDeck functions shuffles the linked list by calling the shuffle function of Collections class. The pickFromTop function returns the deck after removing the first element of the linked list. The insertAtBottom function adds a specific card to the end of the linked list.

ChanceCard

The class inherits the class Card.

Variable: private variable action of the type ChanceActions
Functions: The function getAction returns the action

ChanceDeck
The class inherits the class CardDeck
Functions: It has a getMessage function which returns the text mentioned on the chance card which is picked from the deck.
CommunityChestCard
The class inherits the class Card.
Variable: private variable action of the type CommunityChestActions
Functions: The function getAction returns the action.\

CommunityChestDeck
The class inherits the class CardDeck
Functions: It has a getMessage function which returns the text mentioned on the community chest card which is picked from the deck.

Game
This class controls the functioning of the game. It calls all relevant functions to simulate the game.
Variable: ui of the type UI, board of the type Board, bank of the type Bank, communityChestDeck of the type CommunityChestDeck and chanceDeck of the type ChanceDeck. All the variables are private to the class.
Functions: useGetOutOfJailCard function checks if the player has a get out of jail card. If found the player gets out of jail else he continues to stay there. The communityChestAction function handles all the cases which occurs when a player arrives at a community chest square. The chanceActions function deals with all the cases which takes place when a player arrives on a chance square. The action function checks the type

of square which the player has arrived and deals with it accordingly. For each type of square, it checks whether the property i owned by the bank or a player. The rollDie function simulates die rolls.The simulateTurn functions simulates a turn and the simulateRound function simulates a complete round involving 1 turn of all the players. The displayState function displays the game state at the end of a round.

Monopoly
The class executes the GUI code and is the starting point of execution. The main function starts the execution of the code. The start function launches the GUI.

Player
The class inherits the Actor class.
Variables: All variables are of private access type. It has name of String type, position of type int, inJail of bool type, turnsInJail of int type, chanceGetOutOfJail of ChanceCard type and communityGetOutOfJail of communityChestCard type.
Functions: the class has a getter for name, getter and setter for position, getter for utilities and stations owned. The function ownsAllOfColour check if the player owns all properties of a particular colour. It also has setters for get out of jail card of chance and community chest decks. The sendToJail function sets the position of the player to Jail. the isInJail checks if the player is in jail. The class also have getter and setter for turns taken in jail. It has a checkGetOutOfJail function which checks for the conditions when a player can get out of Jail.

Square
It is a representation of a square on the game board.
Variables: String name which is the name of the square.

Methods: getName to access the name. toString also returns the name as string representation of the square.

PropertySquare (inherits from Square)

It represents one of 3 types of properties: Utility, Station or RealEstate.
Variables: TitleDeed titleDeed stores the title deed associated with the square.
Methods: getOwner returns the owner (player or bank of the title deed). buyProperty allows the player landing on the property to buy it from the bank or from another player.

UtilitySquare (inherits from PropertySquare)

Represents a utility square on the board.

StationSquare (inherits from PropertySquare)

Represents a station square on the board.

RealEstateSquare (inherits from PropertySquare)

Represents a utility square on the board.
Variables: int numberOfHouses stores the number of houses currently on the square. Bool hasHotel indicates whether the square contains a hotel.
Methods: getRent calculates the rent for landing on property using the title deed and the number of houses on the square. getColour returns the colour of the real estate square.

ChanceSquare (inherits from Square)

Represents a chance square on the board.

CommunityChestSquare (inherits from Square)

Represents a community chest square on the board.

GoToJailSquare (inherits from Square)

Represents a Go To Jail square on the board.

JailSquare (inherits from Square)

Represents the jail square on the board.

GoSquare (inherits from Square)

Represents the GO square on the board.

FreeParkingSquare (inherits from Square)
Represents the free parking square on the board.

TaxSquare (inherits from Square)
Represents a Luxury Tax or Income Tax square on the board.
Variables: int amount is the amount of Tax to be paid by the player landing on the board.
Methods: getAmount allows access to the tax amount variable.

TitleDeed
Represents a ticket for any type of property in the game.
Variables: String name stores name of property. Actor owner could be any Player or the Bank. int price is the price of the property. Int mortgageValue is the mortgage value of the property which is unused in our application.
Methods: getName, getOwner, setOwner, getPrice are getter and setter methods for the various properties. toString returns the name as string representation of the title deed.

StationDeed (inherits from TitleDeed)
Represents a title deed for a station property.

UtilityDeed (inherits from TitleDeed)
Represents a title deed for a utility property.

RealEstateDeed (inherits from TitleDeed)
Represents a title deed for a real estate property.
Variables: Colour colour stores the colour of the real estate. Int costOfHouse stores the cost of 1 house on the property.
Methods: getRent calculates the rent for landing on the property. getColour returns the colour of the property.