API Design Document

for

Monopoly

Prepared by:

MAN Furui (19081789d)
LIU Sicheng (19079181d)
WANG Meng (19078543d)
XING Shiji (19079008d)

The Hong Kong Polytechnic University
Software Engineering
Course Project
Prof. Yu PEI

Hong Kong, 22 October 2021

Contents

Chapter 1

Introduction

1.1 Purpose

This document mainly illustrates important design decisions of the architectures of the product. The product is implemented following the MVC model. However, since the product is based on command line, basic prompts are implemented in controller model, while the printing of the game board is handled in a specific class in view model.

1.2 References

- COMP3211 Software Engineering Course Project Description

1.3 Monopoly Class Diagram

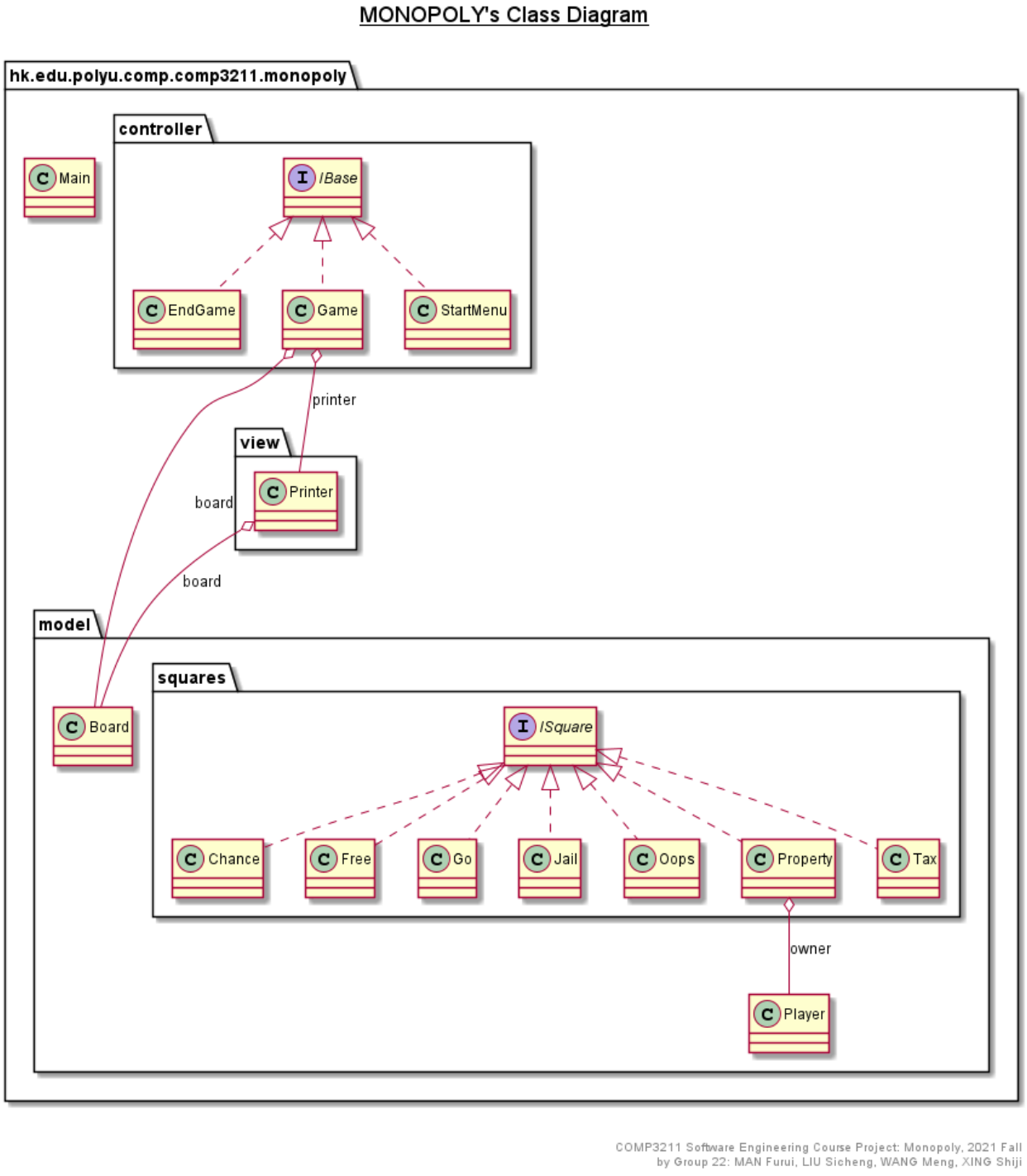

Figure 1.1: Monopoly Class Diagram

Chapter 2

Main

The entry point of the application.

A globally-shared scanner is defined in this class for other classes to use. The main control flow of `Main()` is calling the `terminal()` methods of different controller interface in a infinite while loop. Different controller handles different scenarios in the application, such as start menu, in game and end game. At the beginning of the application, `Main()` will create a StartMenu controller interface and pass the control to it.

Detailed description of controller class will be noted in the controller section.

Chapter 3

Controller

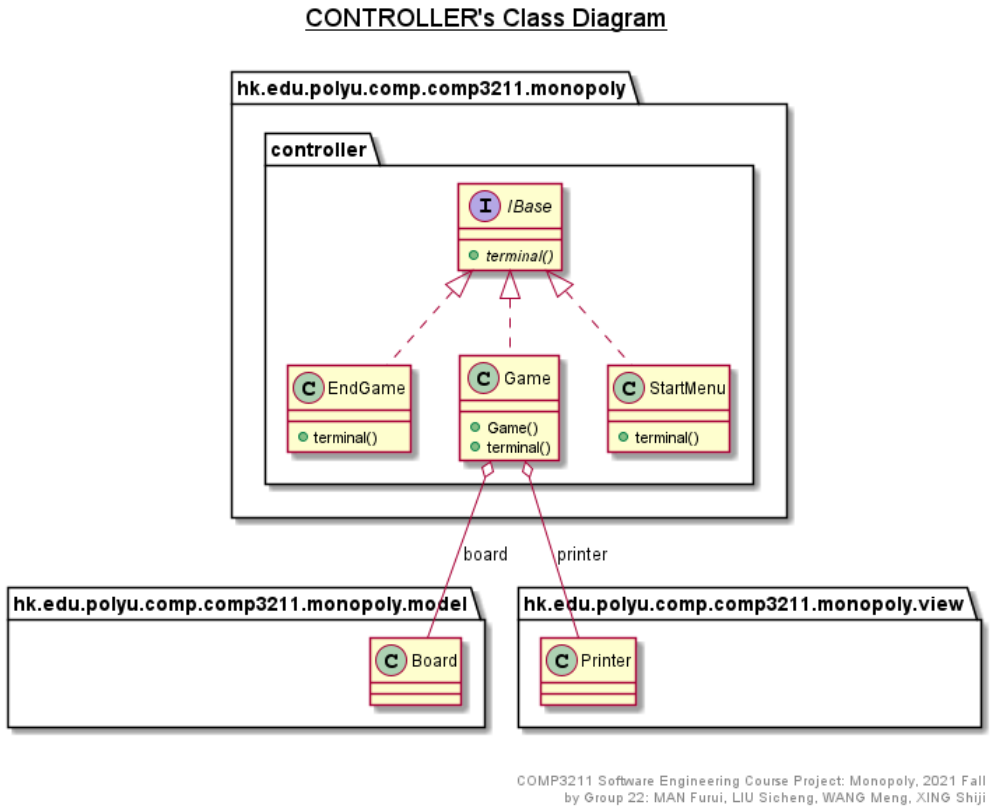

Figure 3.1: Monopoly Controller Class Diagram

All classes in this model should implement the interface IBase, which has a method void `terminal()`. this method is called by `Main()` in a infinite loop, and is designed to process one or more user inputs at a time. Several classes are defined implementing this interface, and they are used in different interfaces. Specifically, `StartMenu.terminal()` is called when the game is in start menu, and `Game.terminal()` is called when it is in the middle of a game. Same applies to `EndGame.terminal()`, when a game ends. The Main class will have a reference to the current interface, and this can be changed by controller classes in order to perform interface switching.

Chapter 4

Model

This section defines the model of the application, in which there are several significant components, namely the board, the square, and the player, implemented respectively by classes under the same reference. Details of each component class are covered in its corresponding section.

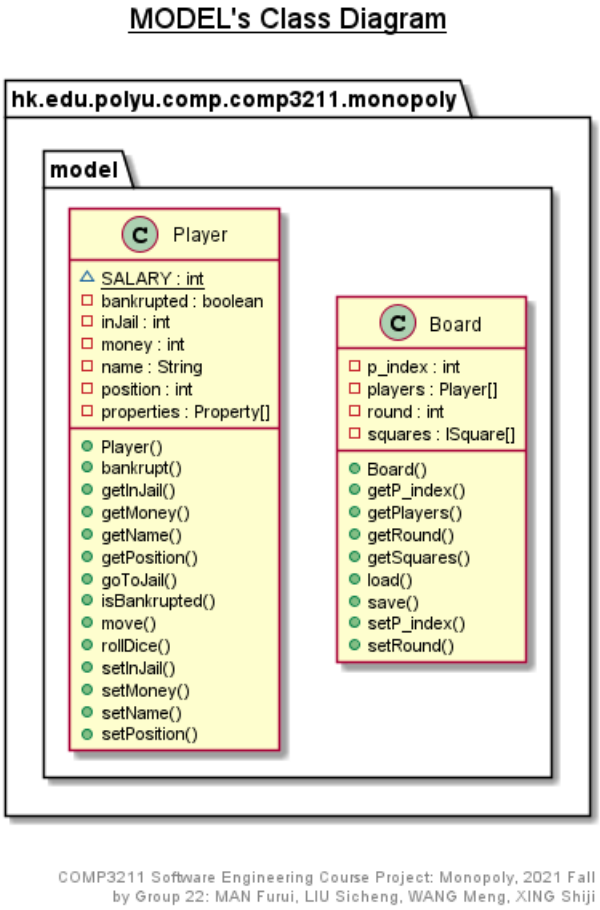

Figure 4.1: Monopoly Model Class Diagram

4.1 Board

This class stores the information of the game. It keeps an array of players and an array of squares. The layout of the board (along with its functionalities) is initialized in the constructor and is stored in the array of squares. Furthermore, it also records the round of the game and current actioning player. The board also supports saving and loading to/from local file.

4.2 Squares

The basic unit of map. All subclasses should implement the interface ISquare, which has one method void `execute(Player player)`. The method is called to perform specific operations on the current user stepping on the square. The following classes implements this interface: Chance, Free, Go, Jail, Oops, Property and Tax. The name of classes correspond to the type of squares in Appendix B of the group project document. Note that class Oops refers to the square "Go to Jail".

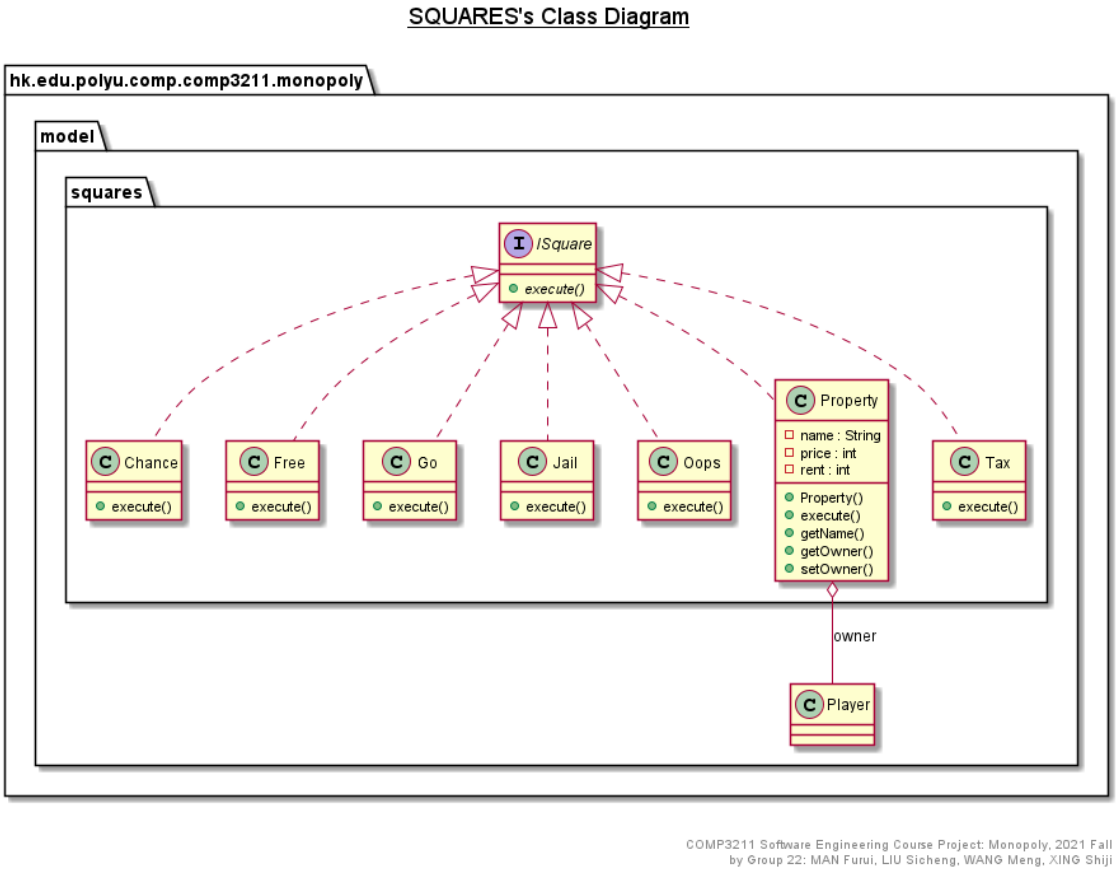

Figure 4.2: Monopoly Squares Class Diagram

4.3 Player

The class storing information about players. It should store players' name, money, current position, jail state, etc. In addition, it should supports auxiliary methods that adds money, moves forward in the map, teleport to Jail and rolls the dice. Moreover, it should have a `Property[]` array to store the properties owned by the player.

Chapter 5

View

The model only contains one class `printer`, which is used to print out the game board during rounds.

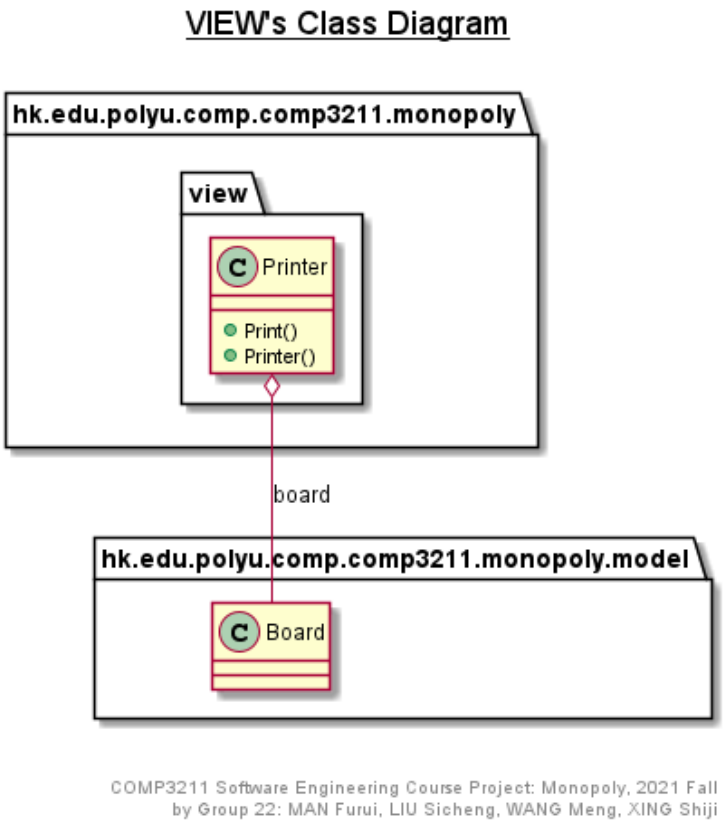

Figure 5.1: Monopoly View Class Diagram

Chapter 6

Appendices

6.1 API Documentation

Packages

Package	Description
hk.edu.polyu.comp.comp3211.monopoly	
hk.edu.polyu.comp.comp3211.monopoly.controller	
hk.edu.polyu.comp.comp3211.monopoly.model	
hk.edu.polyu.comp.comp3211.monopoly.model.squares	
hk.edu.polyu.comp.comp3211.monopoly.view	

Package hk.edu.polyu.comp.comp3211.monopoly

Class Summary

Class	Description
Main	

Package hk.edu.polyu.comp.comp3211.monopoly.controller

Interface Summary

Interface	Description
IBase	

Class Summary

Class	Description
EndGame	
Game	
StartMenu	

Package hk.edu.polyu.comp.comp3211.monopoly.model

Class Summary

Class	Description
Board	A board, containing players, squares, and game status
Player	A player and its status in the game

Package hk.edu.polyu.comp.comp3211.monopoly.model.squares

Interface Summary

Interface	Description
ISquare	Any square of the board

Class Summary

Class	Description
Chance	The Chance square of the board
Free	The Free-Parking square of the board
Go	The Go square of the board
Jail	The In-Jail/Just-Visiting square of the board
Oops	The Go-to-Jail square of the board
Property	The Property squares of the board
Tax	The Income-Tax square of the board

Package hk.edu.polyu.comp.comp3211.monopoly.view

Class Summary

Class	Description
Printer	

Package hk.edu.polyu.comp.comp3211.monopoly

Class Main

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.Main

```
public class Main
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor	Description
`Main()`	

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
static java.util.Scanner	`GetScanner()`	If a method needs to get user input, use this static method to get the scanner
static void	`main`(java.lang.String[] args)	
static void	`setUI`(`IBase` ui)	Switch user interface

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Main

```
public Main()
```

Method Detail

main

public static void main(java.lang.String[] args)

GetScanner

public static java.util.Scanner GetScanner()

If a method needs to get user input, use this static method to get the scanner

Returns:

The scanner

setUI

public static void setUI(IBase ui)

Switch user interface

Parameters:

`ui` - the new interface

Package hk.edu.polyu.comp.comp3211.monopoly.controller

Interface IBase

All Known Implementing Classes:

EndGame, Game, StartMenu

```
public interface IBase
```

Method Summary

All Methods	Instance Methods	Abstract Methods

Modifier and Type	Method	Description
void	**terminal**()	Prompt , parse and process user commands

Method Detail

terminal

```
void terminal()
```

Prompt , parse and process user commands

Package hk.edu.polyu.comp.comp3211.monopoly.controller

Class EndGame

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.controller.EndGame

All Implemented Interfaces:

IBase

```
public class EndGame
extends java.lang.Object
implements IBase
```

Constructor Summary

Constructors

Constructor	Description
EndGame()	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	**terminal**()	Print the game over message, and ask if the player want to restart, load or quit.

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EndGame

```
public EndGame()
```

Method Detail

terminal

```
public void terminal()
```

Print the game over message, and ask if the player want to restart, load or quit.

Specified by:

`terminal` in interface `IBase`

Package hk.edu.polyu.comp.comp3211.monopoly.controller

Class Game

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.controller.Game

All Implemented Interfaces:

`IBase`

```
public class Game
extends java.lang.Object
implements IBase
```

Constructor Summary

Constructors

Constructor	Description
`Game()`	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	`terminal()`	First print game board, current round and player; If he is in jail, refer to the document Otherwise, prompt the user to roll the dice(user needs to input something); Then player should move forward; then call the execute method of the arriving square.

Methods inherited from class java.lang.Object

`clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Game

```
public Game()
```

Method Detail

terminal

```
public void terminal()
```

First print game board, current round and player; If he is in jail, refer to the document Otherwise, prompt the user to roll the dice(user needs to input something); Then player should move forward; then call the execute method of the arriving square. Note that user can save, load or quit at this time. At the beginning of each round, check if the game ends. If so, print the game result and switch to the Endgame Interface

Specified by:

`terminal` in interface `IBase`

Package hk.edu.polyu.comp.comp3211.monopoly.controller

Class StartMenu

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.controller.StartMenu

All Implemented Interfaces:

`IBase`

```
public class StartMenu
extends java.lang.Object
implements IBase
```

Constructor Summary

Constructors

Constructor	Description
`StartMenu()`	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`void`	`terminal()`	Welcome the user.

Methods inherited from class java.lang.Object

`clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

StartMenu

```
public StartMenu()
```

Method Detail

terminal

```
public void terminal()
```

Welcome the user. If there exists a save file in the directory, prompt to the user of the option to load the game. User can try to start a game; load a game if any; and exit in this interface.

Specified by:

`terminal` in interface `IBase`

Package hk.edu.polyu.comp.comp3211.monopoly.model

Class Board

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.model.Board

```
public class Board
extends java.lang.Object
```

A board, containing players, squares, and game status

Constructor Summary

Constructors

Constructor	Description
Board(int num)	Initialize the board with fixed number of squares and customized number of players (2-6)

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
int	**getP_index**()	Get the index of current active player
Player[]	**getPlayers**()	Get all players in the board
int	**getRound**()	Get the index of current round
ISquare[]	**getSquares**()	Get all squares in the board
void	**load**(java.lang.String name)	Load the board from a local file
void	**save**(java.lang.String name)	Save the board to a local file
void	**setP_index**(int p_index)	Set active player index to a custom number
void	**setRound**(int round)	Set round index to a custom number

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Board

public Board(int num)

Initialize the board with fixed number of squares and customized number of players (2-6)

Parameters:

`num` - number of players in the board

Method Detail

getPlayers

public Player [] getPlayers()

Get all players in the board

Returns:

array of players

getSquares

public ISquare [] getSquares()

Get all squares in the board

Returns:

array of squares

getRound

public int getRound()

Get the index of current round

Returns:

round index

setRound

public void setRound(int round)

Set round index to a custom number

Parameters:

`round` - dest round index

getP_index

public int getP_index()

Get the index of current active player

Returns:

round index

setP_index

public void setP_index(int p_index)

Set active player index to a custom number

Parameters:

`p_index` - dest player index

save

public void save(java.lang.String name)

Save the board to a local file

Parameters:

`name` - the path (name) of the local file

load

public void load(java.lang.String name)

Load the board from a local file

Parameters:

`name` - the path (name) of the local file

Package hk.edu.polyu.comp.comp3211.monopoly.model

Class Player

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.model.Player

```
public class Player
extends java.lang.Object
```

A player and its status in the game

Constructor Summary

Constructors

Constructor	Description
Player()	initialize a player and scan input from user

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	**bankrupt**()	The player is bankrupt
int	**getInJail**()	Get the "IN JAIL" status of the player
int	**getMoney**()	Get the balance of the player
java.lang.String	**getName**()	Get the name of the player
int	**getPosition**()	Get the position of the player
void	**goToJail**()	The player goes to jail
boolean	**isBankrupted**()	Judge whether the player is bankrupted
void	**move**(int step)	Advance by the number of steps, note 20th square and 1st square is connected If player goes past the starting position, give him salary(with notice).
int	**rollDice**()	Roll a dice which is uniformly distributed from 1-4
void	**setInJail**(int inJail)	Set the "IN JAIL" status to a custom number
void	**setMoney**(int money)	Set balance of the player to a custom number
void	**setName**(java.lang.String name)	Set name of the player to a custom string
void	**setPosition**(int position)	

Set position of the player to a custom number

Methods inherited from class java.lang.Object

`clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Player

`public Player()`

initialize a player and scan input from user

Method Detail

getName

public java.lang.String getName()

Get the name of the player

Returns:

name of the player

setName

public void setName(java.lang.String name)

Set name of the player to a custom string

Parameters:

`name` - dest. name

getMoney

public int getMoney()

Get the balance of the player

Returns:

balance of the player

setMoney

public void setMoney(int money)

Set balance of the player to a custom number

Parameters:

`money` - dest. balance

getPosition

public int getPosition()

Get the position of the player

Returns:

position of the player

setPosition

public void setPosition(int position)

Set position of the player to a custom number

Parameters:

`position` - dest. position

move

public void move(int step)

Advance by the number of steps, note 20th square and 1st square is connected If player goes past the starting position, give him salary(with notice).

Parameters:

`step` - number of steps

goToJail

public void goToJail()

The player goes to jail

bankrupt

public void bankrupt()

The player is bankrupt

getInJail

public int getInJail()

Get the "IN JAIL" status of the player

Returns:

"IN JAIL" status

setInJail

public void setInJail(int inJail)

Set the "IN JAIL" status to a custom number

Parameters:

`inJail` - dest. "IN JAIL" status

isBankrupted

public boolean isBankrupted()

Judge whether the player is bankrupted

Returns:

true if bankrupted, else false

rollDice

public int rollDice()

Roll a dice which is uniformly distributed from 1-4

Returns:

the result (ranged from 1-4)

Package hk.edu.polyu.comp.comp3211.monopoly.model.squares

Interface ISquare

All Known Implementing Classes:

Chance, Free, Go, Jail, Oops, Property, Tax

```
public interface ISquare
```

Any square of the board

Method Summary

All Methods	Instance Methods	Abstract Methods

Modifier and Type	Method	Description
void	**execute**(**Player** player)	Generate an effect to a player

Method Detail

execute

void execute(Player player)

Generate an effect to a player

Parameters:

player - dest. player

Package hk.edu.polyu.comp.comp3211.monopoly.model.squares

Class Chance

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.model.squares.Chance

All Implemented Interfaces:

`ISquare`

```
public class Chance
extends java.lang.Object
implements ISquare
```

The Chance square of the board

Constructor Summary

Constructors

Constructor	Description
`Chance()`	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`void`	`execute(Player player)`	Generate an effect to a player

Methods inherited from class java.lang.Object

`clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Chance

```
public Chance()
```

Method Detail

execute

public void execute(Player player)

Generate an effect to a player

Chance Effect:

- The player either gains a random amount (n*10) up to 200;
- or loses a random amount (n*10) up to 300.

Specified by:

`execute` in interface `ISquare`

Parameters:

`player` - dest. player

Package hk.edu.polyu.comp.comp3211.monopoly.model.squares

Class Free

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.model.squares.Free

All Implemented Interfaces:

`ISquare`

```
public class Free
extends java.lang.Object
implements ISquare
```

The Free-Parking square of the board

Constructor Summary

Constructors

Constructor	Description
Free()	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	**execute**(**Player** player)	Generate an effect to a player

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Free

```
public Free()
```

Method Detail

execute

public void execute(Player player)

Generate an effect to a player

Free-Parking Effect:

- None.

Specified by:

`execute` in interface `ISquare`

Parameters:

`player` - dest. player

Package hk.edu.polyu.comp.comp3211.monopoly.model.squares

Class Go

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.model.squares.Go

All Implemented Interfaces:

`ISquare`

```
public class Go
extends java.lang.Object
implements ISquare
```

The Go square of the board

Constructor Summary

Constructors

Constructor	Description
`Go()`	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`void`	`execute(Player player)`	Generate an effect to a player

Methods inherited from class java.lang.Object

`clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Go

```
public Go()
```

Method Detail

execute

public void execute(Player player)

Generate an effect to a player

Go Effect:

- The player passes through (not necessarily lands on):

  gets $1500 salary;

- All players starts from this square, and at this time:

  this square has no effect.

Specified by:

`execute` in interface `ISquare`

Parameters:

`player` - dest. player

Package hk.edu.polyu.comp.comp3211.monopoly.model.squares

Class Jail

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.model.squares.Jail

All Implemented Interfaces:

ISquare

```
public class Jail
extends java.lang.Object
implements ISquare
```

The In-Jail/Just-Visiting square of the board

Constructor Summary

Constructors

Constructor	Description
Jail()	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	execute(Player player)	Generate an effect to a player

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Jail

```
public Jail()
```

Method Detail

execute

public void execute(Player player)

Generate an effect to a player

In-Jail/Just-Visiting Effect:

- if the player just passes:

  this square has no effect

- if the player is at in-jail status:

  cannot make a move, and can only get out by:

  1. throwing doubles (two dice with same results) within next 3 turns

     (and if so, immediately moves by the throw);

  2. paying $150 before rolling the dice within the next 2 turns;
  3. if still not out within 3 turns, paying $150 (must) to get out

     (and if `cond.2` or `cond.3` is met, the player can throw once and move).

Specified by:

`execute` in interface `ISquare`

Parameters:

`player` - dest. player

Package hk.edu.polyu.comp.comp3211.monopoly.model.squares

Class Oops

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.model.squares.Oops

All Implemented Interfaces:

ISquare

```
public class Oops
extends java.lang.Object
implements ISquare
```

The Go-to-Jail square of the board

Constructor Summary

Constructors

Constructor	Description
Oops()	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	execute(Player player)	Generate an effect to a player

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Oops

```
public Oops()
```

Method Detail

execute

public void execute(Player player)

Generate an effect to a player

Go-to-Jail Effect:

- The player immediately goes to Jail.

Specified by:

`execute` in interface `ISquare`

Parameters:

`player` - dest. player

Package hk.edu.polyu.comp.comp3211.monopoly.model.squares

Class Property

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.model.squares.Property

All Implemented Interfaces:

`ISquare`

```
public class Property
extends java.lang.Object
implements ISquare
```

The Property squares of the board

Constructor Summary

Constructors

Constructor	Description
Property(java.lang.String s, int p, int r)	Initialize a property with params:

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	**execute**(**Player** player)	Generate an effect to a player
java.lang.String	**getName**()	Get the name of the property
Player	**getOwner**()	Get the owner of the property
void	**setOwner**(**Player** owner)	Set the owner to a custom player

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Property

public Property(java.lang.String s,
int p,
int r)

Initialize a property with params:

Parameters:

`s` - name of the property

`p` - selling price of the property

`r` - rental price of the property

Method Detail

execute

public void execute(Player player)

Generate an effect to a player

Property Effect:

- if property is not owned by any player:

  the visiting player can buy the property at the selling price or do nothing;

- if property is owned by a player:

  then if visiting player is the owner, nothing happens, otherwise the visiting player must pay the owner a rent.

Specified by:

`execute` in interface `ISquare`

Parameters:

`player` - dest. player

getName

public java.lang.String getName()

Get the name of the property

Returns:

name of the property

getOwner

public Player getOwner()

Get the owner of the property

Returns:

owner of the property

setOwner

public void setOwner(Player owner)

Set the owner to a custom player

Parameters:

`owner` - dest. player

Package hk.edu.polyu.comp.comp3211.monopoly.model.squares

Class Tax

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.model.squares.Tax

All Implemented Interfaces:

`ISquare`

```
public class Tax
extends java.lang.Object
implements ISquare
```

The Income-Tax square of the board

Constructor Summary

Constructors

Constructor	Description
`Tax``()`	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`void`	**`execute`**(**`Player`** player)	Generate an effect to a player

Methods inherited from class java.lang.Object

`clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Tax

```
public Tax()
```

Method Detail

execute

public void execute(Player player)

Generate an effect to a player

Income-Tax Effect:

- the player pays 10% of the balance (in form 10*n) as tax.

Specified by:

`execute` in interface `ISquare`

Parameters:

`player` - dest. player

Package hk.edu.polyu.comp.comp3211.monopoly.view

Class Printer

java.lang.Object
    hk.edu.polyu.comp.comp3211.monopoly.view.Printer

```
public class Printer
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor	Description
`Printer(Board board)`	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`void`	`Print()`	Print out current game board

Methods inherited from class java.lang.Object

`clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Printer

public Printer(Board board)

Method Detail

Print

```
public void Print()
```

Print out current game board