**Project : RICH & RUN**

1. Instruction

This project is based on a monopoly game. By implementing and changing a little bit of a rule, we came up with this new-version monopoly, that’s called “RICH&RUN”

And the detail of the rule is:

- Each player will have a starting money of 5000$

- Each round every player will take a turn to roll a dice

- If a tile is a normal tile, if the player that stepping on that tile can buy a tile, they must force buy it

- Since we force buy every tile that player is capable of buying, if the player's money is lower than 0, it will continue to be 0.

- Each player will have a total of 10 turn to have the most assets (player's money and player's assets value)

- Special card will contains a card that will help you win the game!

- Each corner of the board is the special tile

- Island tile will restrain you for 1 round

- Plane tile will let you steal a random player's money in the game for 1000$

- Attack tile will give help you steal 750$ from every player in the game.

1. Implemented Detail

**2.1 Package game.object**

# public abstract class **Card:** This class represents a card that is contained in the game that can be use on player

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| - String name | represent the card’s name. |
| - int value | represent the card’s value (Each type of card’s value can be different thing) |
| # String type | represent the card’s value. |

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + Card(String name, int value) | Constructor method. Initializes with the following specifications   * set card’s name as name * set card’s value as value |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| *+ void use(Player player)* | This method is called when the card is being used by the user. Each type of card has a different action of this method |
| + void setName(String name) | - set the Card’s name to name |
| + void setValue(int value) | - set the card’s value to value |
| + String getName() | - return card’s name |
| + int getValue() | - return card’s value |
| + String getType() | - return card’s type |

# public class **Player:** This class represents a player in the game

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| - String name | represent player’s name |
| - int assets | represent player’s assets (all the tile that this player owns price \* 1.1) |
| - int money | represent player’s current money |
| - int position | represent player’s current position |
| - PlayerStage playerStage | represent player’s stage |
| + int island | represent how many turn that player has stuck at the island  initializing island as 0 |

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + Player(String name) | Constructor method. Initializes with the following specifications  - invoke **initPlayer()**  - set the player’s name as name  - invoke **GameBoard.addStaticPlayer()** with the argument : this player |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void initPlayer() | - set player’s starting money as 5000  - set player’s starting assets as 0  - set player’s starting position as 0  - set player’s PlayerStage as **PlayerStage.NORMAL** |
| + void move(int dice) | This method is called when player has to move in the board.  If player’s position + dice is less than 28  - set player’s position as position + dice  If not (means that player pass the start tile)  - set player’s position as  player’s position + dice -28  and add 1000 to player’s money |
| + void stepOnTile() | This method is called when player step on the tile  get tile instance from **GameBoard.getTileinPosition(Player)**  **-** set the tile’s current stepper to player  - add the player to **tile.allStepper**  - invoke **tile.land()** |
| + void setAssets(int asset) | set the player’s assets as asset |
| + void setMoney(int money) | set the player’s money as money |
| + void setPosition(int position) | set the player’s position as position |
| + void setPlayerStage(PlayerStage stage) | set the player’s PlayerStage to stage |
| + String getName() | get the player’s name |
| + int getAssets() | get the player’s assets |
| + int getPosition() | get the player’s position |
| + int getMoney() | get the player’s money |
| + PlayerStage getPlayerStage() | get the player’s PlayerStage |

# public abstract class **Tile**: This class represents a tile in the game. This is the base class for every tile that contains in the board.

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| # String tileType | represent tile’s type |
| # Player currentStepper | represent the player that is stepping on this tile in their turn |
| # ArrayList<Player> allStepper | represent all the player that are currently on this tile, no matter it’s their turn or not |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| *+ void land* | This is method is called when a player has stepped on the tile, each type will perform different action. |
| + void setCurrentStepper(Player stepper) | set tile’s current stepper |
| + Player getCurrentStepper() | return tile’s current stepper |
| + ArrayList<Player> getAllStepper() | return tile’s all stepper |

**2.2 Package tile.base**

# **2.2.1.** public class **CardTile** extends **Tile**: This class represents a tile that can give a people a card.

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| - Card randomed | represent a card that this tile is having |

*Constructor*

|  |  |
| --- | --- |
| ***Name*** | ***Description*** |
| + CardTile() | Constructor method. Initializes with the following specifications  - set tileType to “Card” |

*Method*

|  |  |
| --- | --- |
| ***Name*** | ***Description*** |
| + void land() | Get the card instance from **GameBoard.getRandomCardInDeck();**  - set randomed to the card instance  - invoke **Card.use()** on the tile’s current stepper |
| + Card getCardInTile() | ­­­return randomed |

# **2.2.2.** public abstract class **CornerTile** extends **Tile** : This class represent a tile that in the corner that will perform an special action when player has stepped on. This is a base class for every corner tile.

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| # String cornerTileType | represent a corner tile’s type |

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + CornerTile() | Constructor method. Initializes with the following specifications  - set corner tile’s type to “Corner” |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| *+ void land()* | This method is called when a player has stepped on the corner tile, each corner tile’s type will perform a different action |
| + String getCornerTileType() | return the corner tile’s type |

# **2.2.3.** public abstract class **NormalTile** extends **Tile**: This class represents a tile that can be bought and has a fee once bought.

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + String name | represent tile’s name |
| + int position | represent tile’s position |
| - Player owner | represent tile’s owner (once bought) |
| # int toBuy | represent a tile’s price to buy |
| # int toPay | represent a tile’s fee (one bought) |
| # String normalTileType | represent a normal tile’s type (bought or not) |

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + NormalTile (String name, int position) | Constructor method. Initializes with the following specifications |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| *+ void land()* | This method is called when a player has stepped on the normal tile. |
| + String getName() | return tile’s name |
| + String getNormalTileType() | return normal tile’s type |
| + int getToBuy() | return tile’s price (tobuy) |
| + int getToPay() | return tile’s fee (topay) |
| + Player getOwner() | return tile’s owner |
| + void setOwner(Player player) | set the tile’s owner as player |

# **2.2.4.** public class **StartTile** extends **Tile**: This class represents a tile that in the start (position = 0) that will give player 1000$ every time the player has stepped on.

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + StartTile () | Constructor method. Initializes with the following specifications.  - set the tile’s type as “Start” |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void land() | This method is called when a player has stepped on the start tile.  - plus the player’s money for 1500$ |

**2.3 Package tile.corner**

# **2.3.1.** public class **AttackTile** extends **CornerTile**: This class represent a tile that is a Attack tile that will let you steal every player’s money 750$ each .

# 

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + AttackTile() | Constructor method. Initializes with the following specifications  - set the corner tile’s type to “Attack” |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void land() | This method is called when a player has stepped on the Attack tile  - get the copy of instance of **GameBoard.allPlayers** by using **Object.clone()**  **-** remove the tile’s current stepper from the instance  - for every player in the instance, minus the player’s money for 750$ and add it to the current stepper |

# **2.3.2.** public class **IslandTile** extends **CornerTile**: This class represents an Island tile that will detain a stepper for 1 round.

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + IslandTile() | Constuctor method. Initializes with the following specifications  - set the tile’s corner type to “Island” |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void land() | This method is called when a player has stepped on the Island tile  - set the stepper’s PlayerStage to **PlayerStage.Island** |

# 2.3.3. public class **PlaneTile** extends **CornerTile**: This class represents a tile that will let you steal 1000$ from a random player.

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + PlaneTile() | Constuctor method. Initializes with the following specifications  - set the corner tile’s type as “Plane” |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void land() | This method is called when a player has stepped on the plane tile  - get the random player instance from GameBoard.getRandomPlayerInBoard();  - plus the current stepper money for 1000$  - minus the random player for 1000$ |

**2.4 Package tile.normal**

# **2.4.1.** public class **Deed** extends **NormalTile**: This class represents a normal tile that hasn’t been bought yet.

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + Deed(String name, int position) | Constructor method. Initializes with the following specifications  - set the tile’s name as name  - set the tile’s position as position  - set normal tile’s type to “Deed”  - set tile’s price (toBuy) to its position \* 100  - set tile’s fee (toPay) to 0 |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + House upgradeTile() | This method is called when the tile has been bought and upgrade to the class “House”  - return house with the following argument  (this.name,this.position,this.toBuy \* 1.4, 99999, this.owner) |
| + void land() | This method is called when the player has stepped on the tile  if the stepper’s money has more than the tile’s price  - set the tile’s owner to the stepper  - minus the stepper’s money for the tile’s price  - plus the stepper’s assets for the tile’s price \* 1.1  if not  - set the stepper’s money to 0 |

# **2.4.2.** public class **House** extends **NormalTile** : This class represents a normal tile that has been bought.

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + House (String name, int position, int topay, int tobuy, Player owner) | Constructor method. Initializes with the following specifications  - set the tile’s name as name  - set the tile’s position as position  - set the tile’s owner as owner  - set the tile’s toPay as topay  - set the tile’s toBuy as tobuy  - set normal tile’s type to “House” |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void land() | This method is called when the player has stepped on the tile  if the stepper’s money has more than the tile’s fee (toPay)  - minus the stepper’s money for the tile’s fee  - plus the owner’s money for the tile’s fee  if not  - set the stepper’s money to 0 |

**2.5 Package card.derived**

# **2.5.1.** public class **MinusCard** extends **Card** : This card represents a card that will minus player’s money.

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + MinusCard(String name, int minus) | Constructor method. Initializes with the following specifications  In MinusCard, the value represents the money that will be minus on the player  - set the card’s name to name  - set the card’s value to minus  - set the card’s type to “Minus” |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void use(Player player) | This method is called when the card is being used on the player.  - minus the player’s money for the card’s value |

# **2.5.2.** public class **MoveCard** extends **Card** : This class represents a card that will move you to the corner tile

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + MoveCard(String name, int position) | Constructor method. Initializes with the following specifications  In MoveCard, the value field represents the position where to move to the player  - set the card’s name to name  - set the card’s type to “Move” |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void use(Player player) | This method is called when the card is being used on the player  if the card’s value minus player’s position is less than 0  - invoke player.move() with the argument of card’s value – player’s position + 28  if not  - invoke player.move() with the argument of card’s value – player’s position |

# **2.5.3.** public class **PlusCard** extends **Card**: This class represents a card that will plus a player’s money

*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + PlusCard(String name, int plus) | Constructor method. Initialize with the following specifications.  - set the card’s name to name  - set the card’s value to plus  - set the card’s type to “Plus” |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void use(Player player) | This method is called when the card is being used on the player.  - plus the player’s money for the card’s value |

**2.6 Package game.logic**

# **2.6.1.** public class **GameBoard**: This class represents a class that will collect all the element from a previous class and use it to create a board.

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + int round | This field represents the round in the game  - starting value as 1 |
| + int turn | This field represents the turn for each player in the game  - starting value as 0 |
| + ArrayList<Card> deck | This field represents a deck that contains all the card that will be used |
| + ArrayList<Tile> tiles | This field represents all the tiles in the board |
| + ArrayList<Player> allPlayers | This field contains a list of all the player in the game. |
| - String[] carreerList | This field contains a list of names for the normal tile. Use to create a tile. |
| + GameBoardStage gameBoardStage | This field represents that which stage is the board are being played. |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void initGame() | This method is called when the program is about to run  - invoke initDeck()  - invoke iniBoard()  - set the GameBoard.GameBoardStage to **GameBoardStage.INIT** |
| + void initTurn(Player player, int move) | This method is called when the player is about to start their turn  if player.PlayerStage is not **PlayerStage.ISLAND**  **-** invoke **player.move(move)**  - invoke **player.stepOnTile** |
| + void updatePlayerStage() | This method is called when the round ends and wants to see if any player has stuck in the island  -Check all the player in GameBoard.allPlayers  - if any of them has a player stage of PlayerStage.ISLAND  - check if player.island is equal to 1  if yes  - set the player’s stage to PlayerStage.NORMAL |
| + Card getRandomCardInDeck() | This method uses to get the random card in the deck |
| + void addStaticPlayer(Player player) | This method is uses to add the player to GameBoard.allPlayers |
| + void initDeck() | This method is uses to create a deck of card  - add 4 plus card in the value of (500,1000,1500,2000)  - add 4 minus card in the value of (500,1000,1500,2000)  - add 3 move card in the value of (7,14,21) |
| + int rollDice() | This method return a random integer number between 1-6 |
| + void initBoard() | This method is uses to create a list of tile in the board  - add 4 Card tile in the position of 4,9,18,23  - add 4 Corner tile in the position of 7,14,21,0  - add 20 Deed tile with the name in the field GameBoard.carreerList in the remaining position |
| + Tile getTileInPosition(Player player) | This method will return a tile that in the same position as player’s |
| + void getBoardCurrentRound() | This method is uses to printout to console which round are we playing |
| + Player getRandomPlayerInBoard() | This method will return a random player in the board. |

# **2.6.2.** public *enum* **GameBoardStage**: This class represents which stage that a board is in.

|  |  |
| --- | --- |
| **Name** | **Description** |
| - INIT | The stage that the board are starting |
| - ROLLING | The stage that the player is rolling the dice |
| - ACTION | The stage that the tile will perform the action |
| - ENDING | The stage that will be the transition in each round |

**2.7 Package player.logic**

# **2.7.1.** public *enum* **PlayerStage**: This class represents a stage that player are currently in

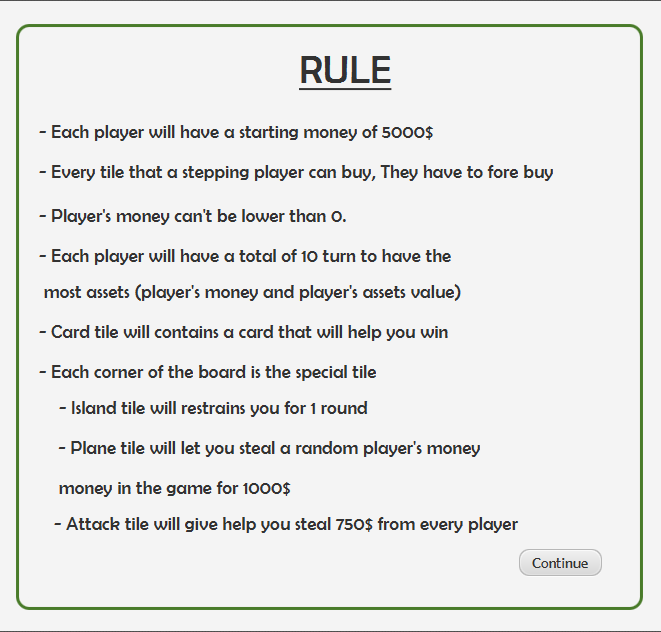
|  |  |
| --- | --- |
| **Name** | **Description** |
| - NORMAL | Normal stage that player are in |
| - ISLAND | A stage that the player are stepped on or waiting at Island tile |

**2.8 Package HomePageFXML**

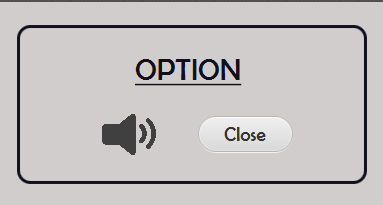
# **2.8.1. Home\_Page.fxml;**

****

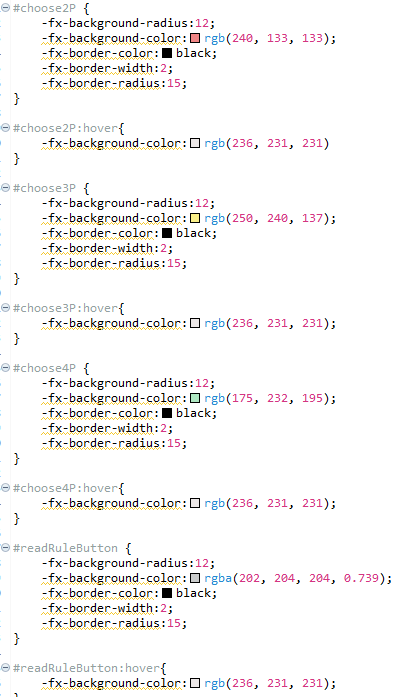
# **2.8.2. Option.fxml;**

****

# **2.8.3. ReadRule.fxml;**

****

# **2.8.4. Home\_Page.css;**

****

# **2.8.5. public class ControlHomePage implements Initializable;**

# : **This class is a controller for Home\_Page.fxml**

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| Button choose 2P | A button that will lead a user to EnterNamePage2P.fxml |
| Button choose3P | A button that will lead a user to EnterNamePage3P.fxml |
| Button choose4P | A button that will lead a user to EnterNamePage4P.fxml |
| Button exitOptionButton | A button that will exit the program |
| Button readRuleButton | A button that will show a rule of the game |
| Button continueButton | A button that will continue the game in the normal tile pop-up |
| ImageView logOutButton | An image that can be clicked on to exit the game |
| ImageView optionButton | An image that can be clicked on to see the option menu |
| ImageView soundButton | An image that can clicked on to play-paused the music |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void playMedia() | This method will init the media |
| + void goToEnterNamePage2P(ActionEvent event) throws IOException, InterruptedException | This method will change the scene to NamePage2P.fxml |
| + void goToEnterNamePage3P(ActionEvent event) throws IOException, InterruptedException | This method will change the scene to NamePage3P.fxml |
| + void goToEnterNamePage4P(ActionEvent event) throws IOException, InterruptedException | This method will change the scene to NamePage4P.fxml |
| + void goToPopUpReadRule(Action Event event) throws IOException | This method will show the rule pop-up |
| + void goToPopUpOption(Mouse Event event) throws IOException | This method will show the option menu pop-up |
| + void closeApp(Mouse Event event) throws IOException | This method will close the pop-up |
| + void closePopupOption(ActionEvent event) throws IOException | This method will close the option menu pop-up |
| + void closePopupRule(ActionEvent event) throws IOException | This method will close the rule pop-up |
| + void isSongRunning(MouseEvent event) throws IOException | This method will play-paused the sound when clicked on |

**2.9 Package NamePageFXML**

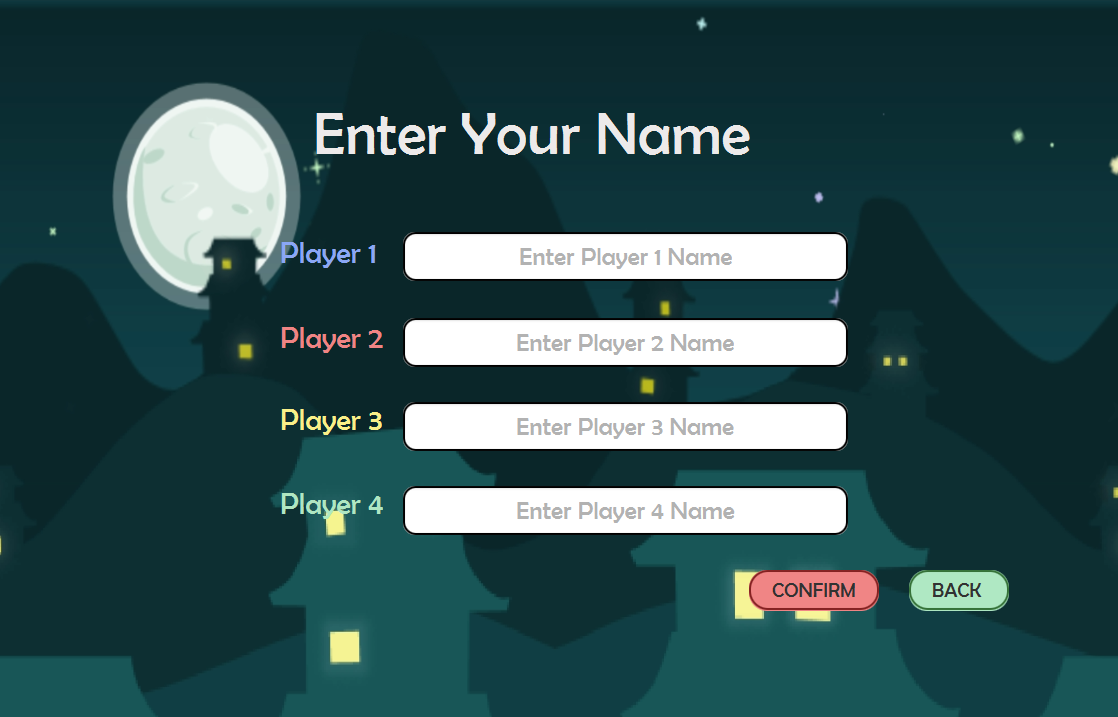
# **2.9.1. EnterName2P.fxml;**

# Graphical user interface Description automatically generated

# **2.9.2. EnterName3P.fxml;**

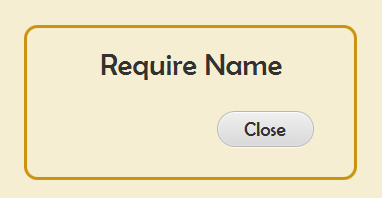
****

# **2.9.3. EnterName4P.fxml;**



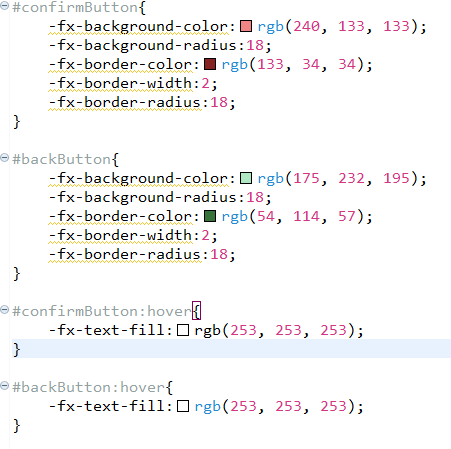
# **2.9.4. requireTextField.fxml;**

# : **This class is**

****

# **2.9.5. EnterName.css;**

# : **This class is**

****

# **2.9.6. public class ControlName implements Initalizable.;**

# : **This class is a controller for EnterName2P.fxml ,EnterName3P.fxml,EnterName4P.fxml**

*Field*

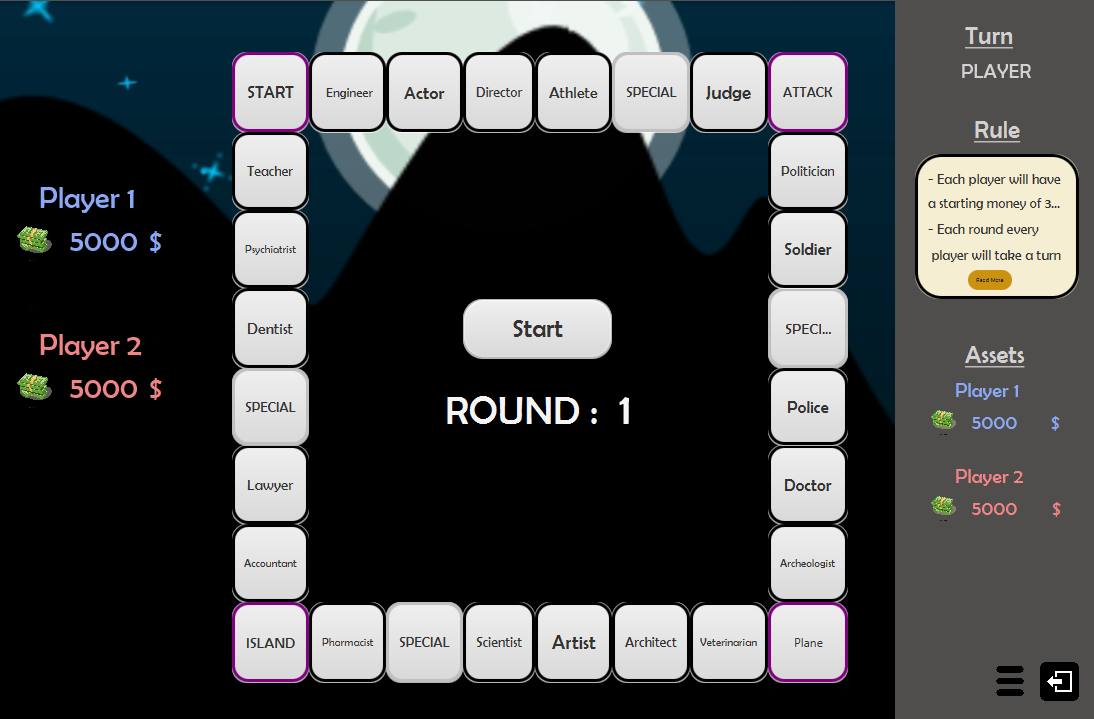
|  |  |
| --- | --- |
| **Name** | **Description** |
| TextField player[x]name ; x=1,2,3,4 | This field represents player’s name |
| Button backButton | The button that go back to HomePage |
| Button confirmButton | The button that go to GamePage |
| Button closeButton | The button that close require name pop-up |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + boolean checkPlayerName() | This method that check player textfield |
| + void goToHomePage(ActionEvent event) throws IOException, InterruptedException | This method that change scene to HomePage |
| + void closePopUpOption(ActionEvent event) throws IOException | This method that close require name pop-up |
| + void goToGamePage(ActionEvent event) throws IOException, InterruptedException | This method that change scene to GamePage |
| + void getPlayer() | This method will check how many player has been inputted and use a player’s constructor to add all the player to GameBoard.allPlayer |

**2.10 Package GamePageFXML**

# **2.10.1. GamePage2P.fxml ;**

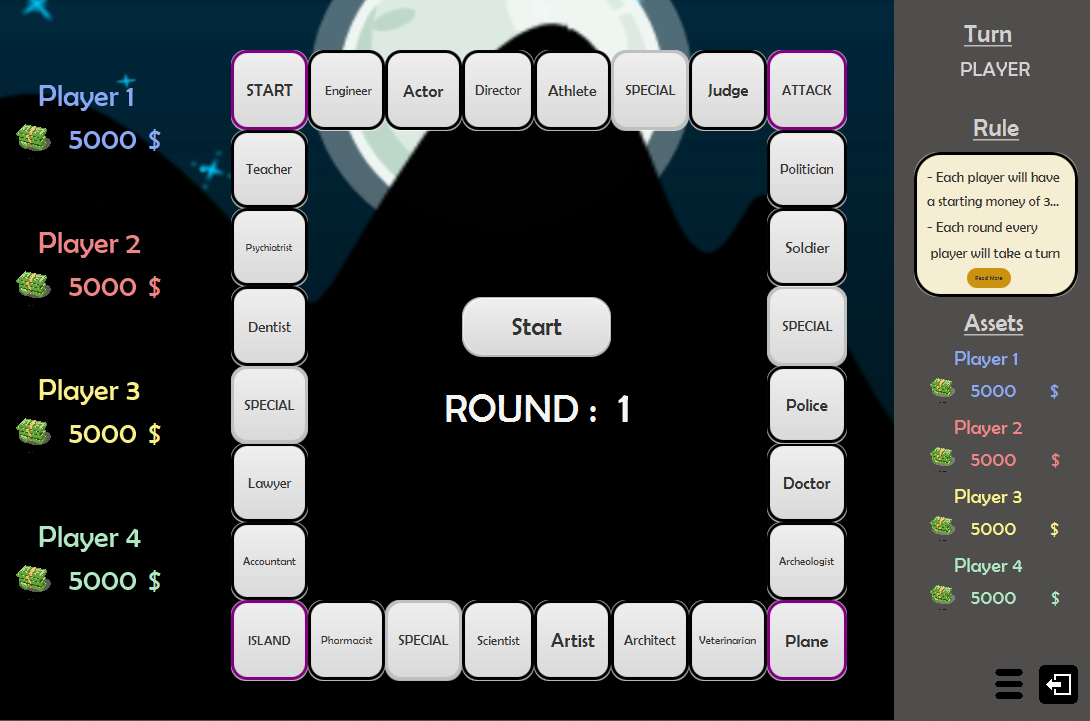
****

# **2.10.2. GamePage3P.fxml ;**

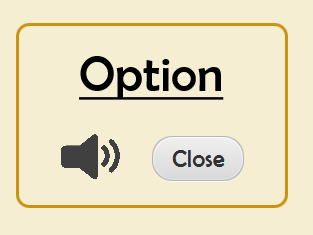
**A screenshot of a computer

Description automatically generated with medium confidence**

# **2.10.3. GamePage4P.fxml ;**

****

# **2.10.4. Option.fxml ;**

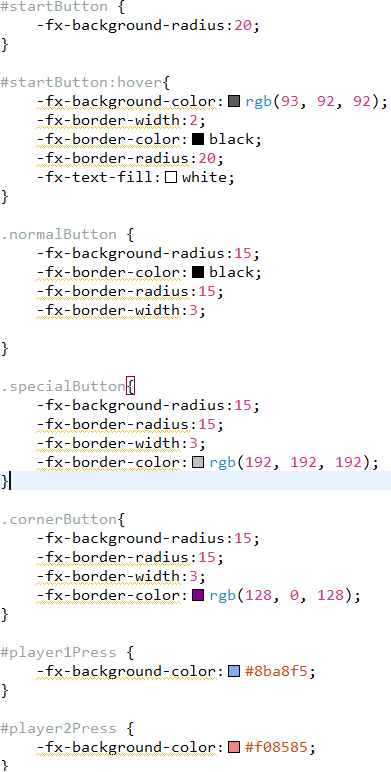
****

# **2.10.5. EndGamePage.fxml ;**

# 

****

# **2.10.6. Gamepage.css ;**

****

# **2.10.7.** public class **ControlGamePage** implements **Initializable**: This class is a controller for the any Gamepage.fxml

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| Button startButton | A button that will start the game |
| Button closeOptionButton | A button that close the option pop-up |
| Button readRuleButton | A button that will open the rule |
| Button continueButton | A button that shows in the normal tile’s information. Click this button to continue the game. |
| Button tile(x) : x = 1,2,3,4…27; | A button that represents a tile |
| Text player(x)Name : x = 1,2,3,4; | A text that shows player’s name |
| Text player(x)money : x = 1,2,3,4; | A text that shows player’s money |
| Text player(x)assets : x = 1,2,3,4; | A text that shows player’s assets |
| Text playerInTurn | A text that shows whose turn is playing |
| Text round | A text that shows how many round have passed |
| ImageView optionButton | An image that can clicked to open the option menu |
| ImageView exitButton | An image that can clicked to exit the option menu |
| ImageView soundButton | An image that can clicked to stop the music |
| + int actualPosition | This field represents an actual position if the player’s have step in to the card tile and got an move card |
| - ArrayList<Button> allTile | This field contains every button that represents a tile in the board |
| - ArrayList<String> allColor | This field contains a color that will change the tile’s color when a player has stepped on |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void closeDice(ActionEvent event) throws IOException | This method will close the dice pop-up |
| + void isSongRunning(MouseEvent event) throws IOException | This method will pause-play the music when clicked on |
| + void gotoOptionPopUp(MouseEvent event) throws IOException | This method will open the option pop-up |
| + void goToMainPage (MouseEvent event) throws IOException | This method will back to HomePage |
| + void goToPopUpReadRule (ActionEvent event) throws IOException | This method will open read rule pop-up |
| + void closePopUpOption(ActionEvent event) throws IOException | This method will close the read rule pop-up |
| + void updateName() | This method will set your name in the screen |
| + void updateMoneyAndAssets() | This method will set your money and assets in the screen |
| + int popUpDice() throws IOException | This method will open the dice pop-up |
| + void popAction(String name) throws IOException | This method will open the normal card tile pop-up |
| + void popCard(String type, int value) throws IOException | This method will open the card tile |
| + void popIsland() throws IOException | This method will open the Island card pop-up |
| + void popPlane() throws IOException | This method will open the Plane card pop-up |
| + void popAttack() throws IOException | This method will open the attack card pop-up |
| + void popEndGame() throws IOException | This method will open the end game pop-up |
| + void initColor() | This method will add all the color that has to be uses in this method in to .allColor |
| + void initTile() | This method will add all the button that represents the tile to the .allTile |
| + void clearColor(int before, Tile tile) | This method is called when a player’s leave a tile and then determine there is another stepper on that tile or not  if there are multiple stepper  - set the color of that tile to the player that is the second stepper after the player that has leaved  if there is not  - set the tile’s color to default |
| + void updateRound() | This method will update round and the information such as turn and round |
| + void startGame(ActionEvent event) throws IOException | This method is the main method of running the game, the progress of this method will check which stage that GameBoard are in, and determine what to do  if GameBoard are in **GameBoardStage.INIT**  (The first stage of starting the game)  - invoke initColor()  - invoke initTile()  - invoke updateName()  - invoke updateBoard()  - invoke updateMoneyAndAssets()  - set GameBoard stage to **GameBoardStage.ROLLING**  if GameBoard are in **GameBoardStage.ROLLING**  (After finished initializing all the important field, the player has to roll a dice and move)  - set actualposition to 0  - get instance(c) from popUpdice()  - get player that are currently playing from GameBoard.allPlayers.get(GameBoard.turn)  [ in case there are multiple stepper in a previous tile of player  - invoke clearColor();  ]  - invoke GameBoard.initTurn(player,c)  - StartButton.setText(“ACTION”)  - set GameBoard stage to **GameBoardStage.ACTION**  if GameBoard are in **GameBoardStage.ACTION**  - invoke the pop-up method of the tile that gets from **GameBoard.getTileInPosition(Player)** and determine which pop-up should invoke according to the tile.type()  - set GameBoard stage to **GameBoardStage.ENDING**  if GameBoard are in **GameBoardStage.ENDING**  - if the round is less than 10 then plus 1 to **GameBoard**.**turn**  - if the player that is playing are the last one in the game then plus 1 to **GameBoard.round**  - check for the player in island, if there is, plus 1 to player.island  - if the round is more than 10  - invoke popEndGame() |

# **2.10.8. public class ControlEndGame implements Initializable: This class represents a controller for the EndGamePage.fxml**

*Field*

|  |  |
| --- | --- |
|  |  |
| **Name** | **Description** |  |
| Text player[x]Name ; x = 1,2,3,4 | This text represents player[x]’s name |
| Text player[x]Money ; x = 1,2,3,4 | This text represents player[x]’s money + player[x]’s assets |

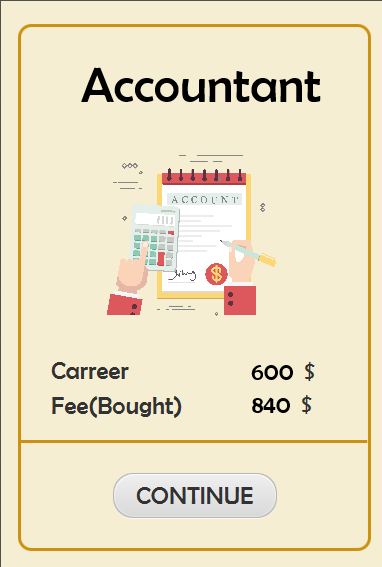
*Constructor*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void initialize(URL arg0, ResourceBundle arg1) | Constructor method. Initializes with the following specifications  - initializes all name with the name of the player’s in **GameBoard.allPlayer**  - initializes all money with the player.getMoney + player.getAssets of the player in **GameBoard.allPlayer** |

**2.11 Package PopGameFXML**

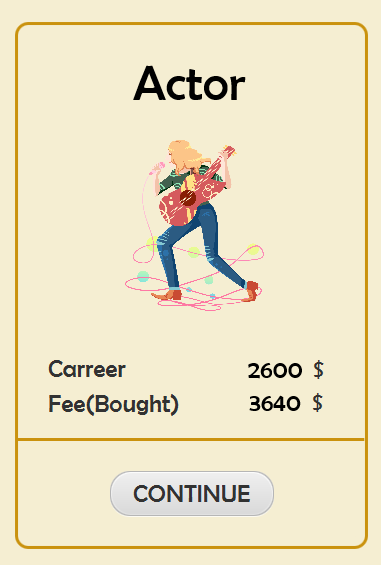
# **2.11.1. Accoutance.fxml**

# **: This class represents accoutance pop-up;**

****

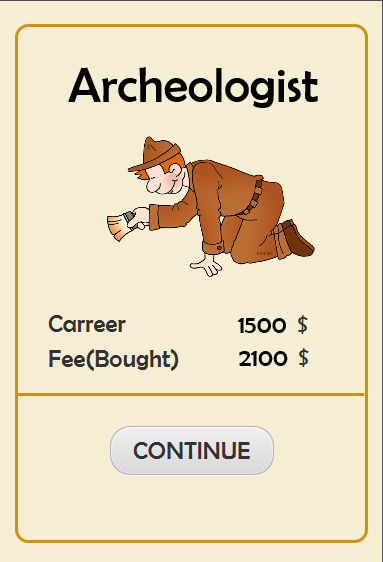
# **2.11.2. Actor.fxml**

# **: This class represents actor pop-up;**

****

# **2.11.3. Archeologist.fxml**

# **: This class represents archeologist pop-up;**

****

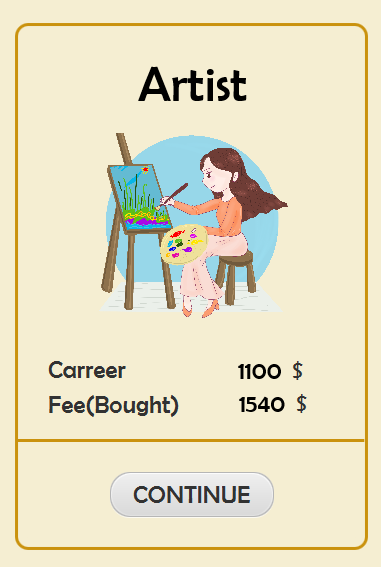
# **2.11.4. Arhitect.fxml**

# **: This class represents archeologist pop-up;**

****

# **2.11.5. Artist.fxml**

# **: This class represents artist pop-up;**

****

# **2.11.6. Artist.fxml**

# **: This class represents artist pop-up;**

****

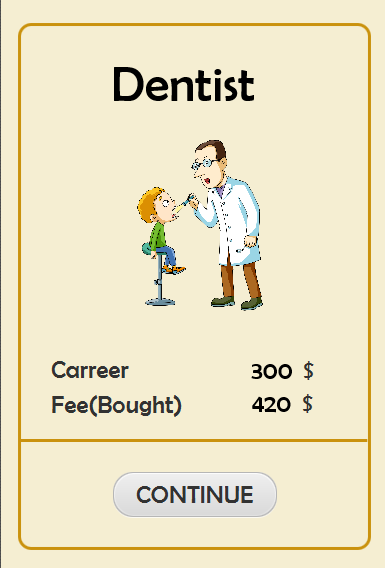
# **2.11.7. Attack.fxml**

# **: This class represents attack pop-up;**

****

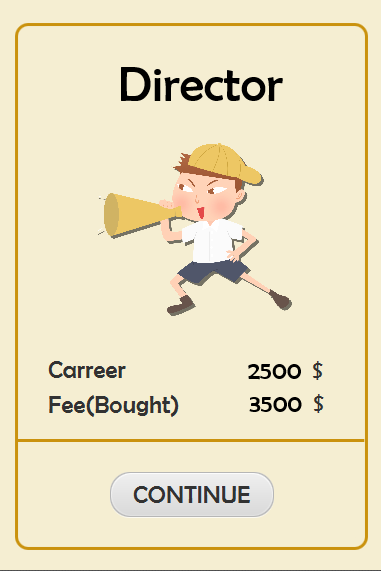
# **2.11.8. Dentist.fxml**

# **: This class represents attack pop-up;**

****

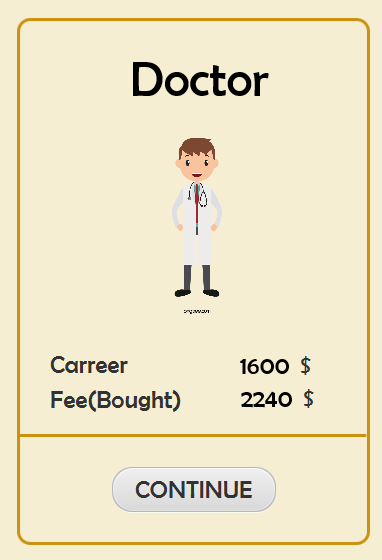
# **2.11.9. Director.fxml**

# **: This class represents director pop-up;**

****

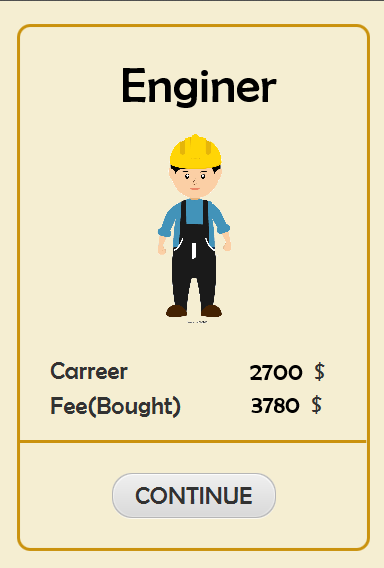
# **2.11.9. Doctor.fxml**

# **: This class represents doctor pop-up;**

****

# **2.11.10. Engineer.fxml**

# **: This class represents engineer pop-up;**

****

# **2.11.11. Island.fxml**

# **: This class represents engineer pop-up;**

****

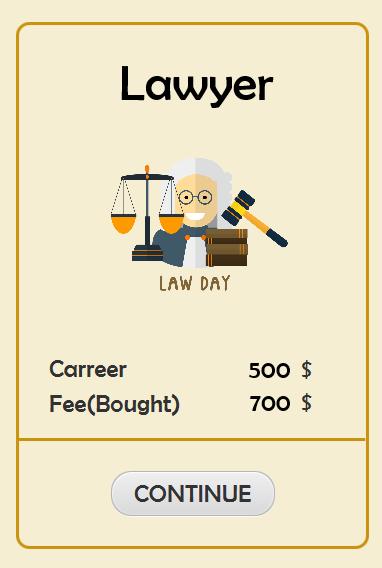
# **2.11.11. Judge.fxml**

# **: This class represents judge pop-up;**

****

# **2.11.12. Lawyer.fxml**

# **: This class represents lawyer pop-up;**

****

# **2.11.13. Move14.fxml**

# **: This class represents “go to plane” pop-up;**

****

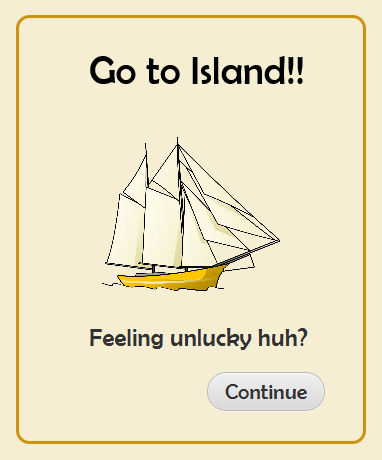
# **2.11.14. Move21.fxml**

# **: This class represents “go to supertile” pop-up;**

****

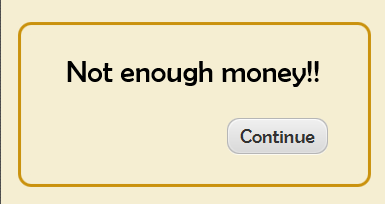
# **2.11.15. Move7.fxml**

# **: This class represents “go to island” pop-up;**

****

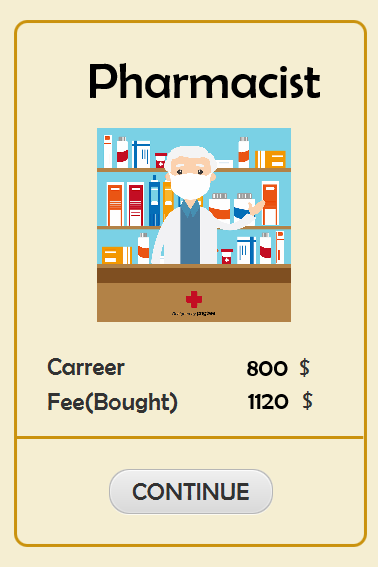
# **2.11.16. NoMoney.fxml**

# **: This class represents no enough money pop-up;**

****

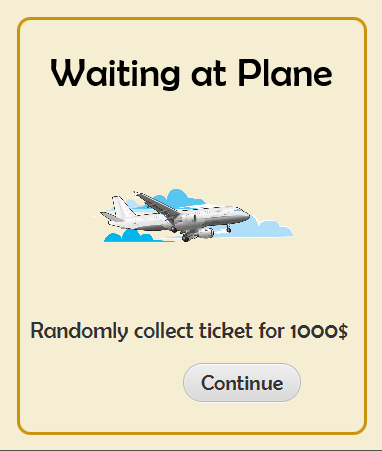
# **2.11.17. Pharmacist.fxml**

# **: This class represents Pharmacist pop-up;**

****

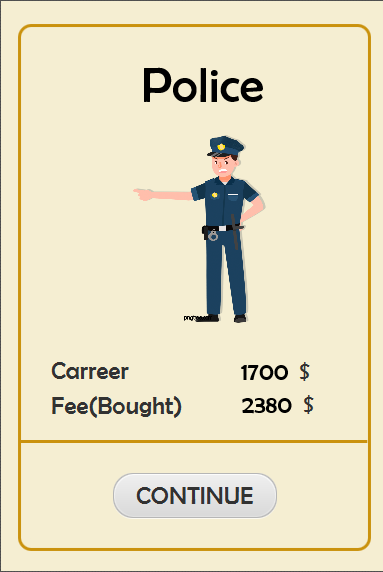
# **2.11.18. Plane.fxml**

# **: This class represents plane pop-up;**

****

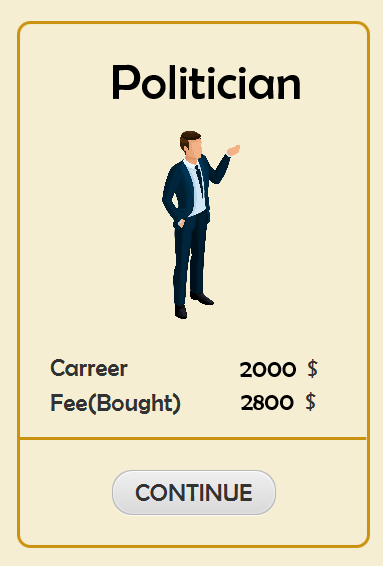
# **2.11.19. Police.fxml**

# **: This class represents police pop-up;**



# **2.11.20. Politician.fxml**

# **: This class represents politician pop-up;**

****

# **2.11.21. Phychiatrist.fxml**

# **: This class representsphychiatrist pop-up;**

****

**2.12 Package application**

**2.12.1** main:

*Field*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + Stage stage | The stage that show the game |
| + Scene scene | The instance that add to the stage |
| + Parent root | The instance that get scenebuilder |
| - File directory | The source of file |
| - File[] files; | Change directory to list file |
| - ArrayList<File> songs | The ArrayList of songs |
| - int songNumber | The index in ArrayList<File>songs |
| - Media media | Get ArrayList song to use in MediaPlayer |
| + MediaPlayer mediaPlayer | The instance that use to play songs |
| + Boolean running | The instance that checks the music is running or not |

*Method*

|  |  |
| --- | --- |
| **Name** | **Description** |
| + start(Stage primaryStage) throws Exception | Upload scene in scenebuilder and show in Stage And use addMusic() |
| + void addMusic() | The function that addmusic to mediaPlayer and play music |