# TongRod99 Documentation

Created by

Tittaya Worawongtad 6431314821

Naphat Wareesawetsuwan 6431315421

2110215 Programming Methodology

Semester 2 Year 2020

Chulalongkorn University

## Missing: Hide and Seek

### Introduction

Missing: Solitary Hide and Seek is inspired by an RPG named “HideAndSeek [The story of Dorothy]” and other series of Big Fish Games’ casual hidden object adventure games.

The game begins with Anna, the girl who lost her own memory, trying to escape from the room she is in. On the way she finds her lovely doll which hides in each room, she can notice that everything is curiouser and curiouser.

### Rules

The player uses keycode to move, observe and use an object. The goal of each room is to find a way to open the door in the limited time(3 minutes for each room).

### Example

### Main Menu Scene

Main menu is the first page of the game which including Start button, Instruction button, Credit button, and Exit button.


Figure 1: Main Menu

- When the mouse enter the button, it will have shadow effect and when you click the button, sound effect will be played.


Figure 2: button when mouse entered


Figure 3: button when mouse exited

- You can click **Instruction button** which show you how to play this game, **Credit button** which show the group's name who making this game and **Exit button** to exit the game. If you are click in the same button, it will close that pane and back to the first main menu.


Figure 4: when clicking the Instruction button


Figure 5: when clicking the Instruction button again


Figure 6: when clicking the Credit button


Figure 7: when clicking the Credit button again

Playing Screen Scene


Figure 8: Playing Screen Scene

After you click the **Start button**, it will bring you to the Playing Screen, but you can click **Menu button** to go back to Main Menu. When the game starts, according to instruction's pane, you can control your player's direction by pressing "A" (Left), "W" (Up), "D" (Right) or "S" (Down) and for interact, pressing "K" (to observe things) and "L" (to use the item).

Left

Up

Right

Down

Figure 9: Character face's direction

Bedroom

Living room

Library

Garden

Figure 10: Rooms

There are 4 rooms: **Bedroom**, **Living room**, **Library**, **Garden**. Each room has only 3 minutes to find a way out by use the items to unlock the door, and as time passes, the room will become darker.


Figure 11: Normal room


Figure 12: When the room getting darker

- For character's emotion has: NORMAL, WORRIED, SHOCK that interact with the game's story.

Normal

WORRIED

SHOCK

Figure 13: Character's emotion

There are two **items** which can be picked in this game, but they are used with different furniture. You can use a **pocket knife** to cut the sofa, a **key** to unlock the door or furniture.

Figure 14: Key

Figure 15: Pocket knife

- When Item is picked, it will be appeared in the itemInHandBox.

Figure 16: Key in itemInHandBox

Figure 17: Pocket knife in itemInHandBox

- Note: All items can only be used with their matched furniture.

- And the **note** which contains the hidden message that will show in Dialogue Pane. The notes are either in the furniture ( table etc.) or hide with the dolls.

Figure 18: Table with note

Figure 19: Dolls

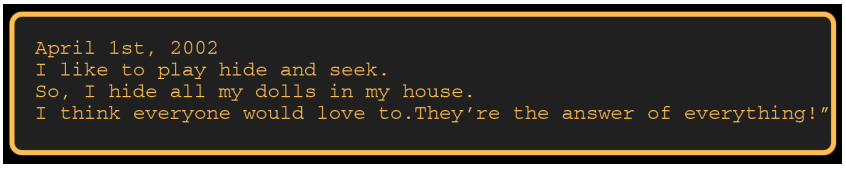



Figure 20: Dialogue Pane

The player must pick the items and use it with the correct furniture as mentioned below.

- The furniture which can be cut by a pocket knife: Sofa

Figure 21: Sofa

- The furniture which can be unlock and opened by a key: Cupboard, Safe

Figure 22: Cupboard

Figure 23: Safe

- The furniture which can contain the items: Bookshelf, Cupboard, TableWithNote, Safe, Mirror

Figure 24: Bookshelf

Figure 25: Mirror

Figure 26: IvyPic

Figure 27: ShieldPic

- The furniture in the rooms that can only show the message in dialoguePane.

Figure 28: TableWithLamp

Figure 29: Bed

Figure 30: Chair

Figure 31: FamilyPic

Figure 32: LongTableWithLamp

Figure 33: MysteryBox

Figure 34: Window

When the player gets a key that can be used to open the door, the screen fades, and changes to the next room until finish the garden scene or time out, it will move to the ending screen

Figure 35: Door

## Ending Screen Scene

There have 2 different screen which depends on the player's condition(win or lose).

Lose condition

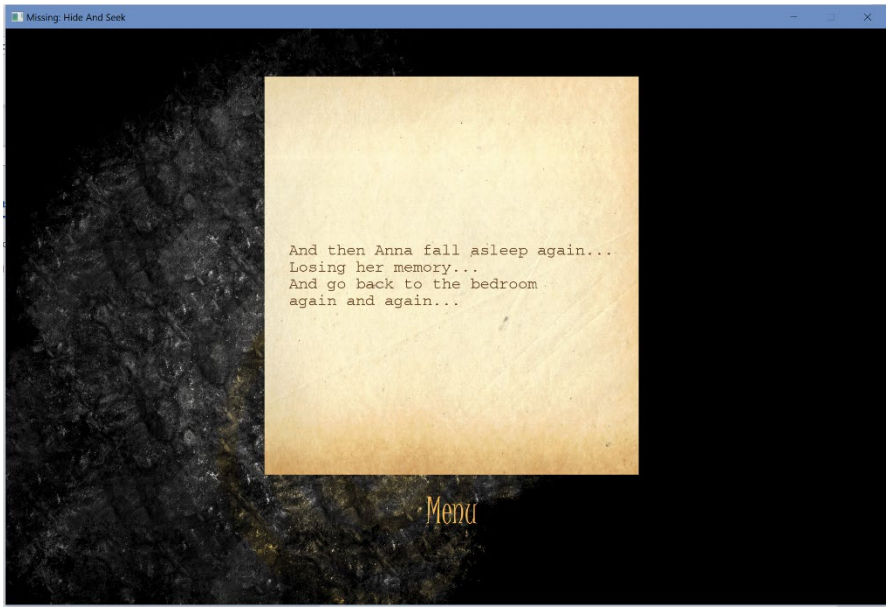



Figure 36: Game's over screen

- If the player cannot unlock all the door before the time is over, it will bring the player to the game over's screen. Player can click Menu button for go back to Main Menu.

Win condition

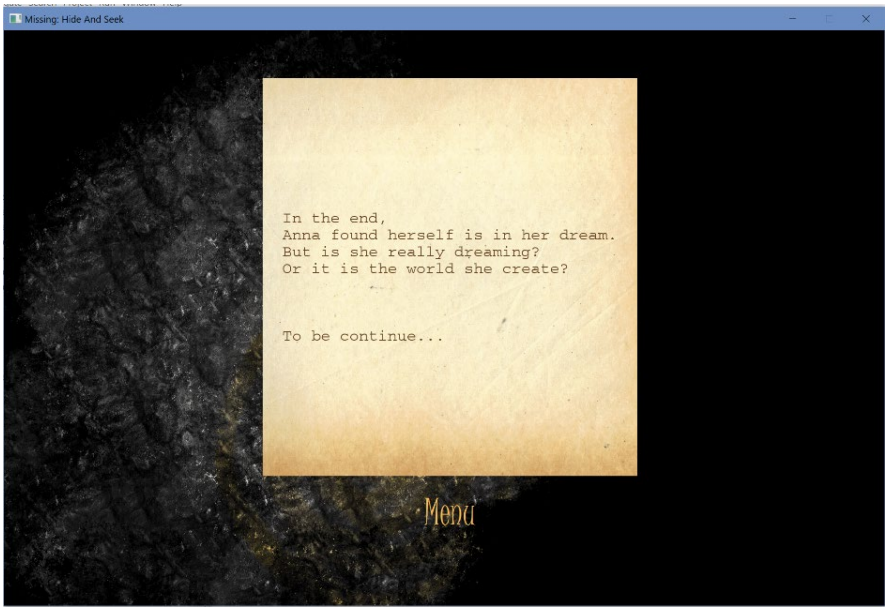
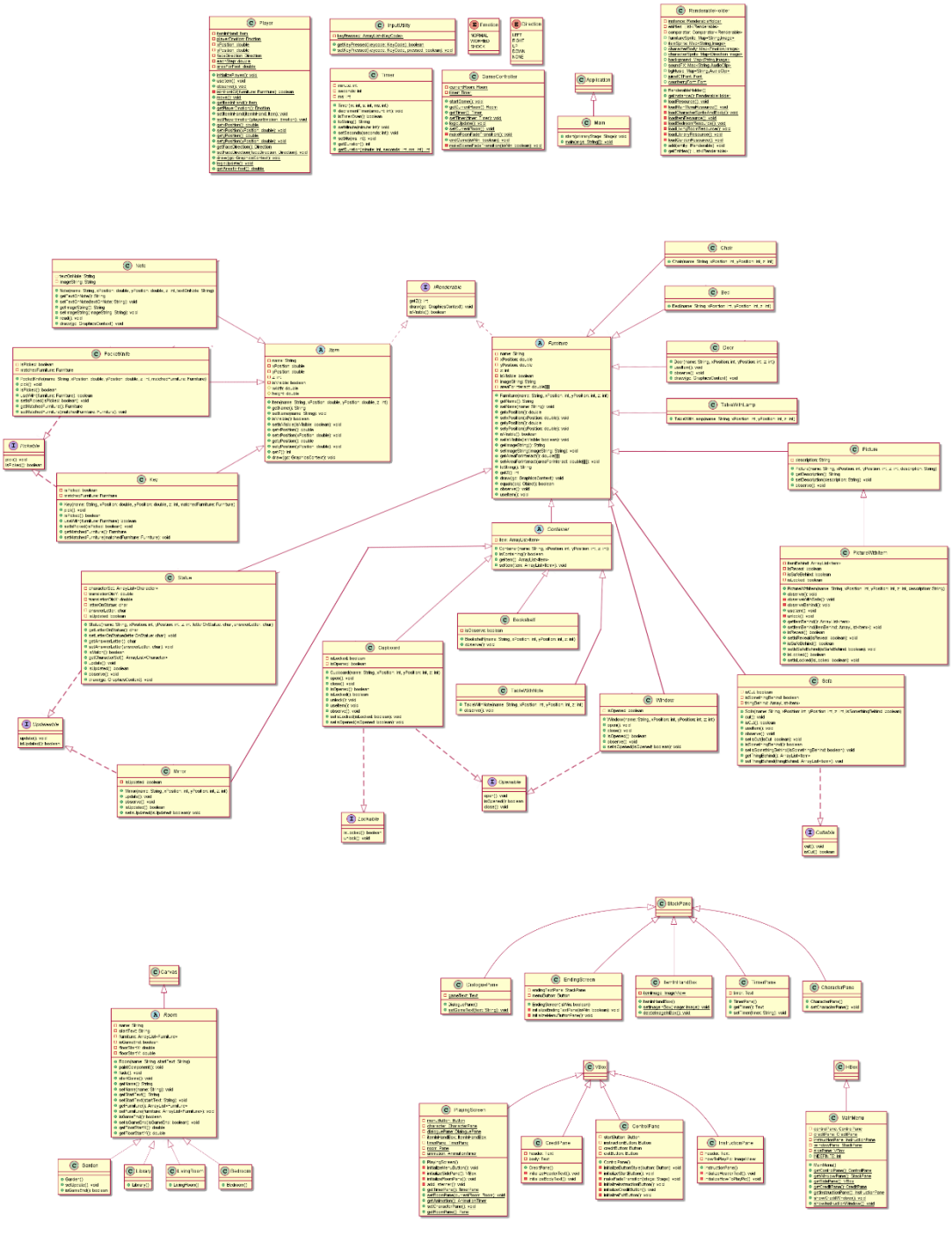



Figure 37: End Game's screen

- If you can unlock all the door before time's over, it will bring the player to the end game's screen. Player can click Menu button for go back to Main Menu.

Class diagram



1. package application

1.1 class Main extends Application

This class contains the main method. It is an entry point of the application.

Method

| Name | Description |
|---|---|
| + void start(Stage primaryStage) throws Exception | - launch up the new StartScreen of the game<br>- Show the primaryStage |
| + void main(String[] args) | The entry point of the application |

2. package entity

2.1 entity.base

2.1.1 Interface Cuttable

This interface defines methods for furniture that can be cut.

Method

| Name | Description |
|---|---|
| + boolean isCut() | Return **true** if the furniture got cut, return **false** otherwise. |
| + void cut() | This method will be called when the furniture got cut. |

### 2.1.2 Interface Lockable

This interface defines methods for furniture that can be locked.

Method

| Name | Description |
|---|---|
| + boolean isLocked() | Return **true** if the furniture got locked, return **false** otherwise. |
| + void unlock() | This method will be called when the furniture got unlocked. |

### 2.1.3 Interface Openable

This interface defines methods for furniture that can be opened.

Method

| Name | Description |
|---|---|
| + boolean isOpened() | Return **true** if the furniture got opened, return **false** otherwise. |
| + void open() | This method will be called when the furniture got opened. |
| + void close() | This method will be called when the furniture got closed. |

### 2.1.4 Interface Pickable

This interface defines methods for item that can be picked.

Method

| Name | Description |
|---|---|
| + boolean isPicked() | Return **true** if the item got picked, return **false** otherwise. |
| + void pick() | This method will be called when the item got picked. |

### 2.1.5 Interface Updateable

This interface defines methods for furniture that can be updated.

Method

| Name | Description |
|---|---|
| + boolean isUpdated() | Return **true** if the furniture got updated, return **false** otherwise. |
| + void update() | This method will be called when the furniture got updated. |

### 2.1.6 Enum Direction

This class represents player's move direction. It contains the following values:

LEFT, RIGHT, UP, DOWN, NONE.

### 2.1.7 Enum Emotion

This class represents character's emotion. It contains the following values: NORMAL, WORRIED and SHOCK.

### 2.1.8 *Abstract* class Container extends Furniture

This class represents the furniture that can contain something.

Field

| Name | Description |
|---|---|
| - ArrayList<Item> item | The list of items in the container |

Constructor

| Name | Description |
|---|---|
| + Container(String name, int xPosition, int yPosition, int z) | - Initialize the container fields with respective values.<br>- Initialize item as an empty ArrayList |

Method

| Name | Description |
|---|---|
| + boolean isContaining() | Return true if item is not equal to 0, return false otherwise |
| + getter and setter for each field | |

### 2.1.9 *Abstract* class Furniture implements IRenderable

This class represents the furniture that is used in the room.

Field

| Name | Description |
|---|---|
| - String name | Name of the furniture which will be displayed and use to identify the furniture. |
| - double xPosition | Position of the furniture in X-axis |
| - double yPositon | Position of the furniture in Y-axis |
| - int z | The number which related to the order of rendering image on the screen. |
| - boolean isVisible | State that the furniture is still visible or not |
| - String imageString | The name of the furniture's image |
| - double[][] areaForInteract | The furniture's scale specify area for player to interact |

Constructor

| Name | Description |
|---|---|
| + Furniture(String name, int xPosition, int yPosition, int z) | - Initialize the Furniture fields with respective values.<br>- Set isVisible as true by default<br>- Set imageString as name<br>- Setup areaForInteract |

Method

| Name | Description |
|---|---|
| + void draw(GraphicsContext gc) | Draw the furniture on its current position. |
| + void observe() | The method will be called when player observe the furniture by default.<br>- Set gameText by using toString() |
| + void useItem() | This method will represent the default text which show that this item cannot use with this furniture.<br>- set gameText as “I think it’s better to use it somewhere.” |
| + int getZ() | Return z |
| + boolean equals(Object obj) | This method which is check equality of two furniture. |
| + boolean isVisible() | Return **true** if the furniture is visible, return **false** otherwise. |
| + String toString() | Returns a formatted string in the format of “This is a normal + <name>.” |
| + getter and setter for each field | |

### 2.1.10 *Abstract* Class Item implements IRenderable

This class represents the item which used in the room

Field

| Name | Description |
|---|---|
| - String name | Name of the item which will be displayed and use to identify the item |
| - double xPosition | Position of the item in X-axis |
| - double yPosition | Position of the item in Y-axis |
| - int z | The number which related to the order of rendering image on the screen. |
| - boolean isVisible | Keeps track if the item has been visible or not |
| # final double width | The width of the item's image<br>- Initialize it to 20 |
| # final double height | The height of the item's image<br>- Initialize it to 20 |

Constructor

| Name | Description |
|---|---|
| + Item(String name, double xPosition, double yPosition, int z) | - Initialize the item fields with respective values.<br>- Set visible as false by default. |

Method

| Name | Description |
|---|---|
| + boolean isVisible() | Return **true** if the furniture is visible, return **false** otherwise. |
| + int getZ() | Return z |
| + void draw(GraphicsContext gc) | Draw the item on its current position. |
| + getter and setter for each field | |

### 2.1.11 *Abstract* class Room extends Canvas

This class represents room's map which appears in the playing screen.

Field

| Name | Description |
|---|---|
| - String name | Name of the room which will be displayed and use to identify the room |
| - String startText | The message that appears when the room is start in dialoguePane |
| - ArrayList<Furniture> furniture | The list of furniture in the room |
| - boolean isGameEnd | State that the game is end or not |
| - final double floorStartX | To identify where the floor starts in X-axis<br>- Initialize it to 0 |
| - final double floorStartY | To identify where the floor starts in Y-axis<br>- Initialize it to 140 |

Constructor

| Name | Description |
|---|---|
| + Room(String name, String startText ) | - Initialize the room fields with respective values.<br>- Set width t0 720 and height to 520<br>- Initialize furniture as an empty ArrayList<br>- It sets the game to not be ended by default<br>- Call startGame() method |

Method

| Name | Description |
|---|---|
| + void paintComponent() | - Draw all the components in the room and character<br>- Call fade() method |
| + void fade() | This method will make the room gradually fade out when the time passes by using graphicsContext |
| + void startGame() | Set gameText as startText |
| + getter and setter for each field | |

## 2.2 entity.furniture

### 2.2.1 Class Bed extends Furniture

This class represents a bed which is one of the furniture.

Constructor

| Name | Description |
|---|---|
| + Bed(String name, int xPosition, int yPosition, int z) | Initialize the bed fields with respective values. |

### 2.2.2 Class Bookshelf extends Container

This class represents a bookshelf which is one of the containers.

Field

| Name | Description |
|---|---|
| - boolean isObserve | State that the bookshelf got observed or not |

Constructor

| Name | Description |
|---|---|
| + Bookshelf(String name, int xPosition, int yPosition, int z) | - Initialize the bookshelf fields with respective values.<br>- It sets to not been observed by default. |

Method

| Name | Description |
|---|---|
| + void observe() | - If the bookshelf is not observed, set isObserve as true and visible all items in bookshelf<br>- if not and it has item in bookshelf, check all items<br>- If it is a note, call read() method<br>- If it is a pocket knife or key, call pick() method and remove it<br>- Otherwise, set gameText with “There is a lot of books here, I would love to read if I have time.” |

### 2.2.3 Class Chair extends Furniture

This class represents a chair which is one of the furniture.

Constructor

| Name | Description |
|---|---|
| + Chair(String name, int xPosition, int yPosition, int z) | Initialize the chair fields with respective values. |

### 2.2.4 Class Cupboard extends Container implements Lockable, Openable

This class represents a cupboard which is one of the furniture that can be locked and opened.

Field

| Name | Description |
|---|---|
| - boolean isLocked | State that the cupboard got locked or not |
| - boolean isOpened | State that the cupboard got opened or not |

Constructor

| Name | Description |
|---|---|
| + Cupboard(String name, int xPosition, int yPosition, int z) | - Initialize the cupboard fields with respective values.<br>- It sets to locked and not been opened by default. |

Method

| Name | Description |
|---|---|
| + boolean isOpened() | Return **true** if the cupboard got opened, return **false** otherwise. |
| + void open() | - If the cupboard is locked, set gameText as “It’s locked.” and play sound effect.<br>- Otherwise, set isOpened as true, imageString as “OpenedCupboard”, play |

| | |
|---|---|
| | sound effect, and visible all items in cupboard |
| + void close() | - Set isOpened as false and imageString as this cupboard's name.<br>- Play sound effect<br>- Set all items' visible as false |
| + boolean isLocked() | Return **true** if the cupboard got locked, return **false** otherwise. |
| + void unlock() | - If the player's hand has a key and can use with this cupboard,<br>- Set isLocked as false<br>- Call open() method<br>- Otherwise, set gameText as "I think it doesn't fit here." |
| + void useItem() | - If the cupboard is locked, call unlock() method<br>- Otherwise, call useItem() method from super class |
| + void observe() | - If the cupboard is opened and item once observed, call close() method and return<br>- If it is closed, call open() method<br>- If the cupboard is locked, return<br>- If it has item in cupboard, check all items in cupboard<br>- If it is a note, call read() method<br>- If it is a pocket knife or key, call pick() method and remove it<br>- Otherwise, set gameText with "It's just an empty cupboard." |

| | |
|---|---|
| + getter and setter for each field | |

### 2.2.5 Class Mirror extends Container implements Updateable

This class represents a mirror which is one of the containers that can be updated.

Field

| Name | Description |
|---|---|
| - boolean isUpdated | State that the mirror got updated or not |

Constructor

| Name | Description |
|---|---|
| + Mirror(String name, int xPosition, int yPosition, int z) | - Initialize the mirror fields with respective values.<br>- Set to not been updated by default. |

Method

| Name | Description |
|---|---|
| + boolean isUpdated() | Return **true** if the mirror got updated, return **false** otherwise. |
| + void update() | - Set imageString as “BehindMirror”, play sound effect and visible all items in mirror<br>- Set isUpdated as true<br>- Set gameText as "My face... Oh, am I Anna? Why…? Why am I here?" |

| | |
|---|---|
| + void observe() | - If the mirror is not updated, call update() method<br>- If the mirror is updated and has items in mirror, check all items in mirror<br>- If it is a note, call read() method<br>- If it is a pocket knife or key, call pick() method and remove it<br>- Otherwise, set gameText as “It’s broken…” |
| + void setIsUpdated(boolean isUpdated) | - Set isUpdated as isUpdated<br>- If currentRoom is a Garden, call setUpdate() method from Garden class |

#### 2.2.6 Class Picture extends Furniture

This class represents a picture which is one of the furniture.

Field

| Name | Description |
|---|---|
| - String description | This represents the picture’s description which observed by the player |

Constructor

| Name | Description |
|---|---|
| + Picture(String name, int xPosition, int yPosition, int z, String description) | - Initialize the picture fields with respective values. |

Method

| Name | Description |
|---|---|
| + void observe() | Set gameText as picture's description |
| + getter and setter for each field | |

### 2.2.7 Class PictureWithItem extends Picture

This class represents a picture which has safe behind.

Field

| Name | Description |
|---|---|
| - ArrayList<Item> itemBehind | The list of items which be hidden behind the picture |
| - boolean isReveal | State that the safe behind picture got revealed or not. |
| - boolean isSafeBehind | State that is safe behind the picture or not. |
| - boolean isLocked | State that the safe behind picture got locked or not. |

Constructor

| Name | Description |
|---|---|
| + PictureWithItem(String name, int xPosition, int yPosition, int z, String description) | - Initialize the pictureWithItem fields with respective values.<br>- Initialize itemBehind as an empty ArrayList |

| | |
|---|---|
| | - Set isSafeBehind, isLocked and isReveal as false by default |

Method

| Name | Description |
|---|---|
| + void observe() | - If it has safe behind the picture, call observeWithSafe() method<br>- Otherwise, call observeBehind() method |
| - void observeWithSafe() | - If safe which behind the picture is not revealed,<br>- Set isReveal as true<br>- Set imageString as “ClosedSafe” and play sound effect<br>- Set gameText “There is a safe behind.” and play sound effect<br>- else if the safe is locked, set gameText as “It’s locked.”<br>- Otherwise, call observeBehind() method |
| - void observeBehind() | - If it has items in the picture, Check all the items<br>- If item is a note, call read() method<br>- If item is a pocket knife or key, call pick() method and remove it<br>- Otherwise, set gameText as “I never thought items can hide here” |
| + void useItem() | - If it has safe behind the picture which got revealed and locked, call unlock() method |

| | |
|---|---|
| | - Otherwise, call useItem() from super class |
| - void unlock() | - If the player's hand has a key and can use with this safe,<br>- Set isLocked as false<br>- Set imageString as "OpenedSafe"<br>- Visible all items in the safe.<br>- Otherwise, set gameText as "I think it doesn't fit here." |
| + getter and setter for each field | |

### 2.2.8 Class Sofa extends Furniture implements Cuttable

This class represents a sofa which is one of the furniture that can be cut.

Field

| Name | Description |
|---|---|
| - boolean isCut | State that the sofa got cut or not. |
| - boolean isSomethingBehind | State that it has something behind the sofa or not. |
| - ArrayList<Item> thingBehind | List of items which behind the sofa |

Constructor

| Name | Description |
|---|---|
| + Sofa(String name, int xPosition, int yPosition, int z, boolean isSomethingBehind) | - Initialize the sofa fields with respective values.<br>- Set to not been cut by default |

| | |
|---|---|
| | - Initialize thingBehind as an empty ArrayList |

Method

| Name | Description |
|---|---|
| + void cut() | - Set isCut as true<br>- Set imageString as “CutSofa” and play sound effect.<br>- Visible all items which behind the sofa |
| + boolean isCut() | Returns **true** if the sofa got cut, return **false** otherwise. |
| + void useItem() | - If there is nothing behind the sofa or sofa got cut, call useItem() method from super class<br>- Otherwise,<br>- If the player’s hand has a pocket knife and can use with this safe, call cut() method.<br>- if not, set gameText as “I think it doesn’t fit here.” |
| + void observe() | - If there is nothing behind the sofa, call observe() method from super class<br>- if not and sofa has not been cut, set gameText as “I think there is something under the sofa”<br>- If not and it has something behind the sofa, check all the items<br>- If it is a note, call read() method |

| | |
|---|---|
| | - If it is a pocket knife or key, call pick() method and remove it<br>- Otherwise, set gameText as “Little sorry, to make it’s torn.” |
| + getter and setter for each field | |

### 2.2.9 Class Statue extends Furniture implements Updateable

This class represents a statue which is one of the furniture that can be updated.

Field

| Name | Description |
|---|---|
| - final ArrayList<Character> characterSet | List that contains the character’s set on statue<br>- Initialize it with ArrayList<character>(Arrays.asList (‘M’, ‘H’, ‘I’, ‘P’, ‘E’, ‘N’, ‘A’, ‘S’, ‘O’)) |
| - final double translationDisY | Where the character’s set that appears on statue in Y-axis<br>- Initialize it to 50 |
| - final double translationDisX | Where the character’s set that appears on statue in X-axis<br>- Initialize it to 35 |
| - char letterOnStatue | The letter that show on the statue |
| - char answerLetter | The answer letter for this statue |
| - boolean isUpdated | State that the statue got updated or not |

Constructor

| Name | Description |
|---|---|
| + Statue(String name, int xPosition, int yPosition, char letterOnStatue, char answerLetter) | - Initialize the statue fields with respective values.<br>- Set to not been updated by default. |

Method

| Name | Description |
|---|---|
| + boolean isMatch() | Return **true** if letterOnStatue is equal to answerLetter, return **false** otherwise |
| + boolean isUpdated() | Return **true** if the statue got updated, return **false** otherwise. |
| + void update() | Set isUpdated as true |
| + void observe() | - If the statue is not been updated, call observe() method from super class<br>- Otherwise,<br>- Set new letterOnStatue by move to the next character from the characterSet<br>- Set gameText as "This statue is strange!\nI think this is the puzzle\n but how can I solve this." and play sound effect. |
| + void draw(GraphicsContext gc) | - Call draw(gc) method from super class<br>- draw the letter on the statue |
| + getter and setter for each field | |

### 2.2.10 Class TableWithLamp extends Furniture

This class represents a table with a lamp on top which is one of the furniture.

Constructor

| Name | Description |
|---|---|
| + TableWithLamp(String name, int xPosition, int yPosition, int z) | - Initialize the tableWithLamp fields with respective values. |

### 2.2.11 Class TableWithNote extends Container

This class represents a table with a note on top which is one of the containers.

Constructor

| Name | Description |
|---|---|
| + TableWithNote(String name, int xPosition, int yPosition, int z ) | - Initialize the tableWithNote fields with respective values. |

Method

| Name | Description |
|---|---|
| + void observe() | - If it has item, Check all items<br>- If it is a note, call read() method<br>- If it is a key or pocket knife, call pick() method and remove it<br>- Otherwise, call observe() method from super class |

### 2.2.12 Class **Window** extends Furniture implements Openable

This class represents a window which is one of the furniture that can be opened.

Field

| Name | Description |
|---|---|
| - boolean isOpened | State that the window got opened or not |

Constructor

| Name | Description |
|---|---|
| + Window(String name, int xPosition, int yPosition, int z) | - Initialize the window fields with respective values.<br>- Set to not been opened by default. |

Method

| Name | Description |
|---|---|
| + void open() | - Play sound effect<br>- Set imageString as “OpenedWindow”<br>- Set isOpened as true |
| + void close() | - Play sound effect<br>- Set imageString as this window’s name<br>- Set isOpened as false |
| + boolean isOpened() | It returns **true** if the widow got opened, returns **false** otherwise. |

| | |
|---|---|
| + void observe() | - If the window is not opened, call open() method. Otherwise, call close() method.<br>- Set gameText as “Why everything is silent?” |
| + getter and setter for each field | |

### 2.2.13 Class Door extends Furniture

This class represents a door which is one of the furniture.

Constructor

| Name | Description |
|---|---|
| + Door(String name, int xPosition, int yPosition, int z) | - Initialize the door fields with respective values. |

Method

| Name | Description |
|---|---|
| + void useItem() | - If item in player’s hand is a key and can use with this door, set isGameEnd as true<br>- Otherwise, set gameText as “I think it doesn’t fit here.” |
| + void observe() | Set gameText as “The door is locked! I must find the key" and play sound effect. |
| + void draw(GraphicsContext gc) | - draw the door |

### 2.3 entity.item

#### 2.3.1 Class **Key** extends Item implements Pickable

This class represents a key which is one of the items that can be picked.

Field

| Name | Description |
|---|---|
| - Furniture matchedFurniture | The furniture that matched with this key |
| - boolean isPicked | State that the key got picked or not |

Constructor

| Name | Description |
|---|---|
| + Key(String name, double xPosition, double yPosition, int z, Furniture matchedFurniture) | - Initialize the key fields with respective values.<br>- Set to not been picked by default. |

Method

| Name | Description |
|---|---|
| + boolean isPicked() | Return **true** if the key got picked, return **false** otherwise. |
| + void pick() | - Set the key's image in itemInHandBox<br>- Set itemInHand with this key<br>- Set isPicked as True<br>- Set isVisible from super class as false<br>- Play sound effect |

| | |
|---|---|
| + boolean useWith(Furniture furniture) | - If furniture is equal to the matchedFurniture,<br>- delete the key's image from itemInHandBox<br>- Set itemInHand as null<br>- Set isPicked as false<br>- Play sound effect<br>- return true<br>- Otherwise, return false |
| + getter and setter for each field | |

### 2.3.2 Class Note extends Item

This class represents a note which is one of the items.

Field

| Name | Description |
|---|---|
| - String textOnNote | The text on the note which appears in dialoguePane |
| - String imageString | The name of the note's image |

Constructor

| Name | Description |
|---|---|
| + Note(String name, double xPosition, double yPosition, int z, String textOnNote) | - Initialize the note fields with respective values.<br>- Set isVisible from super class as false<br>- Set imageString as "" |

Method

| Name | Description |
|---|---|
| + void read() | Set gameText as textOnNote |
| + void draw(GraphicsContext gc) | - If imageString is empty, return<br>- Otherwise, draw this imageString on its current position. |
| + getter and setter for each field | |

### 2.3.3 Class PocketKnife extends Item implements Pickable

This class represents a pocket knife which is one of the items that can be picked.

Field

| Name | Description |
|---|---|
| - Furniture matchedFurniture | The furniture that can be used with this pocket knife |
| - boolean isPicked | State that the pocket knife got picked or not |

Constructor

| Name | Description |
|---|---|
| + PocketKnife(String name, double xPosition, double yPosition, int z, Furniture matchedFurniture) | - Initialize the pocketKnife fields with respective values.<br>- Set to not been picked by default. |

Method

| Name | Description |
|---|---|
| + boolean isPicked() | Return **true** if the pocket knife got picked, return **false** otherwise. |
| + void pick() | - Set the pocket knife's image initemInHandBox<br>- Set itemInHand with this pocket knife<br>- Set isPicked as True<br>- Set isVisible from super class as false<br>- Play sound effect |
| + boolean useWith(Furniture furniture) | - If furniture is equal to the matchedFurniture,<br>- delete the pocket knife's image from itemInHandBox<br>- Set itemInHand as null<br>- Set isPicked as false<br>- return true<br>- Otherwise, return false |
| + getter and setter for each field | |

## 3. Package gui

### 3.1 gui.room

#### 3.1.1 Class **Bedroom** extends Room

This class represents a bedroom which is one of the rooms in this game.

Constructor

| Name | Description |
|---|---|
| + Bedroom() | - Initialize the bedroom fields<br>- Add and setup all furniture and items in this class |

#### 3.1.2 Class **Garden** extends Room

This class represents a garden which is one of the rooms in this game.

Constructor

| Name | Description |
|---|---|
| + Garden() | - Initialize the garden fields<br>- Set player's emotion as NORMAL<br>- Call setCharacterPane() method from Player class<br>- Add and setup all furniture and items in this class |

Method

| Name | Description |
|---|---|
| + void setUpdate() | This method is to set the statue to be update and set player emotion |
| + boolean isGameEnd() | Return **true** if all the letterOnStatue is matched with the answer, return **false** otherwise. |

### 3.1.3 **Class Library** extends Room

This class represents a library which is one of the rooms in this game.

Constructor

| Name | Description |
|---|---|
| + Library() | - Initialize the library fields<br>- Set player's emotion as NORMAL<br>- Call setCharacterPane() method from Player class<br>- Add and setup all furniture and items in this class |

### 3.1.4 Class LivingRoom extends Room

This class represents a living room which is one of the rooms in this game.

Constructor

| Name | Description |
|---|---|
| + LivingRoom() | - Initialize the livingRoom fields<br>- Set player's emotion as NORMAL<br>- Call setCharacterPane() method from Player class<br>- Add and setup all furniture and items in this class |

### 3.2 Class CharacterPane extends StackPane

This class represents a character's emotion image in the playing screen.

Constructor

| Name | Description |
|---|---|
| + CharacterPane() | - Call setCharacterPane() method<br>- Set alignment as CENTER |

Method

| Name | Description |
|---|---|
| + void setCharacterPane() | - Initialize image with player's emotion image<br>- Set the image's size |

| | |
|---|---|
| | - Add image to this pane |

### 3.3 Class ControlPane extends VBox

This class represents all control buttons' pane which showed in the main menu.

Field

| Name | Description |
|---|---|
| - Button startButton | The button for starting the game |
| - Button instructionButton | The button for showing the instruction |
| - Button creditButton | The button for showing the credit |
| - Button exitButton | The button for exiting the game |

Constructor

| Name | Description |
|---|---|
| + ControlPane() | - Call initialize all buttons method<br>- Add all button to this pane<br>- Set alignment to CENTER |

Method

| Name | Description |
|---|---|
| - void initializeButtonStyle(Button button) | - Initialize shadow with DropShadow<br>- Setup shadow and button style<br>- Add EventHandler on mouse entered to have a shadow effect |

| | |
|---|---|
| | - Add EventHandler on mouse exited to set effect as null |
| - void initializeStartButton() | - Initialize startButton with text and setup the style<br>- Add EventHandler on mouse clicked to have a click's sound and go to PlayingScreen's scene by using fadeTransition |
| - void initializeInstructionButton() | - Initialize instructionButton with text and setup the style<br>- Add EventHandler on mouse clicked to have a click's sound and show instruction window |
| - void initializeCreditButton() | - Initialize creditButton with text and setup the style<br>- Add EventHandler on mouse clicked to have a click's sound and show credit window |
| - void initializeExitButton() | - Initialize exitButton with text and setup the style<br>- Add EventHandler on mouse clicked to have a click's sound and close the game |
| - void makeFadeTransition(Stage stage) | This method makes the scene change more smoothly by using fadeTransition for change MainMenu to PlayingScreen |

### 3.4 Class CreditPane extends VBox

This class represents the credit of this game which appears in the main menu.

Field

| Name | Description |
|---|---|
| - Text header | The header text that appears in credit's pane |
| - Text body | The body text that appears in credit's pane |

Constructor

| Name | Description |
|---|---|
| + CreditPane() | - Call initializeHeaderText(), initializeBodyText() method<br>- Setup this pane and background's size<br>- Set visible as false<br>- Add header and body to this pane in correct order |

Method

| Name | Description |
|---|---|
| - void initializeHeaderText() | - Initialize header as text "Credit"<br>- Setup the header |
| - void initializeBodyText() | - Initialize body as text "Designed and Created by\nTongRod99\n\nSupported by\nOur beloved family" |

| | |
|---|---|
| | - Setup the body |

### 3.5 Class **DialoguePane** extends StackPane

This class represents the dialogue's pane that showing the message from what the character feels or sees which appears in the playing screen.

Field

| Name | Description |
|---|---|
| - Text gameText | The text which appears in dialogue's pane<br>- Initialize it with Text |

Constructor

| Name | Description |
|---|---|
| + DialoguePane() | - Setup the borderStroke, gameText and this pane |

Method

| Name | Description |
|---|---|
| + getter and setter for each field | |

### 3.6 Class InstructionPane extends VBox

This class represents how to play this game which appears in the main menu

Field

| Name | Description |
|---|---|
| - Text header | The header text which appears in instruction's pane |
| - ImageView howToPlayPic | The image which shows how to play this game |

Constructor

| Name | Description |
|---|---|
| + InstructionPane() | - Call initializeHeaderText(), initializeHowToPlayPic() method<br>- Setup this pane and background size<br>- Set visible as false<br>- Add header and howToPlayPic in this pane in correct order |

Method

| Name | Description |
|---|---|
| - void initializeHeaderText() | - Initialize header with text as "How to play"<br>- Setup the header |

| | |
|---|---|
| - void initializeHowToPlayPic() | - Initialize howToPlayPic with HowToPlay's image<br>- Setup the howToPlayPic |

### 3.7 Class **ItemInHandBox** extends StackPane

This class represents item in hand box's image which appears in the playing screen

Field

| Name | Description |
|---|---|
| - ImageView itemImage | The item's image that the player picks in his/her hand<br>- Initialize itemImage with ImageView |

Constructor

| Name | Description |
|---|---|
| + ItemInHandBox() | - Set itemImage's effect with dropShadow<br>- Add itemImage to this pane<br>- Setup this pane |

Method

| Name | Description |
|---|---|
| + void setImageInBox(Image image) | - Set setImage method as image |
| + void deleteImageInBox() | - Set setImage method as null |

### 3.8 Class **TimerPane** extends StackPane

This class represents timer pane which appears in the playing screen.

Field

| Name | Description |
|---|---|
| - Text timer | The timer's text |

Constructor

| Name | Description |
|---|---|
| + TimerPane() | - Initialize timer with text "00:00:00"<br>- Setup the timer and this pane<br>- Add timer to this pane |

Method

| Name | Description |
|---|---|
| + getter and setter for each field | |

## 4. Package input

### 4.1 Class InputUtility

This class represents the keycode that the player presses in playing screen.

Field

| Name | Description |
|---|---|
| - ArrayList<Keycode> keyPressed | List of the keycodes that the player presses |

Method

| Name | Description |
|---|---|
| + boolean getKeyPressed(KeyCode keycode) | Return **true** if the keyPressed contains a keycode, return **false** otherwise |
| + void setKeyPressed(KeyCode keycode,boolean pressed) | - If the key is pressed and does not contain this keycode, add keycode in keyPressed<br>- Otherwise, remove keycode from the keyPressed |

## 5. Package logic

### 5.1 Class GameController

This class controls the game by setting or changing the player room's map and screen.

Field

| Name | Description |
|---|---|
| - Room currentRoom | The room which the player is inside or shown on the playing screen's map |
| - Timer timer | The timer which countdown the time before the game is over |

Method

| Name | Description |
|---|---|
| + void startGame() | - Call initializePlayer() method from Player class<br>- Clear all entities<br>- Initialize currentRoom with Bedroom<br>- Initialize timer with 3 minutes |
| + void logicUpdate() | - If the time is over, set endgame as false and return<br>- If the current room is end,<br>- If currentRoom is Garden, set endGame as true and return<br>- Call setCurrentRoom() method, Initialize timer with 3 minutes and return |

| | |
|---|---|
| | - Otherwise, call decrementTimer() method, show remaining time, and call logicUpdate() method |
| + void setCurrentRoom() | - Clear all entities<br>- if currentRoom is Bedroom, Initialize currentRoom with LivingRoom<br>- if not and currentRoom is LivingRoom, Initialize currentRoom with Library<br>- if not and currentRoom is Library, Initialize currentRoom with Garden<br>- call the fade transition and initialize player |
| + void endGame(boolean isWin) | - Stop animation timer and background's music<br>- Call makeSceneFadeTransition(isWin) method |
| - void makeSceneFadeTransition(boolean isWin) | This method makes the scene transfers to EndingScreen smoothly |
| - void makeRoomFadeTransition() | This method makes the room transfers to another room smoothly |
| + getter and setter for each field | |

### 5.2 Class Player

This class represents the player who can show the face's emotion, walk around by presses the keycode as A, W, D, S or interact with furniture, items using L, K to find the way out of the room.

Field

| Name | Description |
|---|---|
| - Item itemInHand | The item in player's hand |
| - Emotion playerEmotion | The player's emotion |
| - double xPosition | The player's position in x-axis |
| - double yPosition | The player's position in y-axis |
| - Direction faceDirection | The face's direction that the player moves to |
| - final double eachStep | The player's speed<br>- Initialize it to 4.5 |
| - final double areaForFoot | The player's foot area<br>- Initialize it to 40 |

Method

| Name | Description |
|---|---|
| + void initializePlayer() | Initialize the player (itemInHand, ItemInHandBox, xPosition, yPosition, and faceDirection) |
| + void useItem() | Call useItem() method with the furniture that is in front of the player |
| + void observe() | Call observe() method with the furniture that is in front of the player |

| - boolean isInfrontOf(Furniture furniture) | Return **true** if player is in front of this furniture, return **false** otherwise |
|---|---|
| + void move() | - Perform the player's move |
| + void setxPosition(double xPosition) | Set the player's move in x-axis |
| + void setyPosition(double yPosition) | Set the player's move in y-axis |
| + void draw(GraphicsContext gc) | Draw the player on her/his current position. |
| + void logicUpdate() | Update the logic when player presses the keycode A, W, D, S for move, K for observe, and L for use item |
| + getter and setter for each field | |

### 5.3 Class Timer

This class represents remaining time before game is over in playing screen.

Field

| Name | Description |
|---|---|
| - int minute | The time that shows in minutes |
| - int second | The time that shows in seconds |
| - int ms | The time that shows in milliseconds |

Constructor

| Name | Description |
|---|---|
| + Timer(int m,int s, int ms) | - Initialize the timer fields with respective values. |

Method

| Name | Description |
|---|---|
| + void decrementTimer(int amount) | This method will represent the time decreasing<br>- If time is over, return<br>- Decrease ms with amount<br>- While ms less than 0<br>- If time is over, set ms with 0 and return<br>- Increase ms with 100 and decrease seconds with 1<br>- While seconds less than 0, increase seconds with 60 and decrease minute with 1 |
| + boolean isTimerOver() | It returns **true** if minute, seconds, and ms less or equal to 0, returns **false** otherwise. |
| + String toString() | Returns a formatted String in the format of "<minute>:< seconds>:<ms>" |
| + int getDuration() | Return ((minute * 60) + seconds )*1000 + ms |
| + int getDuration(int minute, int seconds, int ms ) | Return ((minute * 60) + seconds )*1000 + ms |
| + getter and setter for each field | |

## 6. Package screen

### 6.1 Class **EndingScreen** extends StackPane

This class represents the ending screen of this game.

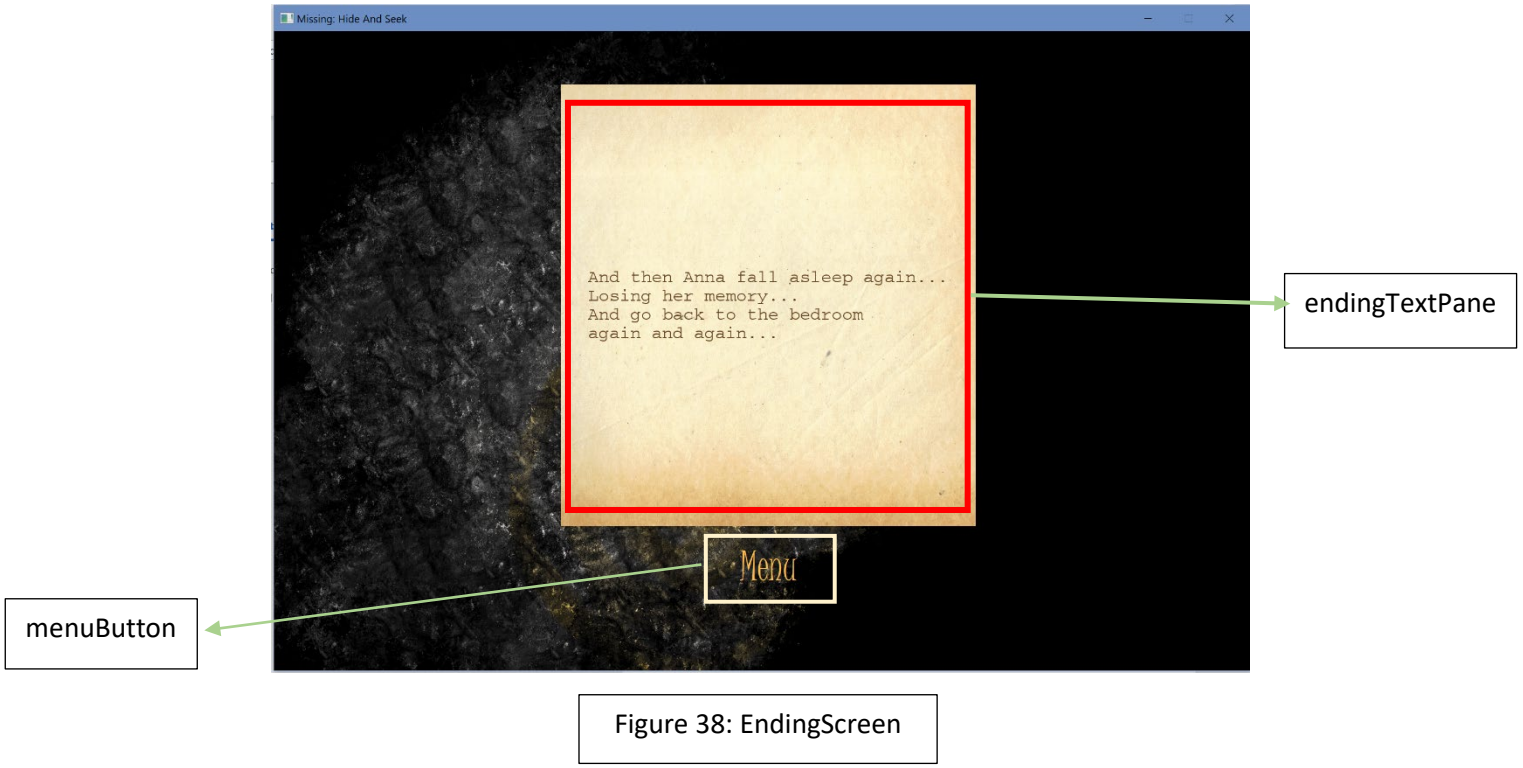



Figure 38: EndingScreen

Field

| Name | Description |
|---|---|
| - StackPane endingTextPane | The pane which including ending text |
| - Button menuButton | The button which returns to the main menu |

Constructor

| Name | Description |
|---|---|
| + EndingScreen(boolean isWin) | - Call initializeEndingTextPane(isWin), initializeMenuButtonPane() method and add them to this pane in order<br>- Setup this pane<br>- Add background's music and set cycle count to INDEFINITE |

Method

| Name | Description |
|---|---|
| - void initializeEndingTextPane(boolean isWin) | - Initialize endingTextPane with StackPane, setup its style<br>- Initialize endingText with different Text for isWin equal true or false<br>- Add endingText to endingTextPane |
| - void initializeMenuButtonPane() | - Initialize menuButton, shadow and setup their style<br>- Add EventHandler on mouse entered to have a shadow effect<br>- Add EventHandler on mouse exited to set effect as null<br>- Add EventHandler on mouse click to add background music and play sound effect, then return to main menu |

### 6.2 Class MainMenu extends HBox

This class represents the main menu of this game.


Figure 39: MainManu

Field

| Name | Description |
|---|---|
| - ControlPane controlPane | Includes all the control buttons in the main menu |
| - CreditPane creditPane | The creditPane where the player can see the credit of this game |

| | |
|---|---|
| - InstructionPane instructionPane | The instructionPane which shows the player how to play this game |
| - StackPane windowPane | The windowPane where to show the creditPane and instructionPane |
| - VBox sidePane | The sidePane that includes the logo image and controlPane |
| + final int INDEFINITE | Variable for looping background music<br>- Initialize it to -1 |

Constructor

| Name | Description |
|---|---|
| + MainMenu() | - Initialize ControlPane, CreditPane, InstructionPane, sidePane and windowPane<br>- Add controlPane and logo's image to the sidePane in correct order<br>- Setup sidePane and windowPane<br>- Add windowPane and sidePane to the mainMenu in correct order<br>- add background's music and set cycle count to INDEFINITE |

Method

| Name | Description |
|---|---|
| + void showCreditWindow() | - If creditPane's visible is false,<br>- Set instructionPane's visible as false |

| | |
|---|---|
| | - Set creditvisible as true<br>- Otherwise, set creditPane's visible as false |
| + void showInstructionWindow() | - If instructionPane's visible is false,<br>- Set creditPane's visible as false<br>- Set instructionPane's visible as true<br>- Otherwise, set instructionPane's visible as false |
| + getter and setter for each field | |

### 6.3 Class PlayingScreen extends VBox

This class represents the playing screen of this game.


Figure 40: PlayingScreen

Field

| Name | Description |
|---|---|
| - Button menuButton | Button which back to main menu |
| - CharacterPane character | characterPane where to show character image and emotion |
| - DialoguePane dialoguePane | dialoguePane where to show the message from what character feels or sees |
| - ItemInHandBox objectInHandBox | ItemInHandBox where show the item which player picks it up |
| - TimerPane timerPane | timePane where to show the timer label remaining before the game is over |
| - Pane room | room where to show the room's map |
| - AnimationTimer animation | Animation timer which be called when the game start |

Constructor

| Name | Description |
|---|---|
| + PlayingScreen() | - Call startGame() method from GameController class<br>- Call initializeRoomPane() method<br>- Initialize sidePane, dialoguePane, itemInHandBox, upperPane and lowerPane<br>- Setup this pane, upperPane, and lowerPane<br>- Add sidePane, room to the upperPane and dialoguePane, itemInHandBox to the lowerPane in correct order |

| | |
|---|---|
| | - Add upperPane, lowerPane to this pane in correct order<br>- Add listerner()<br>- Initialize and start animationTimer |

Method

| Name | Description |
|---|---|
| - void initializeMenuButton() | - Initialize menuButton, shadow and setup their style<br>- Add EventHandler on mouse entered to have a shadow effect<br>- Add EventHandler on mouse exited to set effect as null<br>- Add EventHandler on mouse click to add background music and play sound effect and turn back to main menu |
| - VBox initializeSidePane() | - Call initializeMenuButton() method<br>- Initialize character, timerPane and sidePane<br>- Add menuButton, timerPane, characterPane to the sidePane in correct order<br>- Setup and return the sidePane |
| - void initializeRoomPane() | - Initialize and setup room<br>- Add currentRoom to room |
| - void addListerner() | - Add EventHandler on key pressed to get the keycode by using method from InputUtility |

| | |
|---|---|
| | - Add EventHandler on key released to remove the keycode by using method from InputUtility |
| + void setRoomPane(Room currentRoom) | Clear and add currentRoom to room |
| + void setCharacterPane() | Clear and setup the character from CharacterPane class |
| + getter and setter for each field | |

## 7. Package sharedObject

### 7.1 Interface IRenderable

Method

| Name | Description |
|---|---|
| + int getZ() | Use to ordering the object that have to be drawn on the screen |
| + void draw(GraphicsContext gc) | Draw the object on its current position. |
| + boolean isVisible() | Return **true** if the object is visible, return **false** otherwise |

### 7.2 Class RenderableHolder

Field

| Name | Description |
|---|---|
| - final RenderableHodler instance | Initialize it with RenderableHolder |
| - List<IRenderable> entities | The list of entities that have to be drawn on the screen which be sorted by z |

| | |
|---|---|
| - Comparator<IRenderable> comparator | Use to ordering the object |
| + Map<String, Image> furnitureSprite | Map of image's name to furniture's image |
| + Map<String, Image> itemSprite | Map of image's name to item's image |
| + Map<Emotion, Image> characterFullBody | Map of emotion to character's image |
| + Map<Direction, Image> characterSprite | Map of character direction to character sprite's image |
| + Map<String, Image> background | Map of background's name to background's image |
| + Map<String, AudioClip> soundFX | Map of sound effect's name to the sound effect's audio |
| + Map<String, AudioClip> bgMusic | Map of background music's name to the background's audio |
| + Font juiceICTFont | Font that is used in this game |
| + Font couriterryFont | Font that is used in this game |

Constructor

| Name | Description |
|---|---|
| + RenderableHolder() | - Initialize entities with ArrayList of IRenderable<br>- Initialize comparator to sort the object by using z |

Method

| Name | Description |
|---|---|
| + void loadResource() | Call all Load methods in this class |

| | |
|---|---|
| - void loadMainMenuResource() | Load all backgrounds' image, background's music, music effect, and font that used in main menu |
| - void loadCharacterSpriteAndBody() | Load all character's emotions and face's directions image |
| - void loadItemResource() | Load all items' image |
| - void loadBedroomResource() | Load bedroom's background and furniture's images that used in bedroom |
| - void loadLivingRoomResource() | Load living room's background and furniture's image that used in living room |
| - void loadLibraryResource() | Load library's background and furniture's image that used in library |
| - void loadGardenResource() | Load garden's background and furniture's image that used in garden |
| + void add(IRenderable entity) | Add entity to entities then sort by using comparator |
| + getter and setter for each field | |