**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, pick and use an object. The goal of each room is to find a way to open the door.

**Example**

**Main Menu Scene**

A screenshot of a doll

Description automatically generated with low confidenceMain menu is the first page of the game which including Start button, Instruction button, Credit button, and Exit button.

Figure 1: Main Menu

Logo

Description automatically generatedText, logo

Description automatically generated- When you put mouse enter the button, it will have shadow effect and has sound effect when click the button.

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.

**Text

Description automatically generated**

**Text

Description automatically generated**A screenshot of a doll

Description automatically generated with low confidenceA screenshot of a doll

Description automatically generated with low confidence

Figure 5: when clicking the Instruction button again

Figure 4: when clicking the Instruction button

Figure 7: when clicking the Credit button again

Figure 6: when clicking the Credit button

**Graphical user interface

Description automatically generated with low confidencePlaying 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), “L” (to use the item).

Up

Right

Figure 9: Character face’s direction

Down

Left

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generatedGraphical user interface

Description automatically generatedGraphical user interface

Description automatically generated

Figure 10: Rooms

Bedroom

Living room

Library

Garden

Graphical user interface

Description automatically generated with medium confidenceGraphical user interface

Description automatically generatedThere have 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 12: When the room getting darker

Figure 11: Normal room

**A picture containing toy, doll

Description automatically generated**A picture containing toy, doll

Description automatically generatedA cartoon of a child

Description automatically generated with low confidence- For character’s emotion has: NORMAL, WORRIED, SHOCK which displayed in descending order of time

Figure 13: Character’s emotion

SHOCK

Normal

WORRIED

**A close up of the moon

Description automatically generated with low confidence**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 and find another items which are hidden in different furniture and rooms.

Figure 14: Key

Figure 15: Pocket knife

Logo

Description automatically generatedA picture containing logo

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

Figure 16: Key in itemInHandBox

Figure 17: Pocket knife in itemInHandBox

**- Note: Not all individual items can be use with all furniture. You need to find the matched furniture for using it.**

A picture containing text

Description automatically generated - And the **note** which contains the hidden message that will show in Dialogue Pane. The notes are either on the table or hide with the dolls.

Figure 19: Dolls

Figure 18: Table with note

Text

Description automatically generated

Figure 20: Dialogue Pane

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

A picture containing text, indoor

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

Figure 21: Sofa

A close-up of a door handle

Description automatically generated with medium confidenceA picture containing text, door

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

Figure 23: Safe

Figure 22: Cupboard

A shelf with books on it

Description automatically generated with low confidenceA white rectangle with a black border

Description automatically generated with low confidence- The furniture which can contain the items: Bookshelf, Cupboard, TableWithNote, Safe, Mirror

Figure 24: Bookshelf

Figure 25: Mirror

A picture containing text, picture frame

Description automatically generated****- The furniture in the rooms that do not interact with the player

Figure 26: TableWithLamp

Figure 27: Bed

Figure 28: Chair

Figure 29: FamilyPic

A picture containing indoor, window, screen, plant

Description automatically generatedA picture containing indoor, furniture, wooden, table

Description automatically generatedA close up of a book

Description automatically generated with low confidence

Figure 30: IvyPic

Figure 31: LongTableWithLamp

Figure 32: MysteryBox

A painting in a frame

Description automatically generated with medium confidence

Figure 34: Window

Figure 33: ShieldPic

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

A picture containing text

Description automatically generated

Figure 35: Door

**Ending Screen Scene**

There have 2 different screen which depend on the player’s condition.

Text

Description automatically generatedLose 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 game over’s screen. Player can click Menu button for back to Main Menu.

Text, letter

Description automatically generatedWin condition

Figure 37: End Game’s screen

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

**Class diagram**

**1. package application**

**1.1 class Main**

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

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void start(Stage primaryStage) throws Exception | - Initialize mainMenu with the MainMenu  - Create the scene with the mainMenu  - Set primaryStage scene as the created scene  - Set primaryStage title as “Missing: Hide and Seek”  - Setup primaryStage height as 700  - Set primaryStage width as 1120  - Set primaryStage resizable as false  - Show 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** |
| + void cut() | This method will be called when the furniture got cut. |
| + Boolean isCut() | Return **true** if the furniture got cut, return **false** otherwise. |

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

**2.1.4 Interface Pickable**

This interface defines methods for furniture that can be picked.

Method

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

**2.1.5 Interface Updateable**

This interface defines methods for furniture that can be updated.

Method

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

**2.1.6 Enum Emotion**

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

**2.1.7 *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.  - 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 |  |

**2.1.8 *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.  - Set isVisible as true by default  - Set imageString as name  - Set the area for the player to interact |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void draw(GraphicsContext gc) | Draw the furniture on its current position. |
| + void observe() | This method will represent the furniture’s name which observed by the player  - Set gameText as toString() mehtod |
| + void useItem() | This method will represent the default text which show that this item can not use with this furniture  - 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 boolean object. |
| + 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 |  |

**2.1.9 *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 | Initialize it to 20 |
| # final double height | Initialize it to 20 |

Constructor

|  |  |
| --- | --- |
| **Name** | **Description** |
| + Item(String name, double xPosition, double yPosition, int z) | - Initialize the item fields with respective values.  - 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 |  |

**2.1.10 *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 |
| - String startText | The message that appears when the room is start |
| - ArrayList<Furniture> furniture | The list of furniture in the room |
| - boolean isGameEnd | State that the game is end or not |
| - final double floorStartX | Initialize it to 0 |
| - final double floorStartY | Initialize it to 140 |

Constructor

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

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void paintComponent() | Draw all the components in the room  Call fade() method |
| + void fade() | The room will begin to darken over the time by using graphicsContext |
| + void startGame() | Set gameText as startText |
| + getter and setter for each field |  |

**2.1.11 Enum Direction**

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

LEFT, RIGHT, UP, DOWN, NONE.

**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** |
| - ArrayList<Item> item | The list of items in the bookkshelf |
| - 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.  - Initialize item as an empty ArrayList  - 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  - else if it has item in bookshelf, check all items  - If it is a note, call read() method  - If it is a pocket knife or key, call pick() method and remove it  - Otherwise, set gameText with “There is a lot of books here, I would love to read if I have time.” |
| + getter and setter for each field |  |

**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 |
| - ArrayList<Item> item | The list of items in the cupboard |

Constructor

|  |  |
| --- | --- |
| **Name** | **Description** |
| Cupboard(String name, int xPosition, int yPosition, int z) | - Initialize the cupboard fields with respective values.  - Initialize item as an empty ArrayList  - 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 add sound effect as lock.  - Otherwise, set isOpened as true, imageString as “OpenedCupboard”, add sound effect as open, and visible all items in cupboard |
| + void close() | - Set isOpened as false and imageString as this name.  - Add sound effect as close  - Set all item’s 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,  - Set isLocked as false  - Call open() method  - Otherwise, set gameText as “I think it doesn’t fit here.” |
| + void useItem() | - If the cupboard is locked, call unlock() method  - Otherwise, call useItem() method from super class |
| + void observe() | - Call open() method  - If the cupboard is locked, return  - If it has item in cupboard, check all items in cupboard  - If it is a note, call read() method  - If it is a pocket knife or key, call pick() method and remove it  - 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.  - Set to not been updated by default. |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void update() | - Set imageString as “BehindMirror”, add sound effect as Breaking Glass and visible all items in mirror  - Set isUpdated as true  - 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  - If the mirror is updated and has items in mirror, check all items in mirror  - If it is a note, call read() method  - If it is a pocket knife or key, call pick() method and remove it  - Otherwise, set gameText as “It’s broken…” |
| + boolean isUpdated() | Return **true** if the mirror got updated, return **false** otherwise. |
| + void setIsUpdated(boolean isUpdated) | - Set isUpdated as isUpdated  - If currentRoom is a Garden, call setUpdate() method from GameController 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 in the safe which behind the picture |
| - boolean isReveal | State that the safe behind picture got revealed or not. |
| - boolean isSafeBehind | State that has 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.  - Initialize itemBehind as an empty ArrayList  - Set isSafeBehind, isLocked and isReveal as false |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void observe() | - If it has safe behind the picture, call observeWithSafe() method  - Otherwise, call observeBehind() method |
| + void observeWithSafe() | - If safe which behind the picture is not revealed,  - Set isReveal as true  - Set imageString as “ClosedSafe”  - Set gameText “There is a safe behind.”  - else if the safe is locked, set gameText as “It’s locked.”  - Otherwise, call observeBehind() method |
| + void observeBehind() | - If it has items in the safe, Check all the items  - If item is a note, call read() method  - If item is a pocket knife or key, call pick() method and remove it  - 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,  - Set isLocked as false  - Set imageString as “OpenedSafe”  - Visible all items in the safe.  - Otherwise, set gameText as “I think it doesn’t fit here.” |
| + getter and setter for each field |  |

**2.2.7 Class Sofa extends Container implements Cuttable**

This class represents a sofa which is one of the containers 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.  - Set to not been cut by default  - Initialize thingBehind as an empty ArrayList |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void cut() | - Set isCut as true  - Set imageString as “CutSofa”  - 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 superclass  - Otherwise,  - If the player’s hand has a pocket knife and can use with this safe, call cut() method.  - 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  - if not and sofa has not been cut, set gameText as “I think there is something under the sofa”  - If not and it has something behind the sofa, check all the items  - If it is a note, call read() method  - If it is a pocket knife or key, call pick() method and remove it  - Otherwise, set gameText as “Little sorry, to make it’s torn.” |
| + getter and setter for each field |  |

**2.2.8 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  - Initialize it with ArrayList<character>(Arrays.asList (‘M’, ‘H’, ‘I’, ‘P’, ‘E’, ‘N’, ‘A’, ‘S’, ‘O’)) |
| - final double translationDisY | Initialize it to 50 |
| - final double translationDisX | Initialize it to 35 |
| - char letterOnStatue | The letter on the statue that the player has to change to match with the answerLetter |
| - 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.  - 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  - Otherwise,  - Set new letterOnStatue by move to the next character from the characterSet  - Set gameText as "This statue is strange!\nI think this is the puzzle\n  but how can I solve this." |
| + void draw(GraphicsContext gc) | - Call draw(gc) method from super class  - Setup gc’s font and fill a color  - Call gc.fillText with letterOnStatue on its current position + translationDis in x-axis and y-axis |
| + getter and setter for each field |  |

**2.2.9 Class TableWithLamp extends Furniture implements Updateble**

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. |

**2.2.10 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  - If it is a note, call read() method  - If it is a key or pocket knife, call pick() method and remove it  - Otherwise, call observe() method from super class |

**2.2.11 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.  - Set to not been opened by default. |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void open() | - Add sound effect as open  - Set imageString as “OpenedWindow”  - Set isOpened as true |
| + void close() | - Add sound effect as close  - Set imageString as this name  - 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  - Set gameText as “Why everything is silent?” |
| + getter and setter for each field |  |

**2.2.12 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 window 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  - 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" |
| + void draw(GraphicsContext gc) | - draw the door by using fillRect with (this.getxPosition(), this.getyPosition(), 40, 40) and fill a brown color |

**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.  - Set to not been picked by default. |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + boolean isPicked() | It returns **true** if the key got picked, returns **false** otherwise. |
| + void pick() | - Set the key’s image in itemInHandBox  - Set itemInHand with this key  - Set isPicked as True  - Set isVisible from super class as false |
| + boolean useWith(Furniture furniture) | - If furniture is equal to the matchedFurniture,  - delete the key’s image from itemInHandBox  - Set itemInHand as null  - Set isPicked as false  - Add sound effect as PickItUp  - return true  - 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 which appears on the note |
| - 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.  - Set isVisible from super class as false  - Set imageString as “” |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void read() | Set gameText as textOnNote |
| + draw(GraphicsContext gc) | - If imageString is empty, return  - 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.  - Set to not been picked by default. |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + boolean isPicked() | It returns **true** if the pocket knife got picked, returns **false** otherwise. |
| + void pick() | - Set the pocket knife’s image initemInHandBox  - Set itemInHand with this pocket knife  - Set isPicked as True  - Set isVisible from super class as false |
| + boolean useWith(Furniture furniture) | - If furniture is equal to the matchedFurniture,  - delete the pocket knife’s image from itemInHandBox  - Set itemInHand as null  - Set isPicked as false  - Add sound effect as PickItUp  - return true  - 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  - Set player’s emotion as NORMAL  - Call setCharacterPane() method from Player class  - 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  - Set player’s emotion as NORMAL  - Call setCharacterPane() method from Player class  - Add and setup all furniture and items in this class |

**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  - Set player’s emotion as NORMAL  - Call setCharacterPane() method from Player class  - 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  - Set player’s emotion as NORMAL  - Call setCharacterPane() method from Player class  - 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  - Set alignment as CENTER |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void setCharacterPane() | - Initialize img with player’s emotion image  - Setup the img  - Add img to this pane’s children |

**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 initializeStartButton(), initializeInstructionButton(), initializeCreditButton(), initializeExitButton() method  - Add startButton, instructionButton, creditButton, exitButton to this pane’s children in correct order  - Set alignment to CENTER |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void initializeButtonStyle(Button button) | - Initialize shadow with DropShadow  - Setup shadow and button  - 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 “Start”  - Call initializeButtonStyle( startButton ) method  - Add EventHandler on mouse clicked to have a click’s sound and go to PlayingScreen’s scene |
| + void initializeInstructionButton() | - Initialize instructionButton with text “Instruction”  - Call initializeButtonStyle( instructionButton ) method  - Add EventHandler on mouse clicked to have a click’s sound and show instruction window |
| + void initializeCreditButton() | - Initialize creditButton with text “Credit”  - Call initializeButtonStyle( ceditButton ) method  - Add EventHandler on mouse clicked to have a click’s sound and show credit window |
| + void initializeExitButton() | - Initialize exitButton with text “Exit”  - Call initializeButtonStyle( exitButton ) method  - Add EventHandler on mouse clicked to have a click’s sound and close the game |

**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  - Setup this pane and background size  - Set visible as false  - Add header and body to this pane’s children in correct order |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void initializeHeaderText() | - Initialize header as text “Credit”  - 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** |
| - static Text gameText | The text which appears in dialogue’s pane  - Initialize it with Text |

Constructor

|  |  |
| --- | --- |
| **Name** | **Description** |
| + DialoguePane() | - Initialize borderStroke with BorderStroke  - 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  - Setup this pane and background size  - Set visible as false  - Add header and howToPlayPic in this pane’s children in correct order |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void initializeHeaderText() | - Initialize header with text as “How to play”  - Setup the header |
| + void initializeHowToPlayPic() | - Initialize howToPlayPic with HowToPlay’s image  - 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** |
| - static ImageView itemImage | The item’s image that the player picks in his/her hand  - Initialize itemImage with ImageView |

Constructor

|  |  |
| --- | --- |
| **Name** | **Description** |
| + ItemInHandBox() | - Initialize borderStroke with BorderStoke and setup  - Initialize dropShadow with DropShadow and setup  - Set itemImage’s effect with dropShadow  - Add itemImage to this pane’s children  - Setup this pane |

Method

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

**3.8 Class TimerPane extends Pane**

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

Field

|  |  |
| --- | --- |
| **Name** | **Description** |
| - Text timer | The timer’s label |

Constructor

|  |  |
| --- | --- |
| **Name** | **Description** |
| + TimerPane() | - Initialize timer with text “00:00:00”  - Setup the timer and this pane  - Add timer to this pane’s children |

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** |
| - static ArrayList<Keycode> keyPressed | List of the keycode that the player presses |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + static boolean getKeyPressed(KeyCode keycode) | Return **true** if the keyPressed contains a keycode, return **false** otherwise |
| + static void setKeyPressed(KeyCode keycode,boolean pressed) | - If the key is pressed and does not contain this keycode, add keycode in keyPressed  - 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** |
| - static Room currentRoom | The room which the player is inside or shown on the playing screen’s map |
| - static Timer timer | The timer which showed time remaining before the game is over |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + static void startGame() | - Call initializePlayer() method from Player class  - Clear all entities by using method from RenderableHolder class  - Initialize currentRoom with Bedroom  - Initialize timer with Timer(3,0,0) |
| + static void logicUpdate() | - If the time is over, set endgame as false and return  - If the current room is end,  - If currentRoom is Garden, set endGame as true and return  - Call setCurrentRoom() method, Initialize timer with Timer(3, 0, 0) and return  - Otherwise, call decrementTimer() method, set timer as timer.toString() by using PlayingScreen class, and call logicUpdate() method |
| + static void setCurrentRoom() | - Clear all entities by using method from RenderableHolder class  - if currentRoom is Bedroom, Initialize currentRoom with LivingRoom  - else if currentRoom is LivingRoom, Initialize currentRoom with Library  - else if currentRoom is Library, Initialize currentRoom with Garden  - SetRoomPane as currentRoom  - Call initializePlayer() method from Player class |
| + static void endGame() | - Call animation stop from PlayingScreen class and go to EndingScreen’s scene  - Return |
| + 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** |
| - static Item itemInHand | The item in player’s hand |
| - static Emotion playerEmotion | The player’s emotion |
| - static double xPosition | The player’s position in x-axis |
| - static double yPosition | The player’s position in y-axis |
| - static Direction faceDirection | The face’s direction that the player moves to |
| - final static double eachStep | Initialize it to 4.5 |
| - final static double areaForFoot | Initialize it to 40 |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + static void initializePlayer() | Initialize the player (set the itemInHand as null, delete all image in ItemInHandBox, set xPosition and yPosition to 260, set faceDirection to DOWN) |
| + static void useItem() | Call useItem() method with the furniture that is in front of the player |
| + static void observe() | Call observe() method with the furniture that is in front of the player |
| + static boolean isInfrontOf(Furniture furniture) | Return **true** if player is in front of this furniture, return **false** otherwise |
| + static void move() | - Perform the player’s move by presses the keycode A, W, D, S  - Break it if hit the furniture |
| + static setxPosition(double xPosition) | Set the player’s move in x-axis |
| + static void setyPosition(double yPosition) | Set the player’s move in y-axis |
| + static void draw(GraphicsContext gc) | Draw the player on her/his current position. |
| + static 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) | - If time is over, return  - Decrease ms with amount  - While ms less than 0  - If time is over, set ms with 0 and return  - Increase ms with 100 and decrease seconds with 1  - 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 |
| + static 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**

This class represents the ending screen of this game.

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) | Call initializeEndingTextPane(isWin) and initializeMenuButtonPane() method  - Add endingTextPane, menuButton to this pane’s children in correct order  - Setup this pane  - Set ending screen background-size and style  - Add background’s music and set cycle count to INDEFINITE |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void initializeEndingTextPane(boolean isWin) | - Initialize endingTextPane with StackPane, set min-max size, background, and padding  - Initialize endingText with different Text for isWin equal true or false  - Add endingText to endingTextPane’s children |
| + void initializeMenuButtonPane() | - Initialize menuButton with text “Menu”  - Initialize shadow with DropShadow  - Setup the menuButton and shadow  - Add EventHandler on mouse entered to have a shadow effect  - Add EventHandler on mouse exited to set effect as null  - Add EventHandler on mouse click to add background music and sound effect as ButtonClick and return to main menu |
| + getter and setter for each field |  |

**6.2 Class MainMenu extends Scene**

This class represents the main menu screen of this game.

instructionPane/ creditPane

**sidePane**

**Text

Description automatically generated**

**Logo image**

**windowPane**

**controlPane**

Field

|  |  |
| --- | --- |
| **Name** | **Description** |
| - static ControlPane controlPane | Includes all the control buttons in the main menu |
| - static CreditPane creditPane | The creditPane where the player can see the credit of this game |
| - static InstructionPane instructionPane | The instructionPane which shows the player how to play this game |
| - static StackPane windowPane | The windowPane where to show the creditPane and instructionPane |
| - static VBox sidePane | The sidePane that includes the logo image and controlPane |
| + static final int INDEFINITE | Initialize it to -1 |

Constructor

|  |  |
| --- | --- |
| **Name** | **Description** |
| + MainMenu() | - Initialize ControlPane with the Control Pane, CreditPane with the Credit Pane, InstructionPane with the Instruction Pane, sidePane with the VBox, windowPane with the Stack Pane  - Add controlPane and logo’s image to the sidePane’s children in correct order  - Setup sidePane and windowPane  - Add windowPane and sidePane to the mainMenu’s children in correct order  - add background’s music and set cycle count to INDEFINITE |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void showCreditWindow() | - If creditPane’s visible is false,  - Set instructionPane’s visible as false  - Set creditvisible as true  - Otherwise, set creditPane’s visible as false |
| + void showInstructionWindow() | - If instructionPane’s visible is false,  - Set creditPane’s visible as false  - Set instructionPane’s visible as true  - Otherwise, set instructionPane’s visible as false |
| + getter and setter for each field |  |

**6.3 Class PlayingScreen**

Graphical user interface

Description automatically generated This class represents the playing screen of this game.

room

TimerPane

CharacterPane

ItemInHandBox

DialoguePane

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 |

Constructor

|  |  |
| --- | --- |
| **Name** | **Description** |
| + PlayingScreen() | - Call startGame() method from GameController class  - Initialize sidePane with initializeSidePane() method  - Call initializeRoomPane() method  - Initialize dialoguePane with the Dialogue Pane, itemInHandBox with the ItemInHandBox, and upperPane, lowerPane with the HBox  - Setup this pane, upperPane, and lowerPane  - Add sidePane, room to the upperPane and dialoguePane, itemInHandBox to the lowerPane in correct order  - Add upperPane, lowerPane to this pane’s children in correct order  - Add listerner() method  - Initialize animationTimer |

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| + void initializeMenuButton() | - Initialize menuButton with text “MENU”  - Initialize shadow with the DropShadow  - setup menuButton and shadow  - Add EventHandler on mouse entered to have a shadow effect  - Add EventHandler on mouse exited to set effect as null  - Add EventHandler on mouse click to add background music and sound effect as ButtonClick andreturn to main menu |
| + VBox initializeSidePane() | - Call initializeMenuButton() method  - Initialize character with the CharacterPane, timerPane with the TimerPane, sidePane with the VBox  - Add menuButton, timerPane, characterPane to the sidePane’s children in correct order  - Setup and return the sidePane |
| + void initializeRoomPane() | - Initialize room with Pane  - Setup room  - Add current room from GameController class to room’s children |
| + void addListerner() | - Add EventHandler on key pressed to get the keycode by using method from InputUtility  - Add EventHandler on key released not to get the keycode by using method from InputUtility |
| + static void setRoomPane(Room currentRoom) | Clear and add currentRoom to room’s children |
| + static 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** |
| - static final RenderableHodler instance | Initialize it with RenderableHolder |
| - List<IRenderable> entities | The list of entities that have to drawn on the screen which be sorted by z |
| - Comparator<IRenderable> comparator | Use to ordering the object |
| + static Map<String, Image> furnitureSprite | Map of image’s name to furniture’s image |
| + static Map<String, Image> itemSprite | Map of image’s name to item’s image |
| + static Map<Emotion, Image> characterFullBody | Map of emotion to image’s character |
| + static Map<String, Image> characterSprite | Map of character image’s name to character’s image |
| + static Map<String, Image> background | Map of background’s name to background’s image |
| + static Map<String, AudioClip> soundFX | Map of sound effect’s name to the sound effect’s audio |
| + static Map<String, AudioClip> bgMusic | Map of background music’s name to the background’s audio |
| + static Font juiceICTFont | Font that is used in this game |
| + static Font couriterryFont | Font that is used in this game |

Constructor

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

Method

|  |  |
| --- | --- |
| **Name** | **Description** |
| static | Call loadResource() to load all resources only 1 time |
| + static void loadResource() | Call all Load methods in this class |
| + static void loadMainMenuResource() | Load all backgrounds’ image, background’s music, music effect, and font that used in main menu |
| + static void loadCharacterSpriteAndBody() | Load all character’s emotions and face’s directions image |
| + static void loadItemResource() | Load all items’ image |
| + static void loadBedroomResource() | Load bedroom’s background and furniture’s images that used in bedroom |
| + static void loadLivingRoomResource() | Load living room’s background and furniture’s image that used in living room |
| + static void loadLibraryResource() | Load library’s background and furniture’s image that used in library |
| + void add(IRenderable entity) | Add entity to entities then sort by using comparator |
| + getter and setter for each field |  |