CHULA ENGINEERING
Foundation toward Innovation

Engineering Library
Reservation Application

Developed by

Krist Pornpairin 6031301721

Kantorn Chitchuen 6030038521

This project is under subject 2110215 Programming Methodology (2018/1)

Computer Engineering, Faculty of Engineering, Chulalongkorn University

Engineering Library

Library is an important part of university. Not only being a place for student to learn and studies but also for sharing time with friend. But, in reality, most of the time this popular library is not enough for everyone. Many time, people come back with disappoint when they cannot find any seat for them self. Because of that, our team try to create the system for helping you reserving seat and checking seat status without actually come to the place.

Engineering Library are implemented is based on reality and friendly. How easy, even sign up is not necessary because the authenticate system is depend on Chulalongkorn University. Friendly design never come without pain, we have tried so hard to improve our user interface and friendly layout design because our design objective is “work like a pro without know any manual”. Crossing platform, that sound easy to us because all of the action can be done with only one stroke, integrate with touch screen is our near future plan.

Before the last. We like to thank you for your supporting and make this project finish. Hope this will be apply in real life.

Our team.

1. Overview of Application

The Engineering Library Reservation Application is a GUI application that can view available seats and take reservation on every seats in the library.

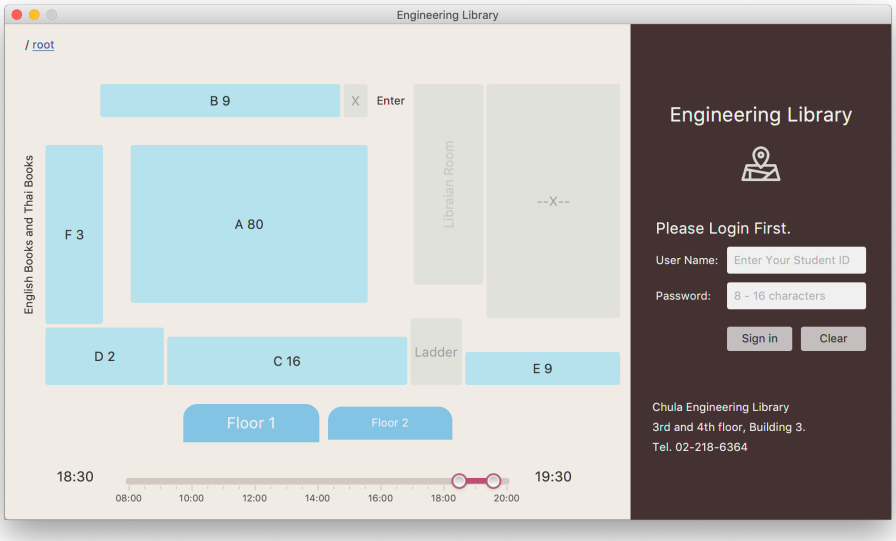


Fig 1: Start screen of the application

Start screen of the application has divided into 2 main parts. The left part shows a floor map of the library, user can click on a button “Floor 1” or “Floor 2” to switch the floor map between floor 1 and floor 2.

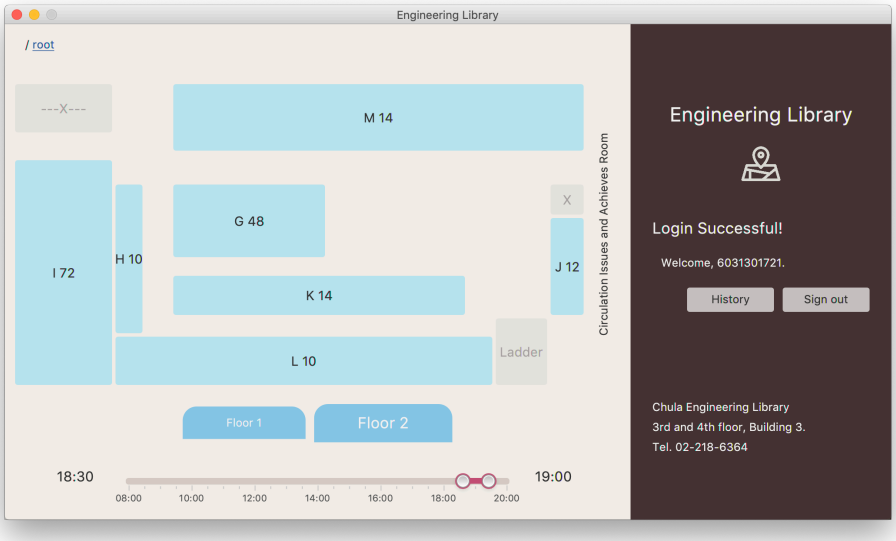


Fig 2: when user click on a “Floor 2” button and login.

When user want to take reservation on which seat, user have to login on the right pane, using student ID and password on the CUNET system. Then, user have to select start and end time user want to use, using slider bar at bottom of screen, the application will show available seat in zone as number after name of the zone. After that, user have to

select a floor, a zone, and a seat which user desired. The application will take the user to appropriate screen.

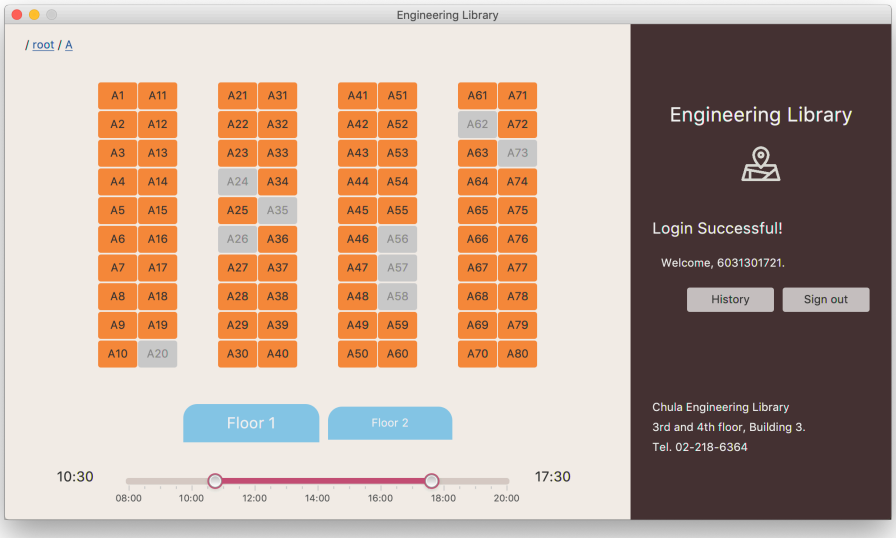


Fig 3: Selecting seat in zone A screen

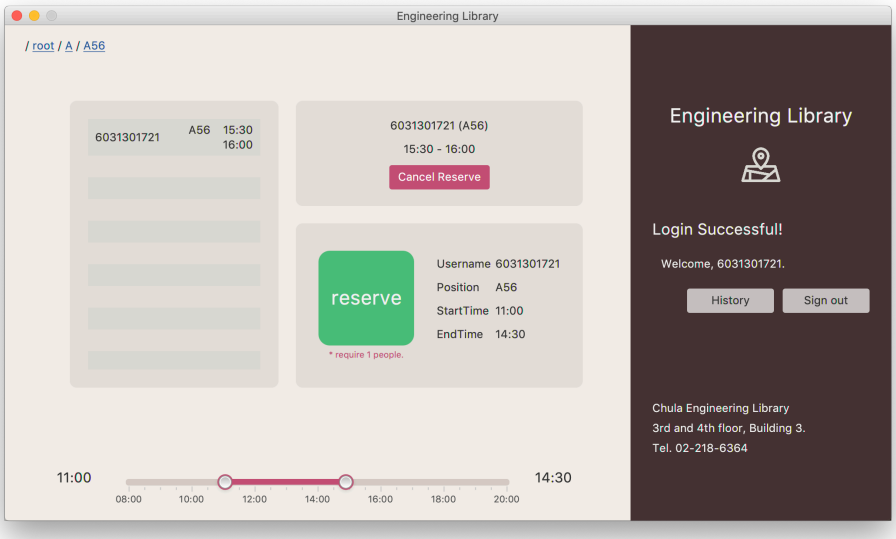


Fig 4: Reserving screen after user selected seat A56

When user have selected required data, because some zone has a table with more than one seat, user have to input ID (or IDs) of who user want to reserve with before take reservation.

Fig 5: Screen when click on “reserve” button after selected required data.

When user login, user can check history of what user reserve, and logout when finish their work. However, user can check history of each seat and view overview map of library.

The details in each pane will show later.

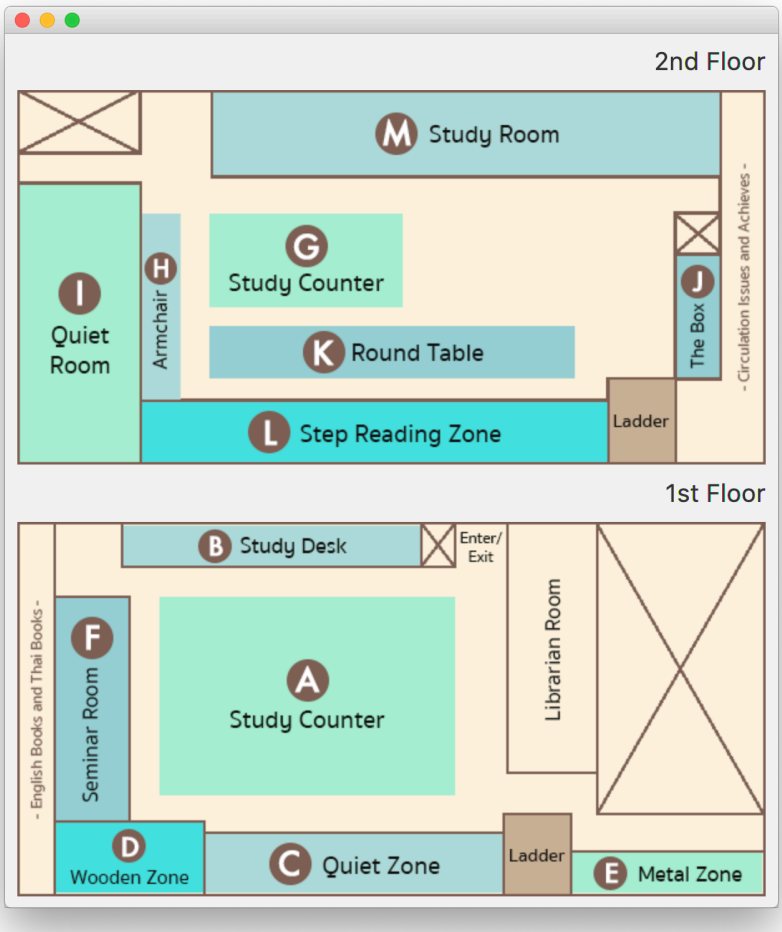
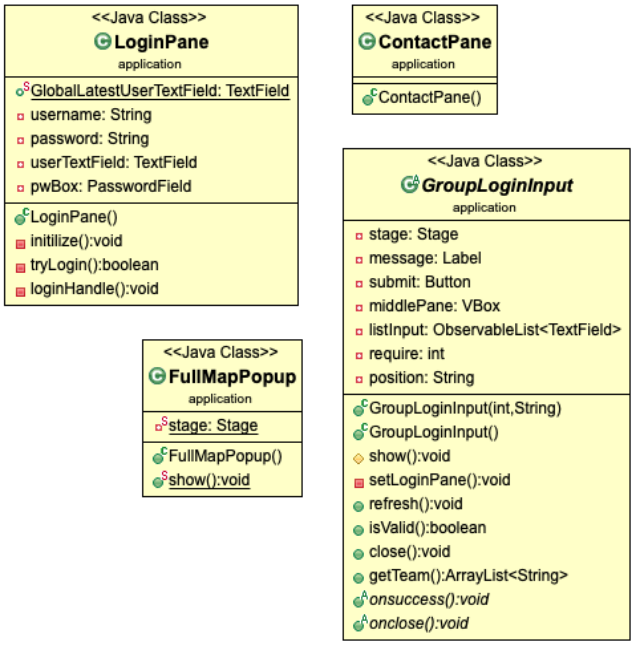


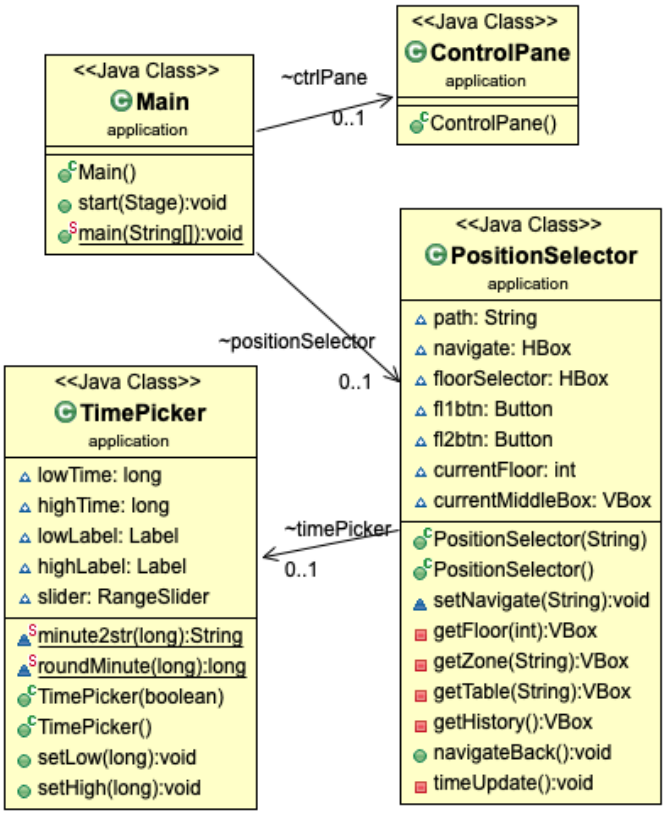
Fig 6: Overview map of library

2. Implementation Details

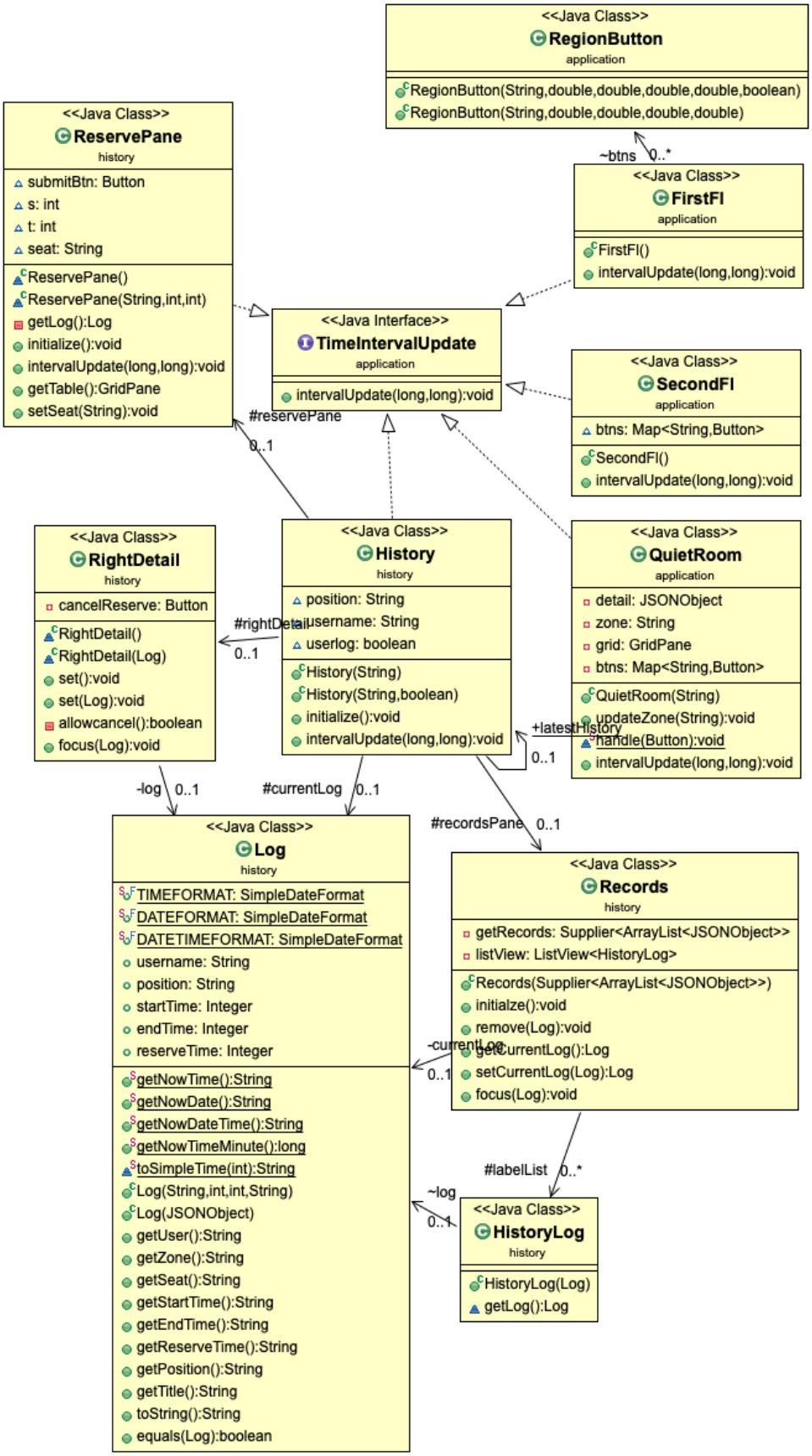
The diagram of the program is illustrated in figures below.



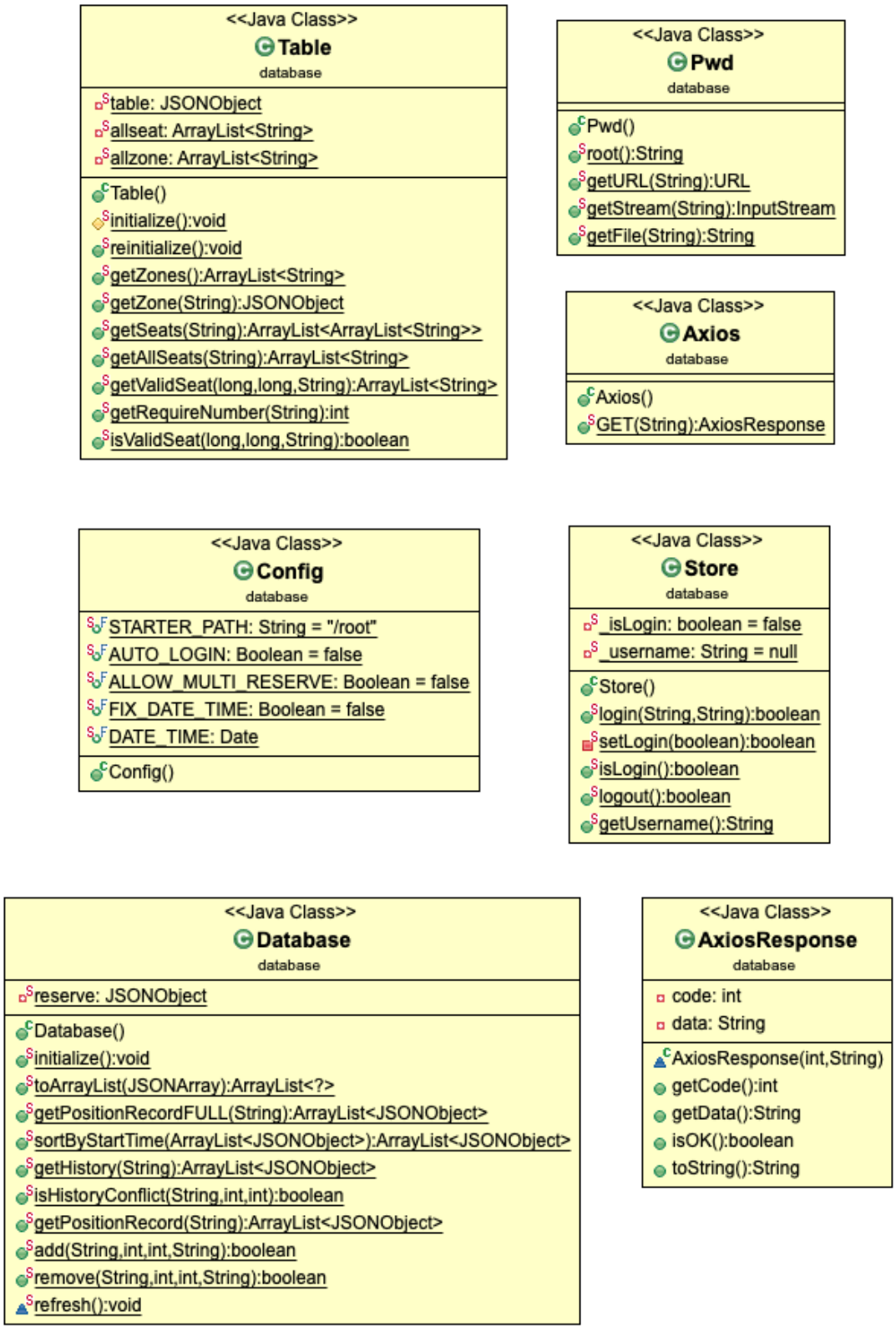
(a)



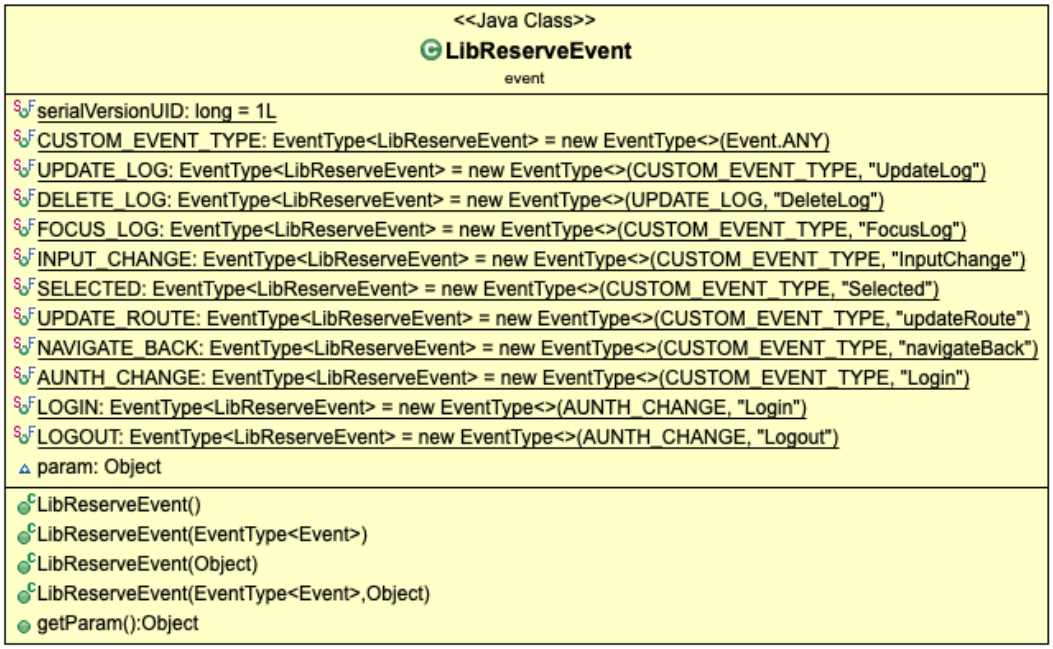
(b)


<<Java Class>>
RegionButton
application
RegionButton(String,double,double,double,double,boolean)
RegionButton(String,double,double,double,double)
~btns
0..*
<<Java Class>>
ReservePane
history
submitBtn: Button
s: int
t: int
seat: String
ReservePane()
ReservePane(String,int,int)
getLog():Log
initialize():void
intervalUpdate(long,long):void
getTable():GridPane
setSeat(String):void
<<Java Class>>
FirstFl
application
FirstFl()
intervalUpdate(long,long):void
<<Java Interface>>
TimeIntervalUpdate
application
intervalUpdate(long,long):void
#reservePane
0..1
<<Java Class>>
SecondFl
application
btns: Map<String,Button>
SecondFl()
intervalUpdate(long,long):void
<<Java Class>>
RightDetail
history
cancelReserve: Button
RightDetail()
RightDetail(Log)
set():void
set(Log):void
allowcancel():boolean
focus(Log):void
#rightDetail
0..1
<<Java Class>>
History
history
position: String
username: String
userlog: boolean
History(String)
History(String,boolean)
initialize():void
intervalUpdate(long,long):void
<<Java Class>>
QuietRoom
application
detail: JSONObject
zone: String
grid: GridPane
btns: Map<String,Button>
QuietRoom(String)
updateZone(String):void
handle(Button):void
intervalUpdate(long,long):void
+latestHistory
0..1
-log
0..1
#currentLog
0..1
#recordsPane
0..1
<<Java Class>>
Log
history
TIMEFORMAT: SimpleDateFormat
DATEFORMAT: SimpleDateFormat
DATETIMEFORMAT: SimpleDateFormat
username: String
position: String
startTime: Integer
endTime: Integer
reserveTime: Integer
getNowTime():String
getNowDate():String
getNowDateTime():String
getNowTimeMinute():long
toSimpleTime(int):String
Log(String,int,int,String)
Log(JSONObject)
getUser():String
getZone():String
getSeat():String
getStartTime():String
getEndTime():String
getReserveTime():String
getPosition():String
getTitle():String
toString():String
equals(Log):boolean
<<Java Class>>
Records
history
getRecords: Supplier<ArrayList<JSONObject>>
listView: ListView<HistoryLog>
Records(Supplier<ArrayList<JSONObject>>)
initialze():void
remove(Log):void
getCurrentLog():Log
setCurrentLog(Log):Log
focus(Log):void
-currentLog
0..1
#labelList
0..*
~log
0..1
<<Java Class>>
HistoryLog
history
HistoryLog(Log)
getLog():Log

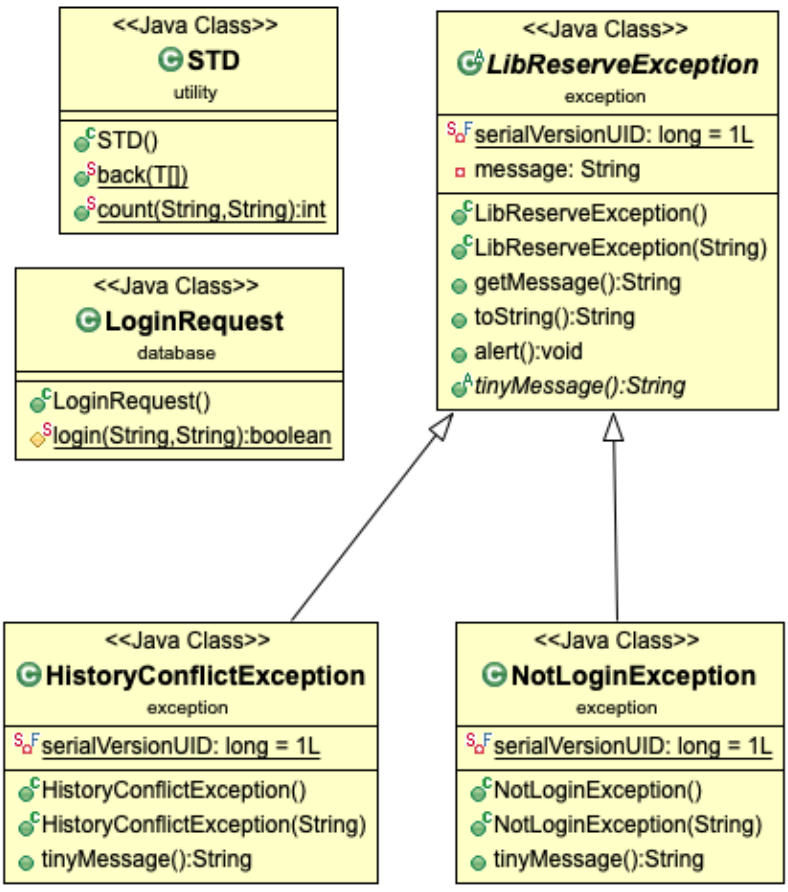
(c)


<<Java Class>>
Table
database
table: JSONObject
allseat: ArrayList<String>
allzone: ArrayList<String>
Table()
initialize():void
reinitialize():void
getZones():ArrayList<String>
getZone(String):JSONObject
getSeats(String):ArrayList<ArrayList<String>>
getAllSeats(String):ArrayList<String>
getValidSeat(long,long,String):ArrayList<String>
getRequireNumber(String):int
isValidSeat(long,long,String):boolean
<<Java Class>>
Pwd
database
Pwd()
root():String
getURL(String):URL
getStream(String):InputStream
getFile(String):String
<<Java Class>>
Axios
database
Axios()
GET(String):AxiosResponse
<<Java Class>>
Config
database
STARTER_PATH: String = "/root"
AUTO_LOGIN: Boolean = false
ALLOW_MULTI_RESERVE: Boolean = false
FIX_DATE_TIME: Boolean = false
DATE_TIME: Date
Config()
<<Java Class>>
Store
database
_isLogin: boolean = false
_username: String = null
Store()
login(String,String):boolean
setLogin(boolean):boolean
isLogin():boolean
logout():boolean
getUsername():String
<<Java Class>>
Database
database
reserve: JSONObject
Database()
initialize():void
toArrayList(JSONArray):ArrayList<?>
getPositionRecordFULL(String):ArrayList<JSONObject>
sortByStartTime(ArrayList<JSONObject>):ArrayList<JSONObject>
getHistory(String):ArrayList<JSONObject>
isHistoryConflict(String,int,int):boolean
getPositionRecord(String):ArrayList<JSONObject>
add(String,int,int,String):boolean
remove(String,int,int,String):boolean
refresh():void
<<Java Class>>
AxiosResponse
database
code: int
data: String
AxiosResponse(int,String)
getCode():int
getData():String
isOK():boolean
toString():String

(d)



(e)



(f)

Fig 7: The UML diagram of the program

The structure of the GUI can be broken down as in figures below.

Fig 8: The outline showing the core structure of the program

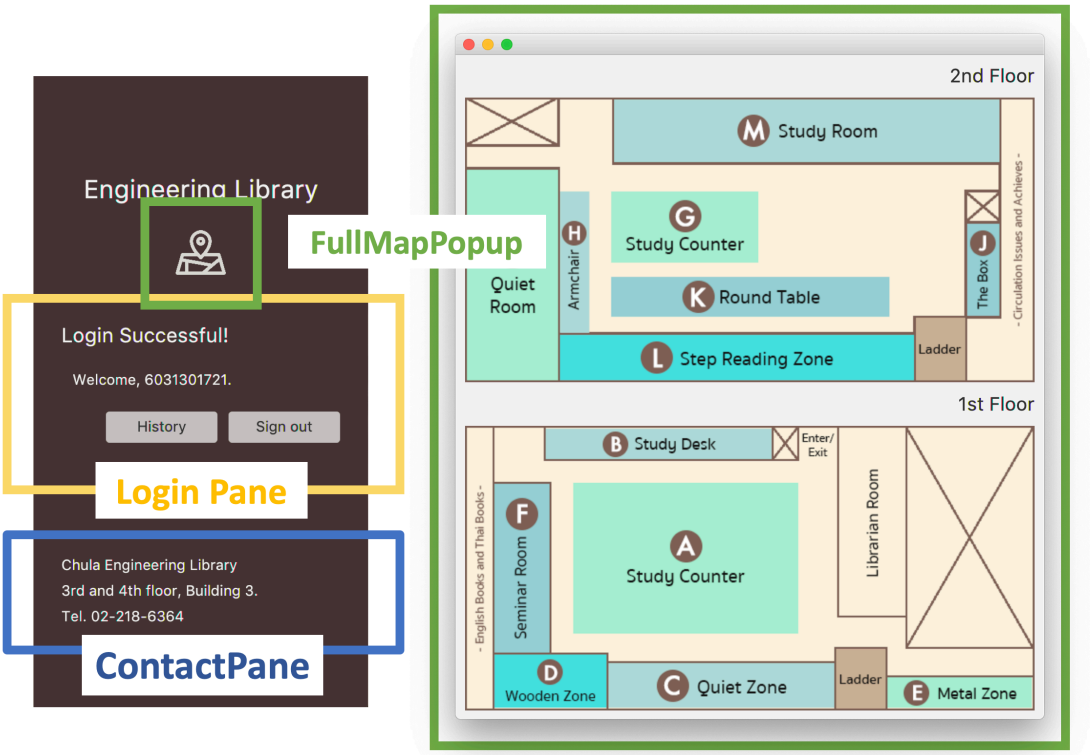


Fig 9: The outline showing components in Control Pane

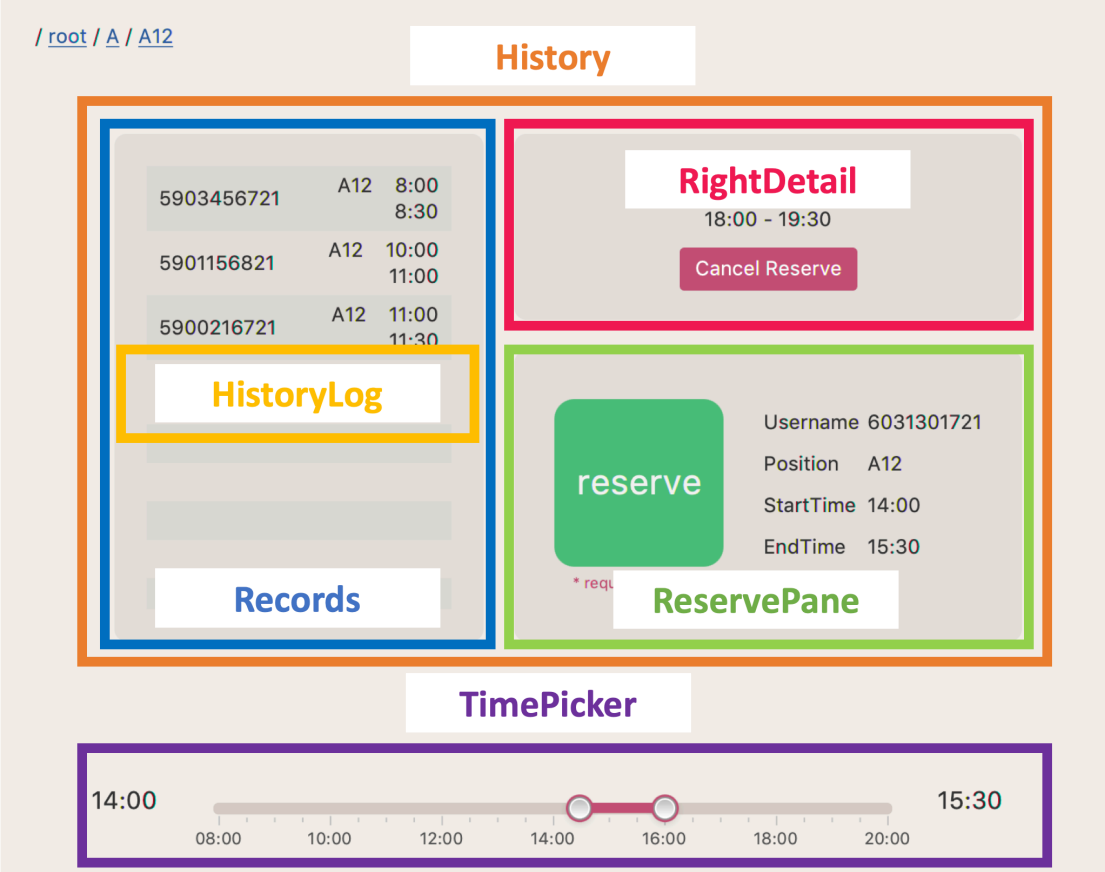


Fig 10: The outline showing components in History Pane

Access Modifier Notations

The Access Modifier Notations in this paper can be listed below.

Permission

+	public
#	protected
-	private

Keywords

bold	final
underlined	static
italic	abstract

Package application

Interface TimeIntervalUpdate

This interface is for make other class to object depend on the time.

Method

void intervalUpdate(long s, long t)	This will be injected by TimePicker.

Class FirstFl extends Pane implements TimeIntervalUpdate

This class is the 1st floor map, each zone on the pane is Button can navigate to each zone.

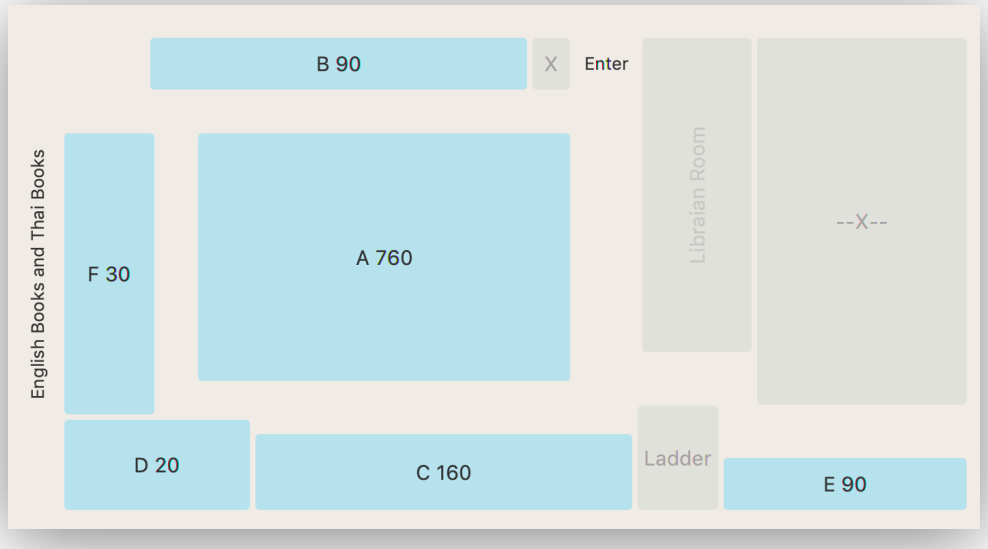


Fig 11: FirstFl when run application

Field

Map<String, RegionButton> btns	Map from String that is the name of zone to `RegionButton` that use for illustrating the zone.

Constructor

+ FirstFl()	Initializes the `FirstFl`. This method should: - Sets a width of 700 and a height of 350. - Instantiates each field: - `RegionButton` of each zone, `zoneA` to `zoneF` include the non-using zone, `x1`, `x2`,

	`ladder`, and `libr` (Librarian room), to illustrate the map of `FirstFl`. - Adds some Label to clarify each zone. - Adds each field to the pane as a children in the correct order. - Adds the zones to `btns` and adds `EventHandler` to each button, to make them navigate to each zone after click on. - Sets the style of each zone.

Method

+ void intervalUpdate(long s, long t)	Refresh the available seats of each zone during input time (s is start time and t is end time)

Class SecondFl extends Pane implements TimeIntervalUpdate

This class is the 2nd floor map, each zone on the pane is Button can navigate to each zone.

Fig 12: SecondFl when run application

Field

Map<String, RegionButton> btns	Map from String that is the name of zone to `RegionButton` that use for illustrating the zone.

Constructor

+ SecondFl()	Initializes the `SecondFl`. This method should: - Sets a width of 700 and a height of 350. - Instantiates each field: - `RegionButton` of each zone, `zoneG` to `zoneM` include the non-using zone, `x1`, `x2`, and `ladder` to illustrate the map of `SecondFl`. - Adds some Label to clarify each zone. - Adds each field to the pane as a children in the correct order. - Adds the zones to `btns` and adds `EventHandler` to each button, to make them navigate to each zone after click on. - Sets the style of each zone.

Method

+ void intervalUpdate(long s, long t)	Refresh the available seats of each zone during input time (s is start time and t is end time)

Class RegionButton extends Button

This helper class is used for creating a specific size button at specific place on pane.

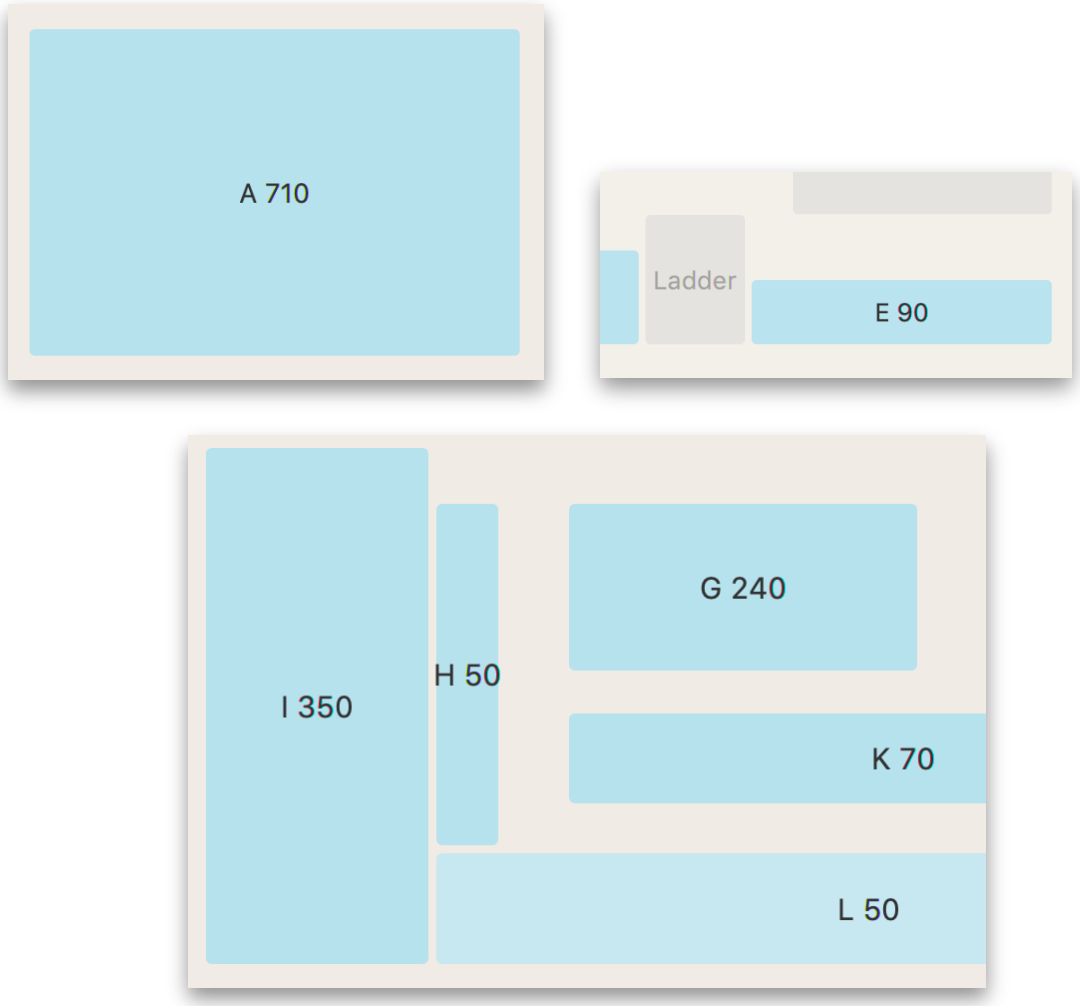


Fig 13: Zone button created by RegionButton

Constructor

+ RegionButton(String text, double sizeX, double sizeY, double posX, double posY, boolean isDisabled)	Creates Button with a text of text, a width of sizeX, a height of sizeY, a horizontal position of posX, and a vertical position of posY. If isDisabled is true, it is set Disable.
+ RegionButton(String text, double sizeX, double sizeY, double posX, double posY)	This constructor will call RegionButton(text, sizeX, sizeY, posX, posY, false)

Class QuietRoom extends VBox implements TimeIntervalUpdate

This class is used for creating a zone map that you desire.

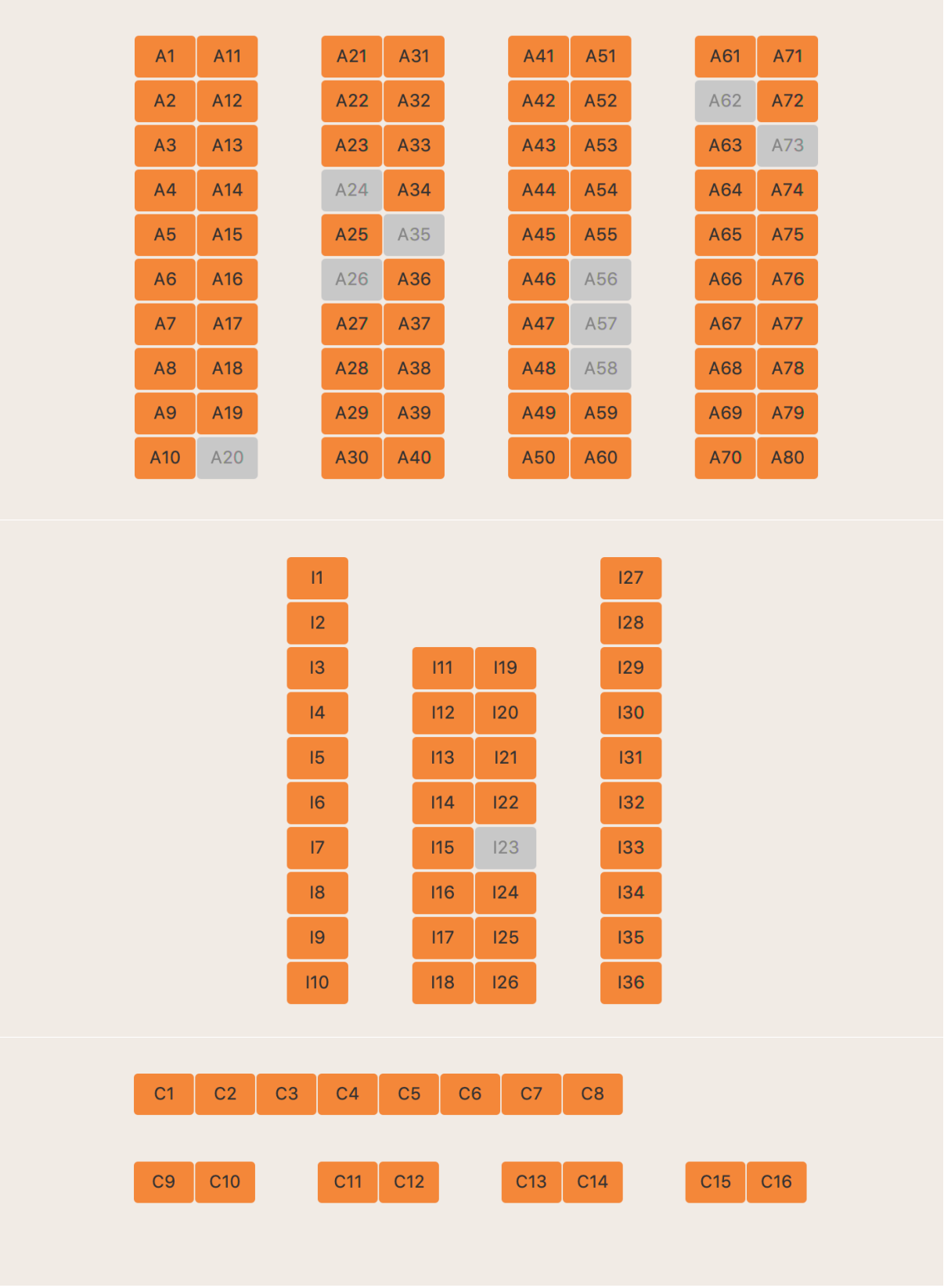


Fig 14: QuietRoom for illustrating zone map (top is A, middle is I, and bottom is C)

Field

- JSONObject detail	JSONObject that contains details of table in the zone.
- String zone	The zone that want to be generate the map.
- GridPane grid	The map of tables in the zone illustrate as a Buttons in `GridPane`.
- Map<String, Button> btns	Map uses for pair a name of a Button (seat name) and a Button.

Constructor

+ QuietRoom(String zone)	Initializes `QuietRoom` using VBox. This method should: - Set spacing of 10 - Run `updateZone(zone)`

Method

+ void updateZone(String zone)	Instantiates each field: - Sets `this.zone` to `zone` - `btns` is an empty `Map<String, Button>` - `grid` is a `GridPane` with horizontal gap of 1, vertical gap of 3. This is set alignment of the pane to `Pos.CENTER`. - Uses try-catch statement to generate Buttons from JSONArray that illustrate Table pattern. Each Button should store in `btns` to make them can be navigate to the reserve panel of each seat. - Clear all children in VBox and add `grid.`
void handle(Button btn)	If Button is clicked, the program will print “No button handle in QuietRoom.java”
+ void intervalUpdate(long s, long t)	Refresh the available seat within time interval that input.

Class TimePicker extends HBox

This class is use for creating slider for selecting start time and end time of reservation. When the slider is changed, it will make other class change.

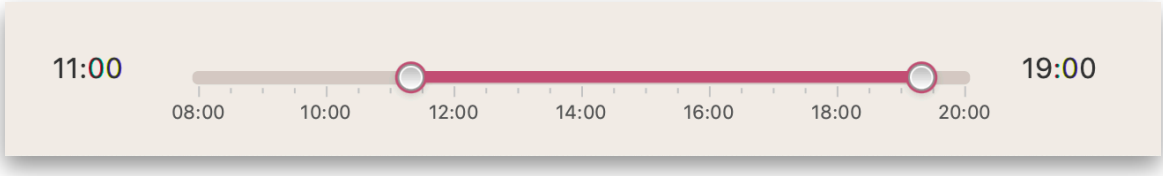


Fig 15: TimePicker that have been created

Field

long lowTime	Start time in term of minutes
long highTime	End time in term of minutes
Label lowLabel	Label of `lowTime` in time format
Label highLabel	Label of `highTime` in time format
RangeSlider slider	`RangeSlider` for selecting time interval

Constructor

+ TimePicker(boolean showLabel)	This method should: - Run `TimePicker()` - If `showLabel` is true, the `lowLabel` and `highLabel` is added to the HBox beside `slider`.
+ TimePicker()	Initializes the `TimePicker`. This method should: - Instantiate each fields: - `TimePicker` using a HBox and sets alignment of the pane to `Pos.CENTER`. - `mn` is a long int that is the Start Time in term of minutes. It should be current time but not lower than 480 (8:00) and not higher than 1200 (20:00). - `mx` is a long int that is the End Time in term of minutes. It should be 60 minutes after current time but should not exceed 1200 (20:00). - `lowTime` is set to `roundMinute(mn)` - `highTime` is set to `roundMinutes(mx)`

	- `lowLabel` is a Label with the text of time format of `lowTime`. - `highLabel` is a Label with the text of time format of `highTime`. - `slider` that is a `RangeSlider` should sets: - minimum number of 480 (8:00) - maximum number of 1200 (20:00) - selected the value on the `RangeSlider` between `lowTime` and `highTime`. - a padding inset of 15, 25, 0, and 25 - `lowValue` to `lowTime` - `highValue` to `highTime` - a width of 500 - a height of 50 - show Tick Labels - show Tick Marks - set Major Tick Unit of 120 - set Label Formatter that can convert number to time format String. - add Listeners of `lowValue` and `highValue` to `fireEvent` `INPUT_CHANGE` for refresh all nodes that associate with time interval. - Adds `slider` to the pane as a children.

Method

String minute2str(long l)	Convert `l` (minutes) to time format (h:mm)
long roundMinute(long l)	Round `l` to the minutes that can be divide by 30.
+ void setLow(long l)	Sets `lowTime` and `lowLabel` of `TimePicker` and `lowValue` of `slider` equal to the value of `l`.
+ void setHigh(long l)	Sets `highTime` and `highLabel` of `TimePicker` and `highValue` of `slider` equal to the value of `l`.

Class PositionSelector extends VBox

This class is the pane can change to floor map, zone map and history pane, use for helping select the required data.

Fig 16: PositionSelector when “Floor 1” button selected

Field

String path	String that describe what current place are shown on the window. This can be “/root/history” (History of reservation for each user) and “/root[/{zone}[/{seat}]]”
HBox navigate	HBox that contains a Label (or many Labels). Each label has a text of the zone, the seat, “root”, or “history” and can click to navigate (change the place that are shown on window) to the place that are shown on it.
TimePicker timePicker	An instance of `TimePicker`

HBox floorSelector	HBox that contains `fl1btn` and `fl2btn` Buttons.
Button fl1btn	Button that change to First Floor Map.
Button fl2btn	Button that change to Second Floor Map.
int currentFloor	The floor number 1 or 2 (Starting from floor 1) that are shown on the window.
VBox currentMiddleBox	VBox that show a map of floor 1 or 2, a map of zone, a reserve history of seat with the `reservePane`, or a reserve history of user with the `reservePane`.

Constructor

+ PositionSelector(String path)	This constructor is call `PositionSelector()` constructor and method `setNavigate(path)`
+ PositionSelector()	Initializes the `PositionSelector`. This method should: - Sets the alignment of the pane to `Pos.TOP_CENTER`, the pane has a width of 720 and a height of 570. - Sets the inset padding of 15, 20, 15, and 20. - Sets the vertical gap of the pane to 20. - Sets path to “/root”. - Instantiates each fields: - `navigate` is a HBox that is set the alignment of the pane to “`Pos.CENTER_LEFT`” and the inset padding of 15, 20, 15, and 20. - `floorSelecter` is a HBox that is set the spacing of 10, the width of 700, and the alignment of the pane to “`Pos.CENTER`”. - `fl1btn` is a Button with a text “Floor 1”. - `fl2btn` is a Button with a text “Floor 2”. - `timePicker` is a `TimePicker` with a Label that can change the time interval while using slider. When user changes the time, the node that associated with should be refresh to this time interval. - Adds each field to the pane as a children in the correct position.

	- Adds `EventHandler` for the `fl1btn` and `fl2btn` Button. - Run `setNavigate("/root")`

Method

void setNavigate(String nav)	Changes the `currentMiddleBox` to the desired place that input to the method. This method should: - Sets path as input `nav` and gets the smallest place (deepest path or the path after the last "/" in the sentence) into `back`. - Clears all children of `navigate`. - Instantiates each fields: - `level` that is an int is set to 0. - If `nav` has "/root" at start of the String - a Label with a text " / " is added to `navigate`, followed by root that is a Label with a text "root" - The label is added `EventHandler` that can navigate to "/root" (can run `setNavigate("/root")`. - `level` is set to `level bitwise OR 1`. - `currentMiddleBox` is Floor Map (`getFloor(currentFloor)`) - If `nav` has "/root/" followed by alphabet A to Z at start of the String - a Label with a text " / " is added to `navigate`, followed by `zoneLabel` that is a Label with a text from the first character of `back`. - The label is added `EventHandler` that can navigate to "/root/" + text of zoneLabel (can run `setNavigate("/root/" + (back.substring(0,1)))` - `level` is set to `level bitwise OR 2`. - `currentMiddleBox` is Zone Map (`getZone(back)`) - If `nav` has "/root/" followed by alphabet A to Z, followed by "/", followed by alphabet A to Z and followed by number 0 to 9 at start of the String

	- a Label with a text " / " is added to navigate, followed by seatLabel that is a Label with a text from back. - level is set to level bitwise OR 4. - currentMiddleBox is Reservation Panel with Reservation History of this seat (getTable(back)). - If nav is "/root/history" - a Label with a text " / " is added to navigate, followed by historyLabel that is a Label with a text "history" - level is set to level bitwise OR 8. - currentMiddleBox is Reservation History of the user that have login already (getHistory()) - hb is a HBox that contains currentMiddleBox. This is set the width of 800, the height of 350 and the alignment of the pane to Pos.CENTER. - Hides floorSelector when currentMiddleBox not be navigate to any zone or any floor. - Hides timePicker when currentMiddleBox is navigate to "/root/history" - run timeUpdate()
- VBox getFloor(int i)	Returns VBox that contains Floor Map that match to the input i. This method should: - Instantiates floor1 with FirstFl and floor2 with SecondFl - Adds EventHandler to the zone Button to make each Button in the Floor Map can be navigate to each zone correctly. - If i is 1 return floor1, if not return floor2.
- VBox getZone(String zone)	Returns VBox that contains Zone Map that match to the input zone. This method should: - Instantiates VBox that contains QuietRoom of specific zone and Adds EventHandler to the seat Button in QuietRoom to make each Button can be navigate to each seat correctly.

- VBox getTable(String seat)	Returns VBox that contains Reservation History of the input seat. This method should:
	- Instantiates `historyPane` with History of input seat.
	- Adds 2 `EventHandlers` to `historyPane`
	  - To refresh `historyPane` when there is new reserve record.
	  - To change the range of `RangeSlider` in `TimePicker` to the `startTime` and `endTime` of the History record that user has selected.
	- Returns VBox of `HistoryPane`.

- VBox getHistory()	Returns Reservation History of user that have login already.
+ void navigateBack()	Sets `navigate` upper for 1 step. In other words, sets `navigate` to `path` that remove "/" and `back` from the end of `path`.
- void timeUpdate()	Refreshes every node in `currentMiddleBox` that implements from `TimeIntervalUpdate` to value within time interval of `timePicker`.

Abstract Class GroupLoginInput

Some seat require many user to process reservation, this component will receive list of that user.

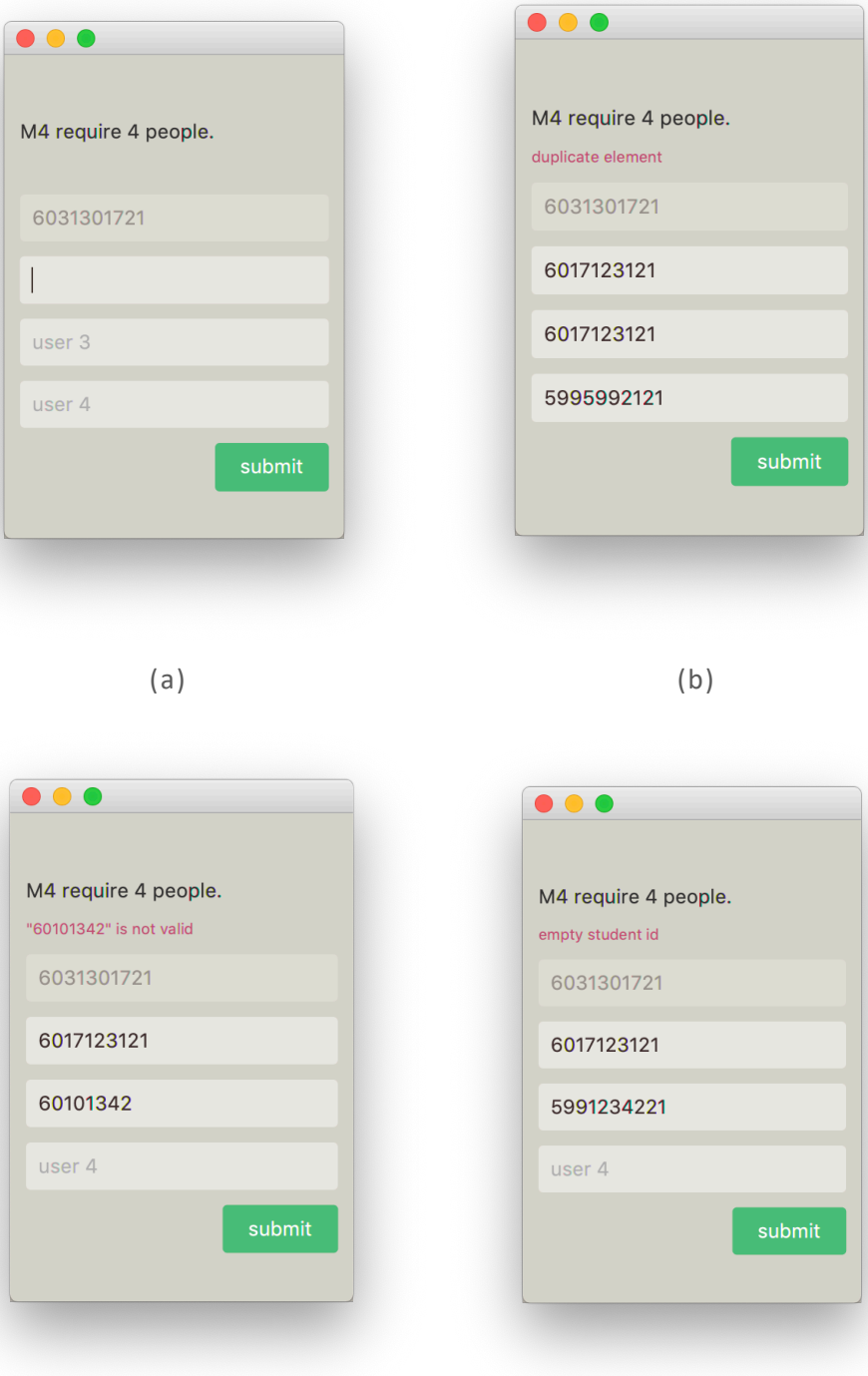


Fig 17: GroupLoginInput for input IDs of people
(a) when created, (b) when input duplicate ID,
(c) when ID is not valid, and (d) when leave the field blank

Field

- Stage stage	Initializes Stage
- Label message	Initializes message using Label
- Button submit	Initializes Button with a text “submit”
- VBox middlePane	Initializes middlePane using VBox
- ObservableList<TextField> listInput	List of TextField of student ids
- int require	Number of people require for reserve seat
- String position	Seat position

Constructor

+ GroupLoginInput(int n, String pos)	This method should: - Sets `require` to `n` - Sets `position` to `pos` - run `show()`

Method

# void show()	Show the window that have `textFields` in `listInput` and “submit” Button. This method should: 	
	- Instantiates each field:
	  - `pane` is a GridPane with a padding inset of 30, 10, 30, and 10, a vertical gap of 10. This is set alignment of the pane to `Pos.CENTER`.
	  - Clear data in `listInput`.
	  - Add `textFields` `require` times in `listInput`.
	  - Set Text at First `TextField` to `username` and Disable this `TextField`.
	  - Run `setLoginPane()`
	  - Add Label to describe the people that required in this position to `pane`.
	- Adds each field to the pane as a children of `pane` in the correct position.
	- Instantiates `Scene` with a Scene that contains `pane`.
	- set `scene` to `stage` and show `stage`.

- void setLoginPane	This method should:

	- Clear all children of `middlePane`
	- Sets `middlePane` a width of 70 and a height of 100.
	- If user login
	  - adds `listInput` to `middlePane`
	  - adds `EventHandler` to run `handle()` when user click on `submit` Button, only if `isValid()` is true, if not, `message` is set text as "Input format is not valid. Please check your student ID"
	- If user not login, `message` is set text as "Please login"

+ void refresh()	This method should: - Clear `message` text if there are any text in `message`. - if user already login and there is no `textField` in `listInput`, run `setLoginPane()`
+ boolean isValid()	Return true if text in `textFields` in `listInput` is contains only numbers.
+ void close()	Close the window
+ ArrayList<String> getTeam()	Return username of the group that input in `textFields` in `listInput.`
+ void handle()	

Class ContactPane extends VBox

This class is the contact card in ControlPane. It contains 3 Labels.

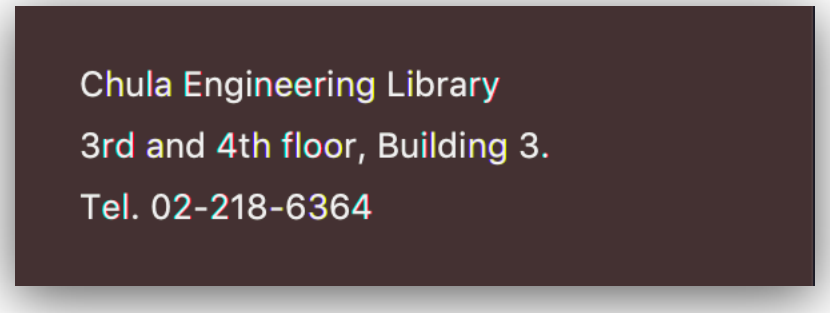


Fig 18: ContactPane in ControlPane

Constructor

+ ContactPane()	Initializes the `ContactPane`. This method should: - Instantiates each fields: - location is a Label with a text “Chula Engineering Library”. - position is a Label with a text “3rd and 4th floor, Building 3.” - phone is a Label with a text “Tel. 02-218-6364” - Sets the alignment of the pane to Pos.CENTER_LEFT, the pane has a width of 300. - Sets the spacing of 6 and the inset padding of 25. - Sets styles of the pane - Adds each field to the pane as a children in the correct order.

Class FullMapPopup

Class that will pop up the map with right scale help user visualizing.

Fig 19: Map button on ControlPane

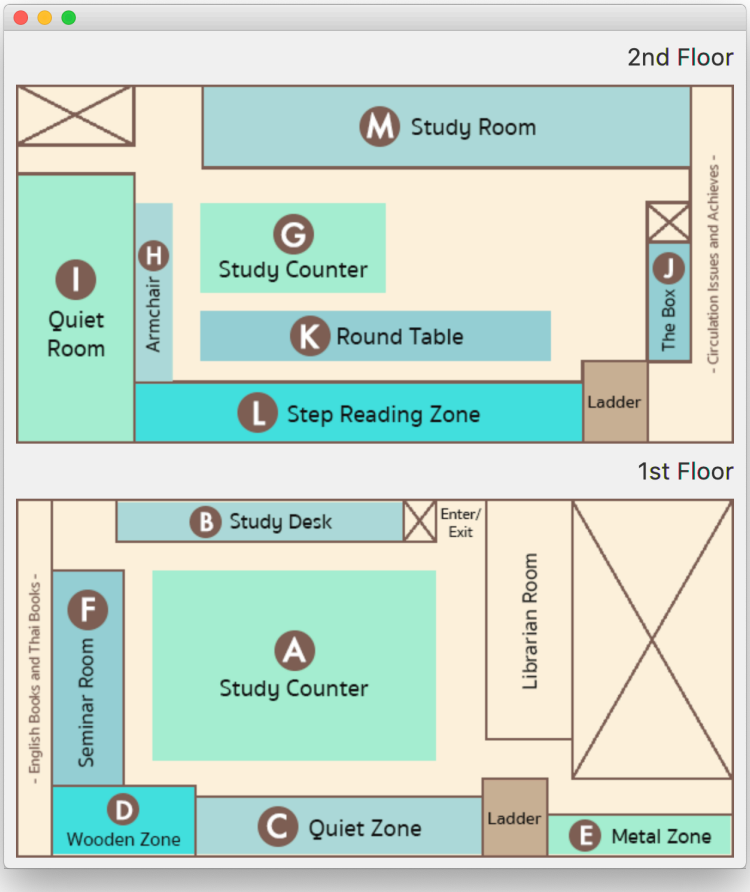


Fig 20: The window that pop up when click the button

Field

- Stage stage	Initializes Stage

Method

+ void show()	Shows the Full Map of Librarian. This method should: - Instantiates each field: - `fullmap` is a VBox with a padding inset of 10, a spacing of 10. This is set alignment of the pane to `Pos.CENTER_RIGHT`. - `firstFlL` is a Label with a text “1st Floor”. This is set font size of 20, a bold font weight, and alignment of the pane to `Pos.CENTER_RIGHT`. - `secondFlL` is a Label with a text “2nd Floor”. This is set font size of 20, a bold font weight, and alignment of the pane to `Pos.CENTER_RIGHT`. - `firstFl` and `secondFl` is `ImageView` that load the image of Floor Map of each floor. - Adds each field to the pane as a children of `fullMap` in the correct position. - Instantiates `mapScene` with a Scene that contains `fullMap`. This is set a width of 620 and a height of 700. - set `mapScene` to `stage` and show `stage`.

Class ControlPane extends VBox

This class is generating the pane contains titles of program, map button to show the library map, loginPane, and the name card. This class is the one of the two main class of program

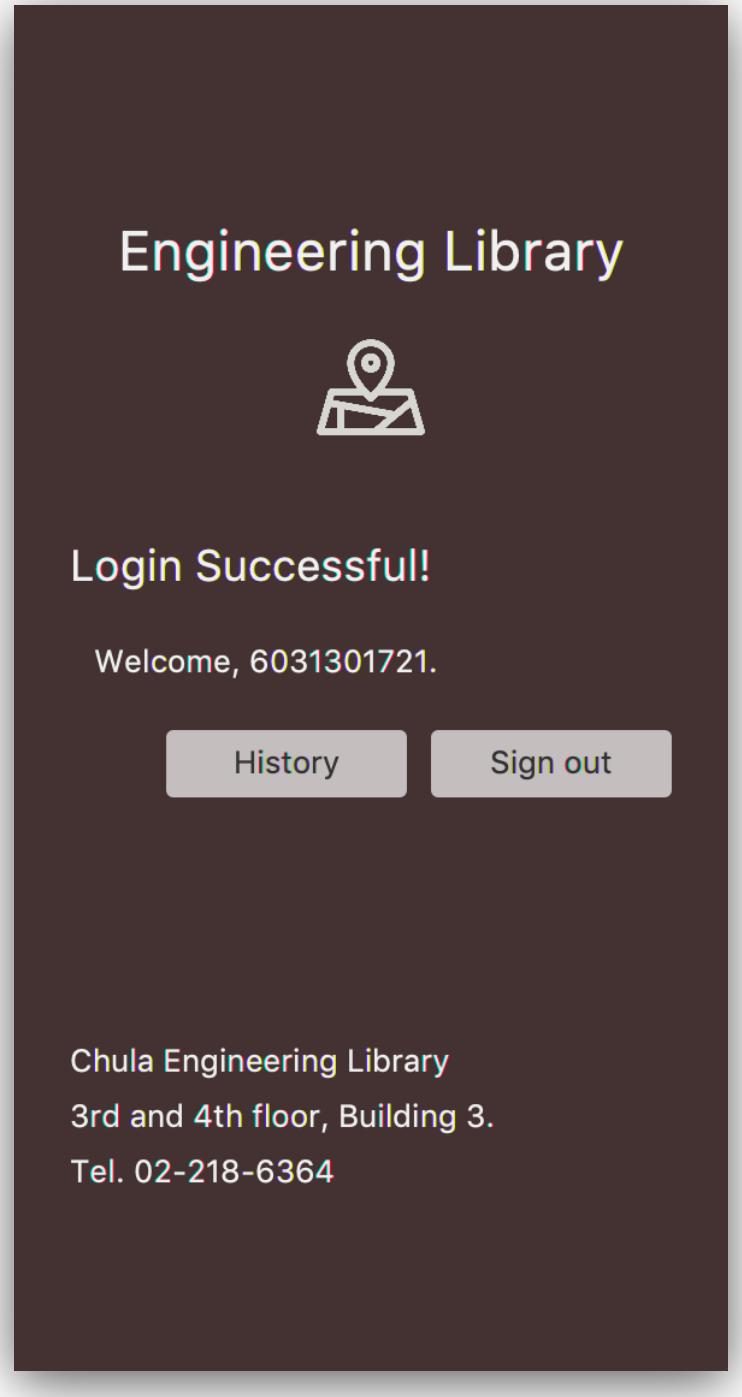


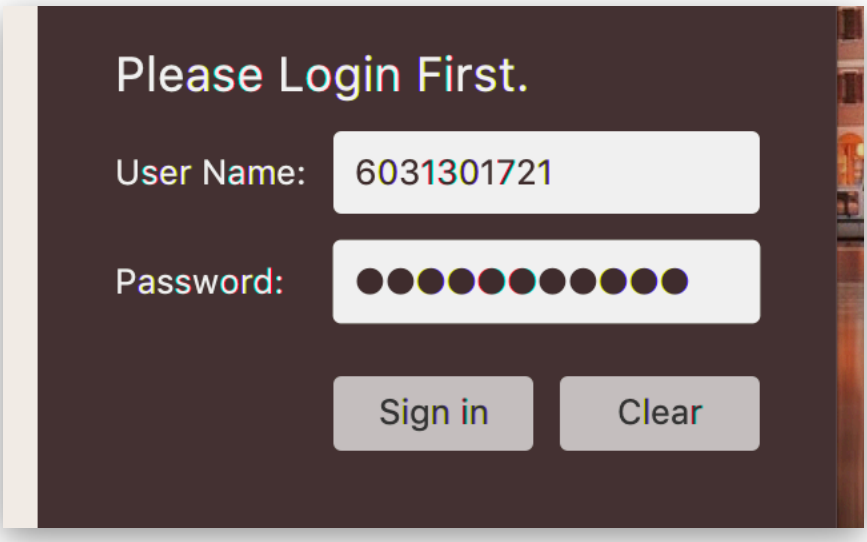
Fig 21: LoginPane when user have login

Constructor

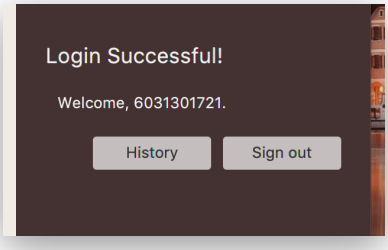
+ ControlPane()	Initializes the `ControlPane`. This method should: - Sets the alignment of the pane to `Pos.CENTER`, the pane has a width of 500. - Sets the inset padding of 90, 0, 50, and 0. - Instantiates each field: - `title` is a Label with the text "Engineering Library" that has a font size of 23 and a bold font weight - `wrapper` is a HBox that contains `mapBtnPane` and is set alignment to `Pos.CENTER` - `mapBtnPane` is a VBox with a `mapBtn` inside, this pane should set the insets padding of 15, the margin of 15, 10, 10, and 10, and sets alignment to `Pos.CENTER_RIGHT`. - `mapBtn` is a Button with a `mapLogo` graphic that has a width of 45 and a height of 45. This Button should pop up the map of the library after click on. - `loginPane` is a `LoginPane` that use for user login. - `rg` is a `region` that is set vertical grow always. - a `ContactPane`. - Adds each field to the pane as a children in the correct order.

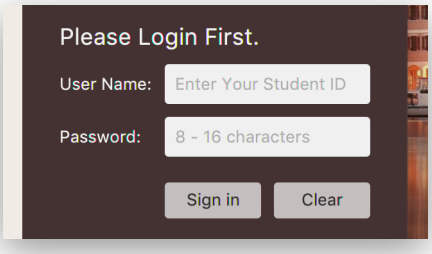
Class LoginPane extends GridPane

Component that will authenticate user before process any seat request.



(a)





(b) (c)

Fig 22: LoginPane when (a) input the username and password (b) have login (c) generated

Field

+ TextField GlobalLatestUserTextField	Object to refer to `userTextField`.
- String username	User's username (Student ID)
- String password	User's password (Password of CUNET account)
- TextField userTextField	`TextField` for receiving the username
- PasswordField pwBox	`PasswordField` for receiving the password

Constructor

+ LoginPane()	Initializes the LoginPane. This method should: - Sets the alignment of the pane to `Pos.TOP_CENTER`, the pane has a width of 300. - Sets the inset padding of 25. - Sets the horizontal gap and vertical gap of the pane to 10. - Run the `initilize()`.

Method

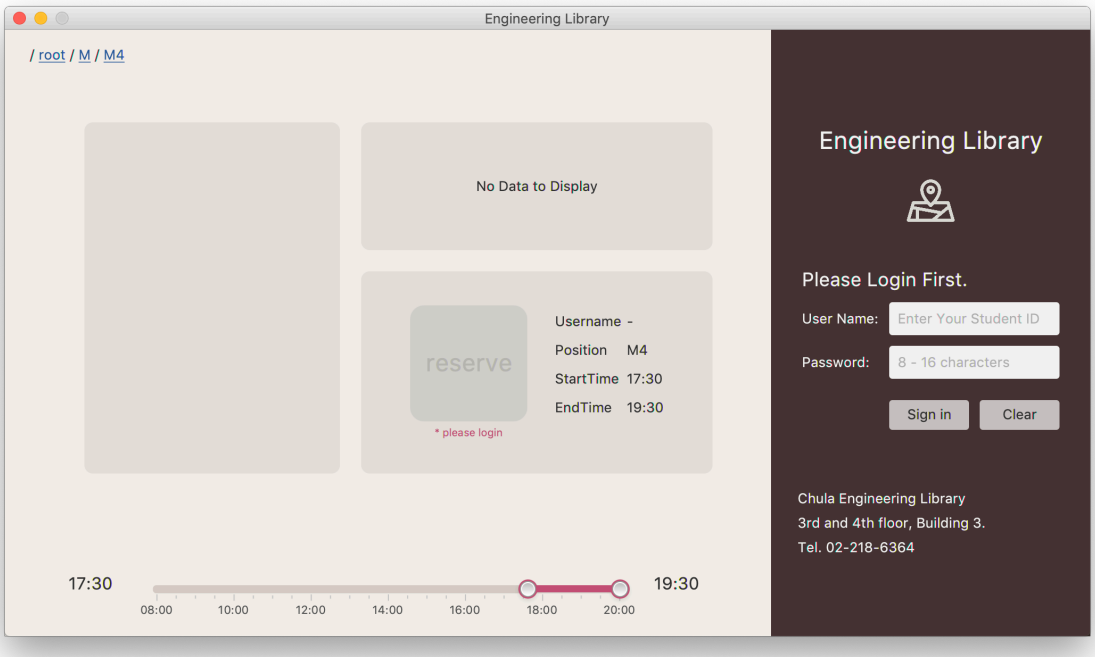
- void initilize()	Initializes the field that need to use while login. This method should: - Clears the `LoginPane`. Instantiates each field: - `scenetitle` is a Label with the text "Please Login First.". This is a header of the `LoginPane`. This is set a font size of 18 and a bold font weight. - `userName` is a Label with the text "User Name:" to clarify the `userTextField`. - `userTextField` is a `TextField` that is set a promptText as "Enter Your Student ID". - `GlobalLatestUserTextField` is set the value as same value as `userTextField.` - pw is a Label with the text "Password:" to clarify the pwBox. - pwBox is a `PasswordField` that is set a promptText as "8 - 16 characters". - hbBtn is a HBox that contains `signinBtn` and `clearBtn`. This is set the spacing to 10 and alignment of the pane to `Pos.BOTTOM_RIGHT`. - `signinBtn` is a Button that run `tryLogin()` when user click on it. This has the text "Sign in" and a width of 75. - `clearBtn` is a `Button` that clear the text in `userTextField` and pwBox. This has the text "Clear" and a width of 75. - Adds `EventHandler` for each fields:

	- pwBox can do `trylogin()` when user press ENTER key. - `signinBtn` can do `trylogin()` when user click on. - `clearBtn` can clear the text in `userTextField` and pwBox when user click on. - Adds each field to the pane as a children in the correct position.
- boolean tryLogin()	Doing the Login process using the username and password input. This method should: - Disable `userTextField` and pwBox. - Get Text from `userTextField` and pwBox to store in `username` and `password` - Run `Store.login(username, password)` method to check that the `username` and `password` are correct. - if it is correct, show the Alert "Login Successful!", run `loginHandle()`, and return true. - if not, show the Alert "Wrong Username of Password!", enable `userTextField` and pwBox, and return false.
- void loginHandle()	Initializes the field that show the profile info when login successful. This method should: - Clears the `LoginPane`. - Instantiates each field: - `success` is a Label with the text "Login Successful!". This is a header of the `LoginPane`. This is set a font size of 18, a bold font weight, and a width of 300. - `wellcome` is a Label with the text "Welcome {username}." that have inset padding of 10 This use for showing who is login. - `bottonPane` is a HBox that contains `signoutBtn` and `historyBtn`. This is set the spacing to 10 and alignment of the pane to `Pos.BOTTOM_RIGHT`. - `signoutBtn` is a `Button` that run `Store.logout() and initilize()` when user click on it (user can be login again). This has the text "Sign out".

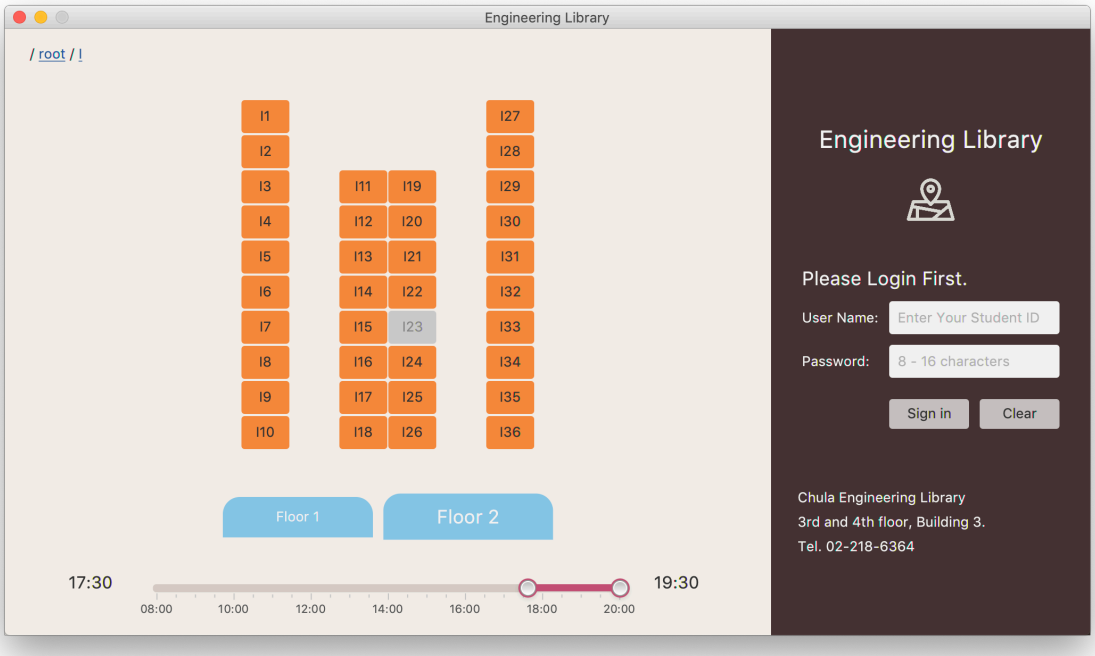
	- `historyBtn` is a `Button` that show the history of reservation of this user when user click on it (user can be login again). This has the text “History”. - Add `EventHandler` to bypass the `LOGIN` and `LOGOUT` event from `LoginPane`. - Adds each field to the pane as a children in the correct position.

Class Main extends Application

Main stage for showing left and right panel.



(a)



(b)

Fig 23: Main application

Field

PostionSelector positionSelector	An instance of `PostionSelector`
ControlPane ctrlPane	An instance of `ControlPane`

Method

+ void start(Stage primaryStage)	The main entry point of the JavaFX application. This method should: - Creates a `root` container using `GridPane`. - Initializes `PostionSelector` with `Config.STARTER_PATH` and `ControlPane` and add them to the root container. - Sets the style for the scene using “style.css” - Sets the scene with title “Engineering Library”, not resizable and then show the stage. - Adds `EventHandler` to bypass the UPDATE_ROUTE Event from `ControlPane` to `PositionSelector`.
+ void main(String[] args)	An entry point of the application.

CSS Style for application (style.css)

Use for add style to the components in the application package.

Package database

Class Axios

Helper class for write GET require easier.

Method

+ AxiosResponse GET(String url)	Method for get the response from HTTP request.

Class AxiosResponse (Helper class of Axios)

Response of HTTP request with status code and body.

Field

- int code	HTTP status code receive from request (eg. success = 200, bad request = 400).
- String data	Body of HTTP request as String.

Constructor

AxiosResponse(int code, String data)	Constructor of `AxiosResponse` setting to code & data without any checking.

Method

+ int getCode()	Getter method for `code` field.
+ String getData()	Getter method for `data` field.
+ boolean isOK()	Check if HTTP response status code in `code` is accepted by or not. Accept if `code = 2xx`.
+ String toString()	Return `code` and `data` as String format.

Class Config

Note: This class are using for helping while developing software.

Field

+ String STARTER_PATH	The constant value for `STARTER_PATH` is set to be “/root”.
+ boolean AUTO_LOGIN	The constant value for `AUTO_LOGIN` is set to be “false”.

+ boolean ALLOW_MULTI_RESERVE	The constant value for ALLOW_MULTI_RESERVE is set to be “false”.
+ boolean FIX_DATE_TIME	The constant value for FIX_DATE_TIME is set to be “false”.
+ Date DATE_TIME	The constant value for DATE_TIME is sets as “2018-07-06 10:00” if FIX_DATE_TIME is true. If not, this should be current date and time.

Class Database

Main class to connecting with server and HTTP request.

Field

- JSONObject reserve	JSONObject that contains all of reservation.

Method

+ void initialize()	Get data from JSONObject to store in reserve.
+ ArrayList<?> toArrayList(JSONArray arr)	Return ArrayList that converted from JSONArray.
+ ArrayList<JSONObject> sortByStartTime(ArrayList<JSONObject> arr)	Return sorted ArrayList of JSONObject by StartTime.
+ ArrayList<JSONObject> getPositionRecordFULL(String key)	Get JSONObject that contains the full details of reservation in the seat key and return ArrayList of JSONObject, sorted by StartTime
+ ArrayList<JSONObject> getHistory(String username)	Get JSONObject that contains the full details of reservation of username and return ArrayList of JSONObject, sorted by StartTime
+ boolean isHistoryConflict(String username, int start, int end)	Return true, if username has reservation in the time interval input already. If not, return false.
+ ArrayList<JSONObject> getPositionRecord(String key)	Get JSONObject that contains the details of reservation (without user details in reservation) in the seat key and return ArrayList of JSONObject, sorted by StartTime

+ boolean add(String username, int startTime, int endTime, String position) throws NotLoginException, HistoryConflictException	Send HTTP Request to add the query of reservation, to make the reservation. Make sure that user have login and don't have reservation at the time interval that user request. If process is success, this method will return true.
+ boolean remove(String username, int startTIme, int endTime, String position)	Send HTTP Request to remove the query of reservation, to cancel the reservation. If process is success, this method will return true.
void refresh()	Refresh the data in `Database` class and `Table` class by using `initialize()`

Class LoginRequest

Helper class to request authentication.

Method

# boolean login(String username, String password)	Send `username` and `password` via HTTP request to external API. This will return true if `username` and `password` are correct.

Class Pwd

Helper class to get directory path.

Field

+ String root	Full path directory of current project.
+ String file	Full path directory of current project with prefix "`file:`"

Class Store

Wrapper for all authentication event.

Field

- boolean _isLogin	The login state of this application. This is set to "false" at start.
- String _username	The username that user login to this application. This is set to "null" at start.

Method

+ boolean login(String username, String password)	if `LoginRequest(username,password)` are true, set `_username` to `username` and run and return setLogin(true). If not, set `_username` to null and run and return setLogin(false).
- boolean setLogin(boolean b)	Store the login state in `_isLogin` as input b and return the same value.
+ boolean isLogin()	Return `_isLogin`.
+ boolean logout()	- Sets `_username` to null - If `isLogin()` is true, run `setLogin(false)` and return true. If not, run `setLogin(true)` and return false.
+ String getUsername()	If `isLogin()` is true, return `_username`, else, return null.

Class Table

Containing all static seat detail eg. require number, position on grid.

Field

- JSONObject table	JSONObject for store the seat patterns in library
- ArrayList<String> allseat	ArrayLst for store every seats in library
- ArrayList<String> allzone	ArrayLst for store every zones in library

Method

+ void initialize	Read the JSON file and convert them into JSONObject, ArrayList<String> in `table`, `allseat`, and `allzone` varaibles
+ void reinitialize	Call initialize()
+ ArrayList<String> getZones()	Return allzone
+ JSONObject getZone(String zone)	Return the seat pattern of input `zone`
+ ArrayList<ArrayList<String>> getSeats(String zone)	Return the seat pattern of input `zone` as 2D ArrayList

+ ArrayList<String> getAllSeats(String zone)	Return the seat pattern of input `zone` as 1D ArrayList
+ ArrayList<String> getValidSeat(long s, long t, String search)	Return list of seats (from `allseat`) that interval `[s, t)` can be place to that *seat* in with out collition. Check by function `Database.getPositionRecord(`*seat*`)`
+ int getRequirenumber(String pos)	Return table[pos.at(0)][“require”]
+ boolean isValidSeat(long s, long t, String position)	Return true if `s != t` and `position` appear in `getValidSeat(s, t, position)`

JSON File (table.json)

Store table pattern of each zone in JSONObject format.

Package event

Class LibReserveEvent extends Event

This class is using to provide an event to communicate with direct children component that specifically relate to the program logic.

Field

+ long serialVersionUID	
+ EventType<LibReserveEvent> CUSTOM_EVENT_TYPE	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> UPDATE_LOG	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> DELETE_LOG	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> FOCUS_LOG	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> INPUT_CHANGE	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> SELECTED	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> UPDATE_ROUTE	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> NAVIGATE_BACK	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> AUNTH_CHANGE	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> LOGIN	Declare the EventType to use for specific group of Event.
+ EventType<LibReserveEvent> LOGOUT	Declare the EventType to use for specific group of Event.
Object param	

Constructor

+ LibReserveEvent()	This method run `LibReserveEvent(null)`

+ LibReserveEvent(EventType<? extends Event> typ)	This method run `LibReserveEvent(typ, null)`
+ LibReserveEvent(Object param)	This method run `LibReserveEvent(CUSTOM_EVENT_TYPE, null)`
+ LibReserveEvent(EventType<? extends Event> typ, Object param)	This method run `Event(typ)` and sets `this.param` to `param`

Method

+ Object getParam()	Return `param`.

Package exception

Class HistoryConflictException extends LibReserveException

Field

- long serialVersionUID	

Constructor

+ HistoryConflictException()	This method run HistoryConflictException("History Conflict is occur. One user can reserve 1 table at same time.")
+ HistoryConflictException(String message)	Sets `message` of superclass (`LibReserveException`) to `message`.

Method

+ String tinyMessage()	Return "Reserve time is conflict to previous record. Please check your history."

Abstract Class LibReserveException extends Exception

Field

- long serialVersionUID	
- String message	The error message of the Exception.

Constructor

+ LibReserveException()	This method run LibReserveException("LibReserve Exception is occer")
+ LibReserveException(String message)	Sets `this.message` to `message`.

Method

+ String getMessage()	Return `message`.
+ String toString()	Return `getMessage()`.
+ void alert()	- Show Error Alert with a text of tinyMessage(). A title of Alert is “Error” and the Header Text is “Cannot Reserve”. - When close the Alert, the cursor will focus on userTextField for helping user login.
+ String tinyMessage()	Return message in tiny version if possible.

Class NotLoginException extends LibReserveException

Field

- long serialVersionUID	

Constructor

+ NotLoginException()	This method run NotLoginException(“NotLoginException is occur. Please login to process.”)
+ NotLoginException(String message)	Sets `message` of superclass (`LibReserveException`) to `message`.

Method

+ String tinyMessage()	Return “You are not login.”

Package history

Class History extends GridPane implements TimeIntervalUpdate

Controller of reserve and unreserve at specific position and time.

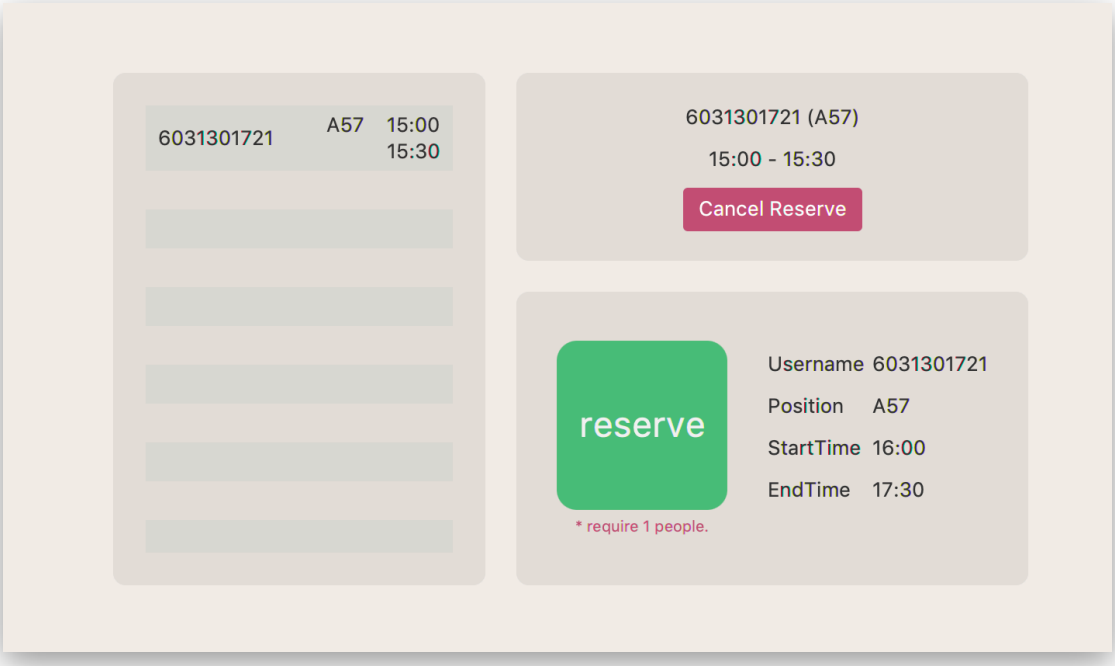


Fig 24: HistoryPane with 1 record.

Field

+ History latestHistory	Object for refer to the `latestHistory` Object
String position	Seat position (Start with 1 alphabet and end with number)
String username	Username of user that login
boolean userlog	To clarify that the method should return History of user or History on position. It true it could be History of user.
# Log currentLog	The Log of specific user or position that requested.
# Records recordsPane	The pane that contains many of Log.
# RightDetails rightDetail	The pane that contains details of current reservation and cancel reservation.

# ReservePane reservePane	The pane that contains details of doing reservation.

Constructor

+ History(String str)	This method run `History(str, false)`
+ History(String str, boolean userlog)	Initializes the `History`. This method should: - If `userlog` is true, `username` is set to `str` and `position` is set to "-" - If not, `username` is set to "-" and `position` is set to `str`. - Set the horizontal and vertical gap of 20. - Sets `this.userlog` to `userlog` - Instantiates each field: - If `userlog` is true, `recordsPane` is a `Records` of `Database.getHistory(username)`. If not, `recordsPane` is a `Records` of `Database.getPositionRecordFULL(position)` - `rightDetail` using `RightDetail` - `reservePane` using `ReservePane`, and setSeat of the `reservePane` to `position` - Add `EventHandler` to each field: - `fireEvent` from `rightDetail` to `recordsPane` when `delete` log to make `recordsPane` run `remove` method. - `fireEvent` from `recordsPane` to `rightDetail` when `focus` on `recordsPane` to make `rightDetail` run `focus` method - when the `reservePane` update log, make this pane run `initialize()` - Adds each field to the pane as a children of this pane in the correct position. - Run `initialize()` - Sets `latestHistory` refer to this. - Add `EventHandler` to make this run initialize again when login and logout.

Method

+ void initialize()	The method run `reservePane.initialize()` and `recordsPane.initialze()`
+ void intervalUpdate(long s, long t)	Refresh the `History` during input time (s is start time and t is end time)

Class HistoryLog extends HBox

Wrapper for reserving detail.

Fig 25: Example of Historylog

Field

Log log	log of the reserve store in Log.

Constructor

+ HistoryLog(**Log log**)	Initializes the `HistoryLog` using HBox. This method should: - Instantiates this pane and sets alignment of the pane to `Pos.CENTER`. - Sets `this.log` to `log`. - Instantiates each field: - `nm` is a `final Label` with a text of `username`. - `tm1` is a `final Label` with a text of `position` and `startTime`. - `tm2` is a `final Label` with a text of `endTime`. - `vb` is a VBox that contains `tm1` and `tm2` and is set alignment of the pane to `Pos.CENTER_RIGHT`. - `rg` is a `final Region` that is set horizontal grow as `Priority.ALWAYS`. - Adds each field to the pane as a children of this pane in the correct order.

Method

Log getLog()	Return `log` (Getter for `Log` Field).

Class Log

Wrapper for seat recorded data.

Field

+ SimpleDateFormat TIMEFORMAT	Constant variable for using. This value is `new SimpleDateFormat("HH:mm")`
+ SimpleDateFormat DATEFORMAT	Constant variable for using. This value is `new SimpleDateFormat("yyyy-MM-dd")`
+ SimpleDateFormat DATETIMEFORMAT	Constant variable for using. This value is `new SimpleDateFormat("yyyy-MM-dd HH:mm")`
+ String username	Username of user that login
+ String position	Seat position (Start with 1 alphabet and end with number)
+ Integer startTime	Start time of the reservation that user want.
+ Integer endTime	End time of the reservation that user want.
+ Integer reserveTime	The time that user request the reservation.

Constructor

+ Log(String username, int startTime, int endTime, String position)	Instantiates each field: - `this.username` is set to `username` - `this.startTime` is set to `startTime` - `this.endTime` is set to `endTime` - `this.position` is set to `position` - `this.reserveTime` is set to 0.
+ Log(JSONObject jo)	Instantiates each field by extact from JSONObject.

Method

+ String getNowTime()	Return current time in `TIMEFORMAT` format.

+ String getNowDate()	Return current date in DATEFORMAT format.
+ String getNowDateTime()	Return current date and time in DATETIMEFORMAT format.
+ long getNowTimeMinute()	Return current time in minutes (60 * HH + mm)
String toSimpleTime(int tm)	Return time that input as int in TIMEFORMAT format.
+ String getUser()	Return username.
+ String getZone()	Return zone of the seat position.
+ String getSeat()	Return number of seat in the zone.
+ String getStartTime()	Return startTime in TIMEFORMAT format.
+ String getEndTime()	Return endTime in TIMEFORMAT format.
+ String getReserveTime()	Return reserveTime in TIMEFORMAT format.
+ String getPosition()	Return position (getZone() + getSeat())
+ String getTitle()	Return getZone() + getSeat() + " " + getStartTime()
+ String toString()	Return "LOG: " + getUser() + " " + getPosition() + " " + getStartTime() + " " + getEndTime()
+ boolean equals(Log log)	This method return true if this.log and log can be print (use method toString()) as the same String.

Class Records extends VBox

Rendering records which allow to select item.

Fig 26: Records Pane that have 4 logs

Field

# ObservableList<HistoryLog> labelList	
- Supplier<ArrayList<JSONObject>> getRecords	
- ListView<HistoryLog> listView	
- Log currentLog	

Constructor

+ Records (Supplier<ArrayList<JSONObject>> getRecords)	Initializes the `Records` using VBox. This method should:
	- Instantiates this pane and sets alignment of the pane to `Pos.CENTER`.
	- Instantiates each field:
	  - `listView` is a `ListView<HistoryLog>` of `labelList`.
	  - Sets `this.getRecords` to `getRecords`.
	- Adds `ListView` to the pane as a children of this pane.
	- Run `initialize()`

Method

+ void initialize()	
+ void remove(Log param)	remove `Log` on `labelList` that equals to input `param`.
+ Log getCurrentLog()	Return `currentLog`
+ Log setCurrentLog(Log currentLog)	This method should: - Sets `this.currentLog` to `currentLog` if `this.currentLog` is not equal to `currentLog` - Run `initialze()`. - Run `focus(currentLog)` - return `currentLog`
+ void focus(Log log)	Focus on `Log` on `labelList` that equals to input `log`.

Class ReservePane extends HBox implements TimeIntervalUpdate

Reserving button showing

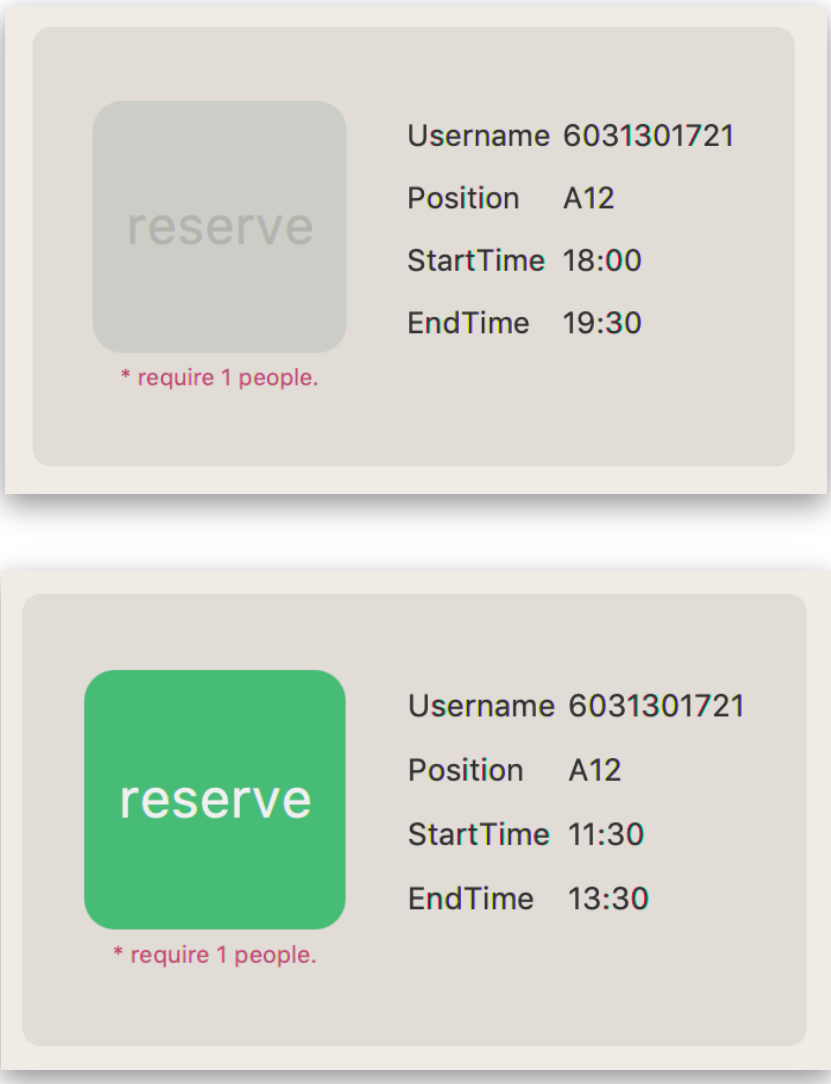


Fig 27: ReservePane when can reserve and cannot reserve

Field

Button submitBtn	
int s	Memoi startTime
int t	Memoi endTime
String seat	Memoi Seat position

Constructor

ReservePane()	Redirect to `ReservePane("-", 0, 0)`
ResrevePane(String seat, int s, int t)	Copy arguments to class field, set style and call initialize()

Method

- Log getLog()	Return Log from class field data and `Store.getUsername()` but set username to "-" if not login yet
+ void initialize()	Clear child. Set helper text if not login: "* please login" else if seat == "-": "* history mode" else: "* require {Table.getRequirenumber (seat)} people" create `submitBtn = "reserve"` handle click by if not Database.isHistoryConflict(*log) then popup GroupLoginInput (save to currentGroupLoginInput) implement unimplement method @onsucess If Database.add(*log) success then fireEvent(UPDATE_LOG) and this.close() for close pop up Else set submitBtn to disable @onclose set submitBtn to disable set submitBtn to enable if isLogin() & Table.isValidSeat & log.startTime != log.endTime else set submitBtn to disable set style and display all component
+ void intervalUpdate(long s, long t)	Update s and t then initialize()

+ GridPane getTable()	Return GridPane display info from `getLog()`
+ void setSeat(String position)	Set seat field = position

Class RightDetail extends VBox

Fig 28: RightDetail pane when nothing selected, cannot cancel reserve and can cancel reserve

Field

- Button cancelReserve	Holding Cancel Reserve Button
- Log log	Field data for display

Constructor

RightDetail()	Set style and calling `set()`
RightDetail(Log log)	Set style and calling `set(log)`

Method

+ void set()	Clear all field & children cimponent. Disply `Label("No Data to Display")`
+ void set(Log log)	Showing data from `log.` Create `Button("CancelReserve")` and save to `cancelReserve` if `allowcancel()` then `cancelReserve have been press you must remove log from Database (Database.remove(*log))` , `fireEvent(DELETE_LOG, log)` and clear screen by `set()` else set `cancelReserve to Disable`
- boolean allowcancel()	Return true is Store.isLogin() and log.getUser() equal to Store.getUsername()
+ void focus(Log param)	Call set(param)

Package utility

Class STD

Helper class for easier implementation.

Method

+ <T> T back(T[] ar)	Return the last data of input Array.
+ int count(String str, String tok)	Count specific word (tok) in the sentence (str).