BACH KHOA HA NOI UNIVERSITY

Object-oriented programming Project

Topic number 15: “Quản lý cán bộ một trường học”

Professor

Đỗ Thị Ngọc Diệp

Member List:

Nguyễn Minh Đức – 20207664

Lê Minh Nghĩa – 20207694

Phạm Đức Phúc – 20207698

No	Name	Student ID	Work assignment	Completion	Note
01	Nguyễn Minh Đức (Leader)	20207664	-Planning - Incharge Interface design -Code class mainscene, App and class staff -Report writing	100%	
02	Lê Minh Nghĩa	20207694	-2nd designer - Write Debugging code -Research to get Info -Code class Secondscene And everything relate to calculation - Making UML diagram	100%	The most enthusiastic person in the group
03	Phạm Đức Phúc	20207698	-Research and code for save and load function - Implement alert box -Write comment - Complete the UML diagram	100%	The most inquisitive person in the team

Prologue

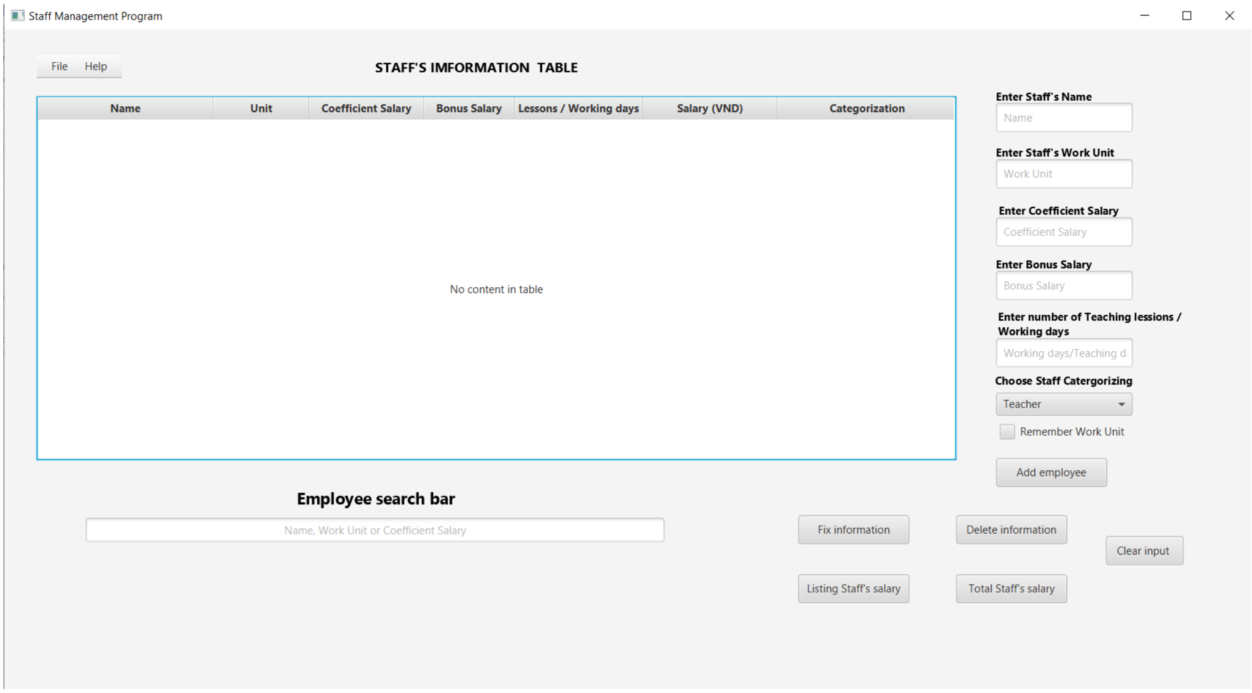
Before i begin, i would like to say hello and greeting to anyone who read this report. This is our first and biggest project we ever have in university. Most of us is unknown to java, or rather beginner. Our topic is 15, why is 15? I think the most proper answer for this question is that we think 15 is a lucky number, that’s all. To be honest, this report won’t contain all the information, there are no such thing will explain why we do this or why we do that, so i just want to inform you that before before we dive into this report. Long anwser short, lets get started!

I.In details

We started project on November 4th. On the first week, Nguyen Minh Duc, the leader in our team planned which library we would use and the rough estimated design for what we would do in the later stages. He decided that every Friday we would have a meeting call at 8 pm to report and express what each members wanted to change little by little and what they had to do.

In the process we chose JavaFx which would be our library for the GUI coding instead of Swing. The reason was that we found out JavaFx had Scene Builder, a program that make designing GUI easier for programmers.

By the start of the second week, we took what we learn into coding. Starting with class App. Class App is basically where we launch the code. After having lauched successfully several times, we moved to class MainScene . This was when Scene Builder came in handy, the Scene Builder helped us design the Scene (Tableview.fxml) where users will see it when they launch the Program.



Name	Unit	Coefficient Salary	Bonus Salary	Lessons / Working days	Salary (VND)	Categorization

Along with programming MainScene, we also programmed class Staff, which is demonstrated in the table as you can see. For example Chau is a Staff.

The MainScene could be said to be the most difficult part for us because we had to change it many times for almost every week but we will talk about it later in **Problems** section.

After 2 weeks of fixing and coding class MainScene, Nghia was in charge of coding class SecondScene. He would write the code first, then Duc would be the 1st tester, then Nghia would act as the 2nd tester. The process was repeated until it was finished.

In the late December, Phuc would be the one coding the last function of the Program (save and load).

In 20/12 we officially finished the project.

II.How to install new library

- Install JavaFx, SceneBuilder and their respective libraries.

+) Download link for Javafx library: https://openjfx.io

+) Download link for Scene Builder: Scene Builder - Gluon (gluonhq.com)

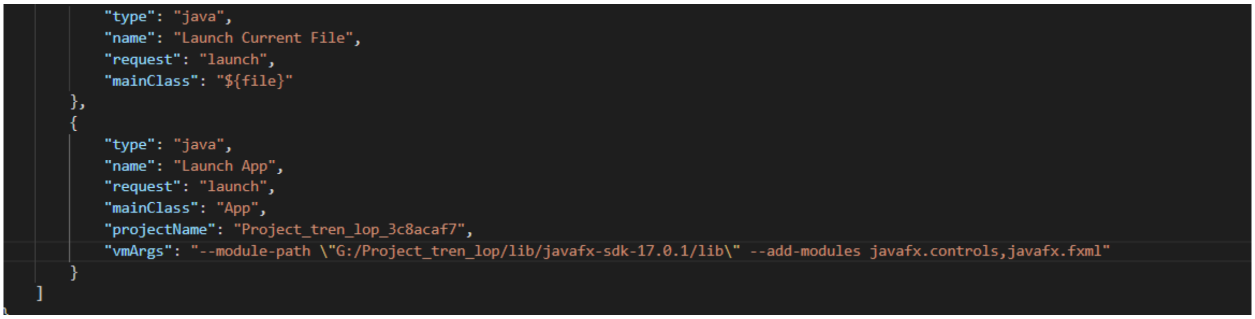
+) Instruction Video: https://www.youtube.com/watch?v=Ope4icw6bVk

+) Source code for JavaFX installation:

= > Linux/Mac: --module-path /path/to/javafx-sdk-15.0.1/lib --add-modules javafx.controls,javafx.fxml

=> Windows: --module-path "\path\to\javafx-sdk-15.0.1\lib" --add-modules javafx.controls,javafx.fxml

Example



```
            "type": "java",
            "name": "Launch Current File",
            "request": "launch",
            "mainClass": "${file}"
        },
        {
            "type": "java",
            "name": "Launch App",
            "request": "launch",
            "mainClass": "App",
            "projectName": "Project_tren_lop_3c8acaf7",
            "vmArgs": "--module-path \"G:/Project_tren_lop/lib/javafx-sdk-17.0.1/lib\" --add-modules javafx.controls,javafx.fxml"
        }
    ]
```

+) Javafx coding guide (English version): https://www.youtube.com/playlist?list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG

+) Javafx coding guide (Vietnamese version): : https://www.youtube.com/playlist?list=PL33lvabfss1yRgFCgFXjtYaGAuDJjjH-j

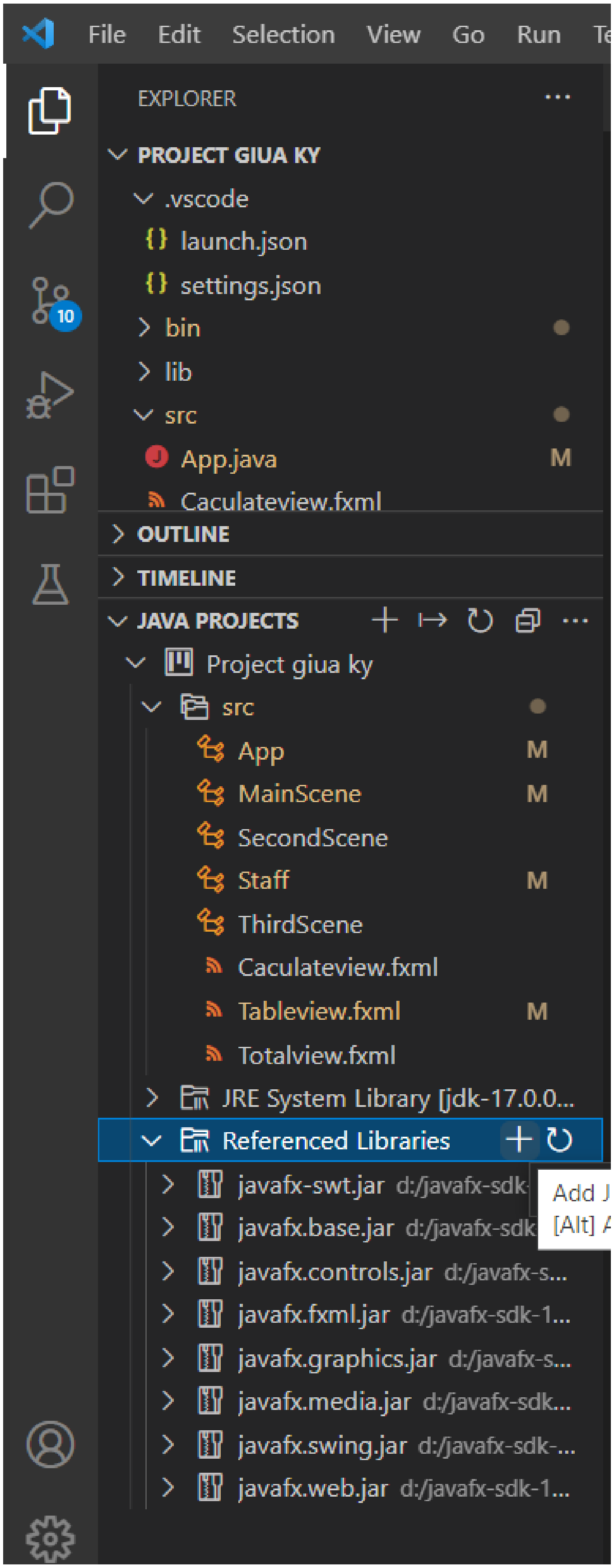
+)JavaFX ObservableList: https://www.youtube.com/watch?v=XvnJAVItaAw

For us, we used Vscode:

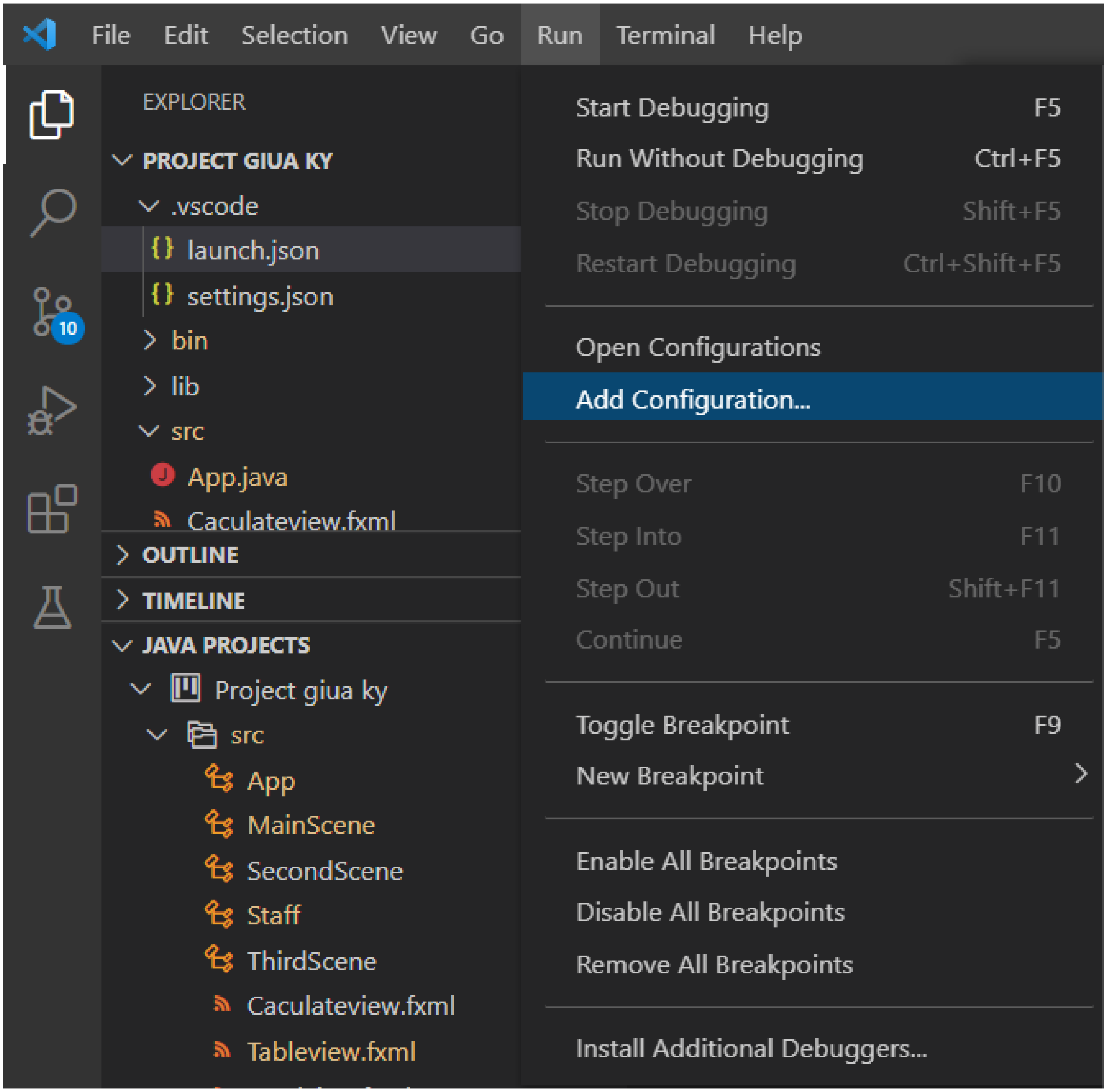
Step 1 : Download Javafx library and extract it in a new folder.

Step 2: Create a Project Java in Vscode

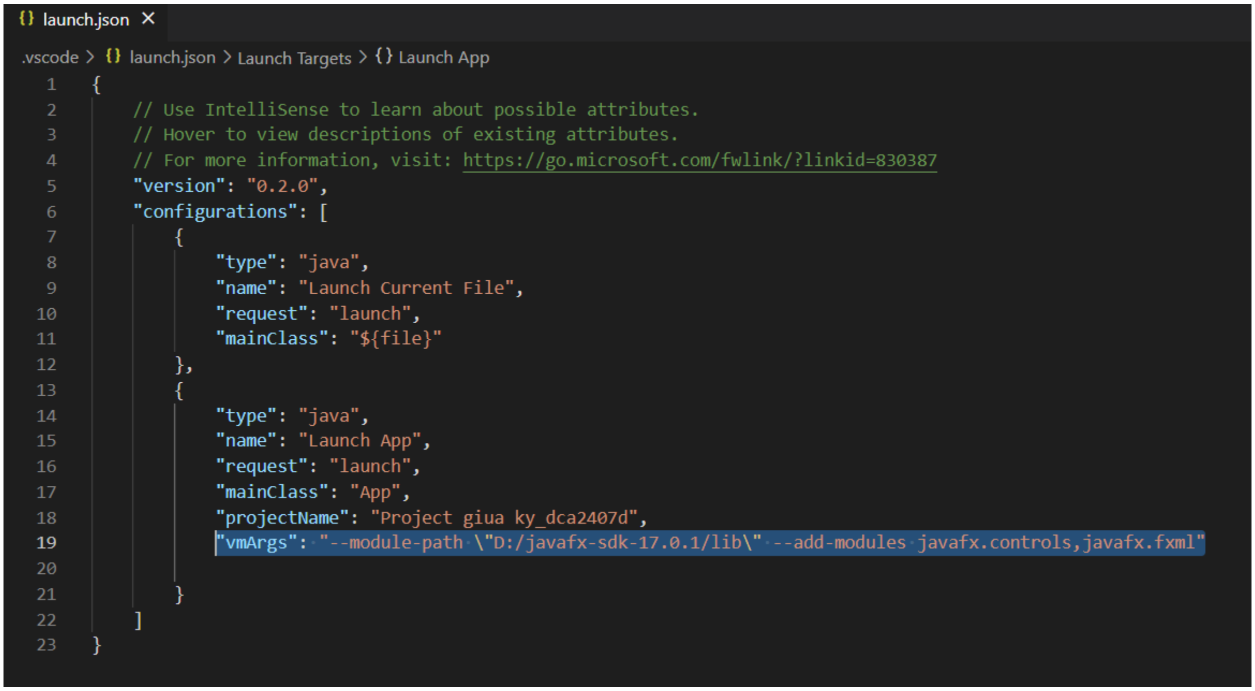
Step 3: Add a new library (all the extracted file) by clicking the (+) symbol in Referenced Libararies. In our case we have put the library in the folder lib in our project folder.


File
Edit
Selection
View
Go
Run
Te
EXPLORER
PROJECT GIUA KY
.vscode
launch.json
settings.json
bin
lib
src
App.java
M
Caculateview.fxml
OUTLINE
TIMELINE
JAVA PROJECTS
Project giua ky
src
App
M
MainScene
M
SecondScene
Staff
M
ThirdScene
Caculateview.fxml
Tableview.fxml
M
Totalview.fxml
JRE System Library [jdk-17.0.0...
Referenced Libraries
javafx-swt.jar d:/javafx-sdk-
Add J
[Alt] A
javafx.base.jar d:/javafx-sdk
javafx.controls.jar d:/javafx-s...
javafx.fxml.jar d:/javafx-sdk-1...
javafx.graphics.jar d:/javafx-s...
javafx.media.jar d:/javafx-sdk...
javafx.swing.jar d:/javafx-sdk-...
javafx.web.jar d:/javafx-sdk-1...
10

Step 4: Create launch.json by choosing Run in Menu. After that, select Add configuration, a launch.json will appear.

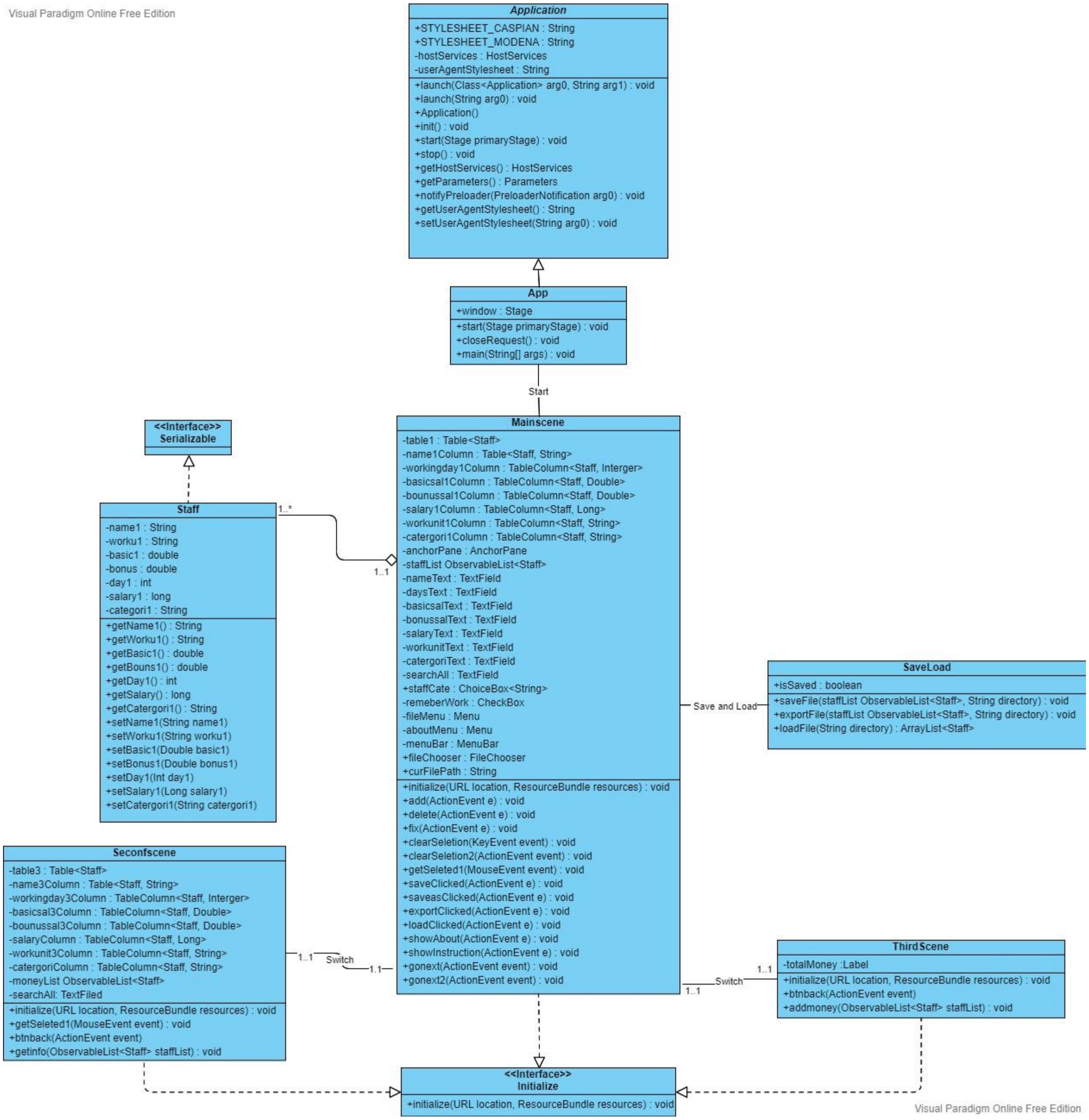


Step 5: Adding source code in launch.json and JavaFX library is ready to use.



```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "java",
            "name": "Launch Current File",
            "request": "launch",
            "mainClass": "${file}"
        },
        {
            "type": "java",
            "name": "Launch App",
            "request": "launch",
            "mainClass": "App",
            "projectName": "Project giua ky_dca2407d",
            "vmArgs": "--module-path \"D:/javafx-sdk-17.0.1/lib\" --add-modules javafx.controls,javafx.fxml"

        }
    ]
}
```

III.UML diagram


Visual Paradigm Online Free Edition
Application
+STYLESHEET_CASPIAN : String
+STYLESHEET_MODENA : String
-hostServices : HostServices
-userAgentStylesheet : String
+launch(Class<Application> arg0, String arg1) : void
+launch(String arg0) : void
+Application()
+init() : void
+start(Stage primaryStage) : void
+stop() : void
+getHostServices() : HostServices
+getParameters() : Parameters
+notifyPreloader(PreloaderNotification arg0) : void
+getUserAgentStylesheet() : String
+setUserAgentStylesheet(String arg0) : void
App
+window : Stage
+start(Stage primaryStage) : void
+closeRequest() : void
+main(String[] args) : void
Start
<<Interface>>
Serializable
Mainscene
-table1 : Table<Staff>
-name1Column : Table<Staff, String>
-workingday1Column : TableColumn<Staff, Interger>
-basicsal1Column : TableColumn<Staff, Double>
-bounussal1Column : TableColumn<Staff, Double>
-salary1Column : TableColumn<Staff, Long>
-workunit1Column : TableColumn<Staff, String>
-catergori1Column : TableColumn<Staff, String>
-anchorPane : AnchorPane
-staffList ObservableList<Staff>
-nameText : TextField
-daysText : TextField
-basicsalText : TextField
-bonussalText : TextField
-salaryText : TextField
-workunitText : TextField
-catergoriText : TextField
-searchAll : TextField
+staffCate : ChoiceBox<String>
-remeberWork : CheckBox
-fileMenu : Menu
-aboutMenu : Menu
-menuBar : MenuBar
+fileChooser : FileChooser
+curFilePath : String
+initialize(URL location, ResourceBundle resources) : void
+add(ActionEvent e) : void
+delete(ActionEvent e) : void
+fix(ActionEvent e) : void
+clearSeletion(KeyEvent event) : void
+clearSeletion2(ActionEvent event) : void
+getSeleted1(MouseEvent event) : void
+saveClicked(ActionEvent e) : void
+saveasClicked(ActionEvent e) : void
+exportClicked(ActionEvent e) : void
+loadClicked(ActionEvent e) : void
+showAbout(ActionEvent e) : void
+showInstruction(ActionEvent e) : void
+gonext(ActionEvent event) : void
+gonext2(ActionEvent event) : void
Staff
-name1 : String
-worku1 : String
-basic1 : double
-bonus : double
-day1 : int
-salary1 : long
-categori1 : String
+getName1() : String
+getWorku1() : String
+getBasic1() : double
+getBouns1() : double
+getDay1() : int
+getSalary() : long
+getCatergori1() : String
+setName1(String name1)
+setWorku1(String worku1)
+setBasic1(Double basic1)
+setBonus1(Double bonus1)
+setDay1(Int day1)
+setSalary1(Long salary1)
+setCatergori1(String catergori1)
1..*
1..1
Save and Load
SaveLoad
+isSaved : boolean
+saveFile(staffList ObservableList<Staff>, String directory) : void
+exportFile(staffList ObservableList<Staff>, String directory) : void
+loadFile(String directory) : ArrayList<Staff>
Seconfscene
-table3 : Table<Staff>
-name3Column : Table<Staff, String>
-workingday3Column : TableColumn<Staff, Interger>
-basicsal3Column : TableColumn<Staff, Double>
-bounussal3Column : TableColumn<Staff, Double>
-salaryColumn : TableColumn<Staff, Long>
-workunit3Column : TableColumn<Staff, String>
-catergoriColumn : TableColumn<Staff, String>
-moneyList ObservableList<Staff>
-searchAll: TextFiled
+initialize(URL location, ResourceBundle resources) : void
+getSeleted1(MouseEvent event) : void
+btnback(ActionEvent event)
+getinfo(ObservableList<Staff> staffList) : void
1..1
Switch
1..1
ThirdScene
-totalMoney :Label
+initialize(URL location, ResourceBundle resources) : void
+btnback(ActionEvent event)
+addmoney(ObservableList<Staff> staffList) : void
1..1
Switch
1..1
<<Interface>>
Initialize
+initialize(URL location, ResourceBundle resources) : void
Visual Paradigm Online Free Edition

We have class App is an extends of class Application.

Class App connect with class Mainscene through method Start.

- Through the window.setScene(sence) and sence will set as root which getSource from Tableview.fxml

Class Staff implements interface Serializable. We have to implements this interface because this help SaveLoad class to identify object Staff.

The relationship of Staff class and Mainscene can be understood that one Mainscene is made up of many Staff objects.

SaveLoad will apply through method Savefile, Exportfile and method Loadfile.

Class Mainscene connect with class Secondscene through method gonext.

This also apply for class Thirdscene through method gonext2

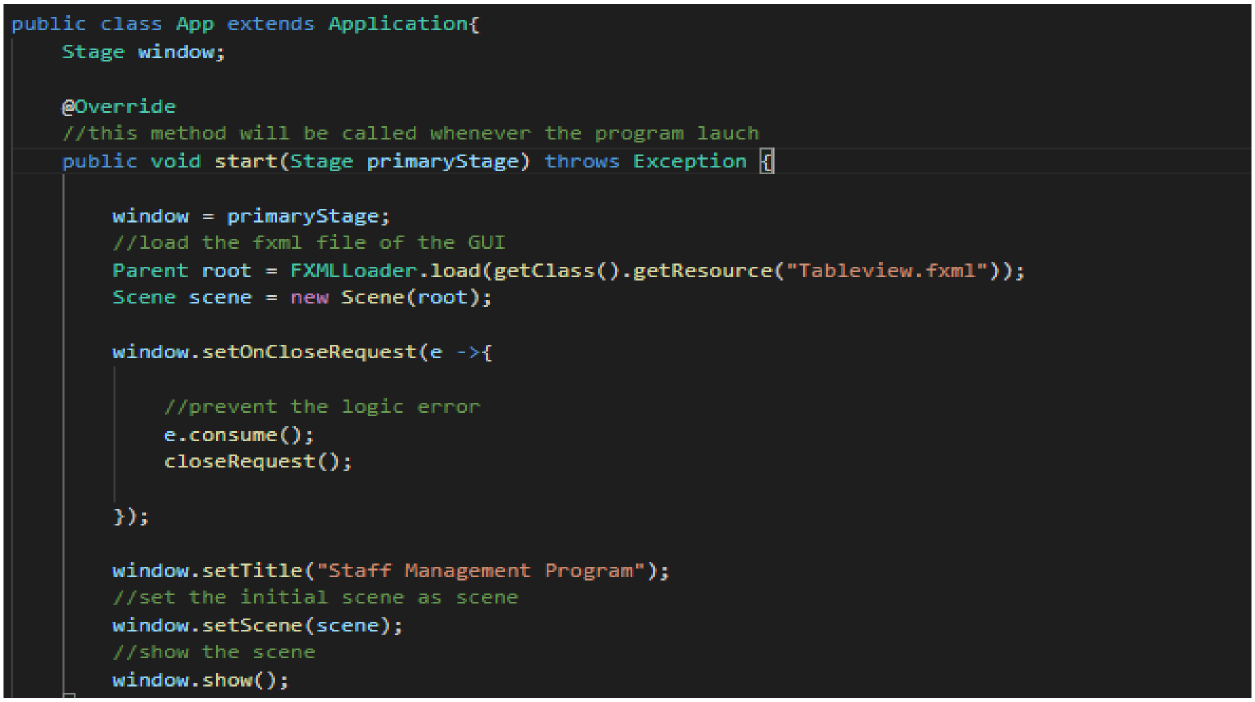
- Or precisely the Secondscene and the Thirdscene is a controller to get info (staffList) from Mainscene to those 2 class.

All Mainscene, SecondSence and ThirdSence implement interface Initialize.

IV.Analysis

Before going to the analysis section, I would like to inform you what is GUI. According to Wikipedia, “GUI or **Graphical user interface** is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs), which require commands to be typed on a computer keyboard”.

Class App



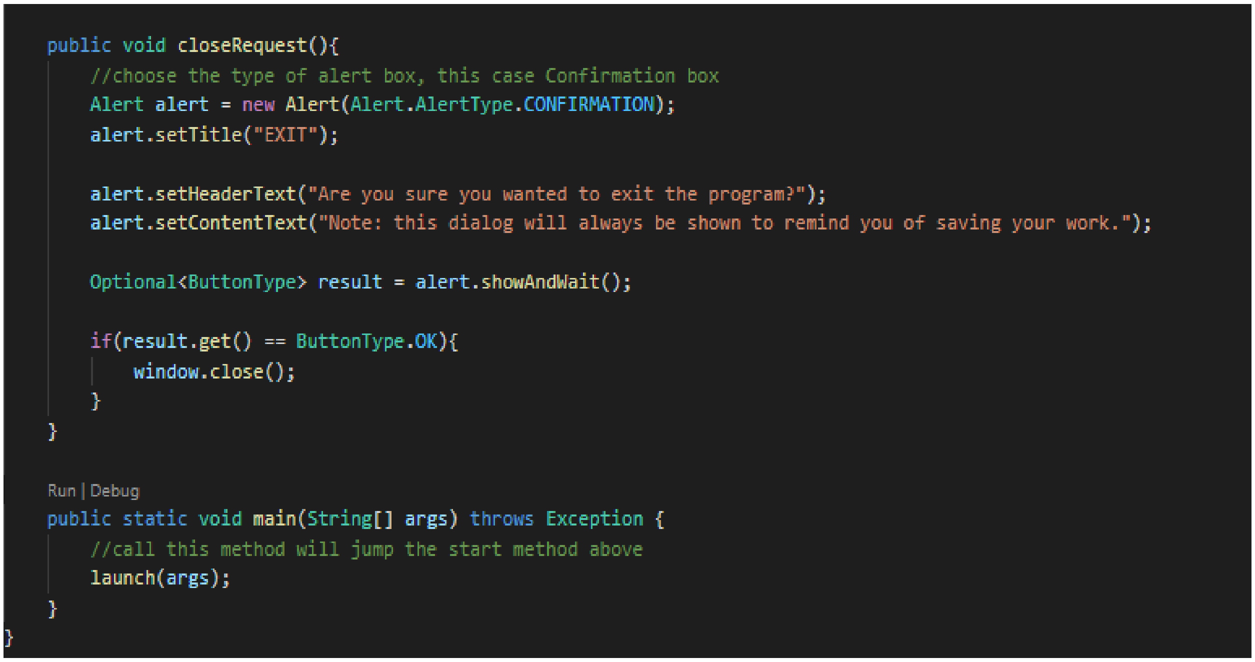
```java
public class App extends Application{
    Stage window;

    @Override
    //this method will be called whenever the program lauch
    public void start(Stage primaryStage) throws Exception {

        window = primaryStage;
        //load the fxml file of the GUI
        Parent root = FXMLLoader.load(getClass().getResource("Tableview.fxml"));
        Scene scene = new Scene(root);

        window.setOnCloseRequest(e ->{

            //prevent the logic error
            e.consume();
            closeRequest();

        });

        window.setTitle("Staff Management Program");
        //set the initial scene as scene
        window.setScene(scene);
        //show the scene
        window.show();
```

Class App can be seen as a main class for the program, it contains the method that launch the app and the method that controls the stage as we see in the program.

The first attribute is Stage window, this will allow us to work with the Stage of our program outside the start() method’s scope.

The process of running the program can be say like this: the static void main method run first and inside the “launch(args)” method launch the program, then the program proceed to run the start() method:

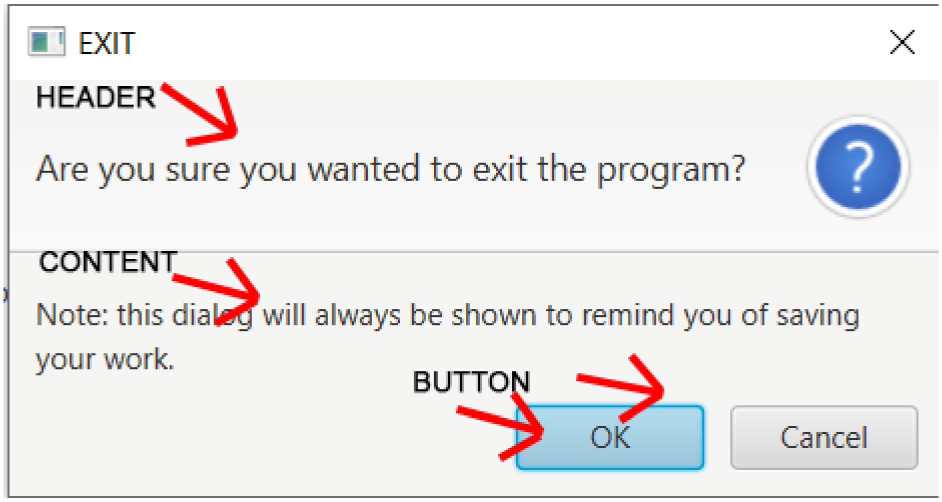
- When the start method run, it reference the window to the primaryStage which is the Stage that the start() method used
- The next two lines load the directory of the .fxml file (using the getResource() method) which then loads into the Parent object root. Then a Scene object is created, using the root (Parent root). Thank to this, the scene will display the layout from the .fxml file which is TableView.fxml.



```
    public void closeRequest(){
        //choose the type of alert box, this case Confirmation box
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("EXIT");

        alert.setHeaderText("Are you sure you wanted to exit the program?");
        alert.setContentText("Note: this dialog will always be shown to remind you of saving your work.");

        Optional<ButtonType> result = alert.showAndWait();

        if(result.get() == ButtonType.OK){
            window.close();
        }
    }

    public static void main(String[] args) throws Exception {
        //call this method will jump the start method above
        launch(args);
    }
}
```

About the closeRequest() method, as mentioned, the closeRequest method will be activated when the user hits the close button on the window of the stage. Inside the closeRequest() method:

- Firstly an AlertBox object is created with a reference variable named “alert”, with the alert type is CONFIRMATION. As the name suggests, the Alert class is an alert box type, which will pop-out whenever you make changes or trigger the Event method that contains them and you can only dismiss them by closing them. The type is Confirmation, which will have both the header text and the content text, aside from that, the Confirmation alert box will have two buttons which are “Ok” and “Cancel”.
- Next, the header text and content text are set with String text (look at above).
- Next, An Optional object is created which will receive the return value of the choice we made in the alert window thanks to the showAndWait() method on its other end. We can get the value of the user choice by this way.



EXIT

Are you sure you wanted to exit the program?

Note: this dialog will always be shown to remind you of saving your work.

- Finally, if the user chooses the “OK” button the program will close (because the value result get has the same value as the static final ButtonType.OK. If the user chooses “Cancel” or clicks the close button, the program will be back on the Mainscene. Both of the scenario, the alert box always closes.

Class Staff

Atribute

- Class staff has : +) 1 double variable name basic1

+) 3 string varibale name name1 , worku1 and categori1

+) int variable name day1

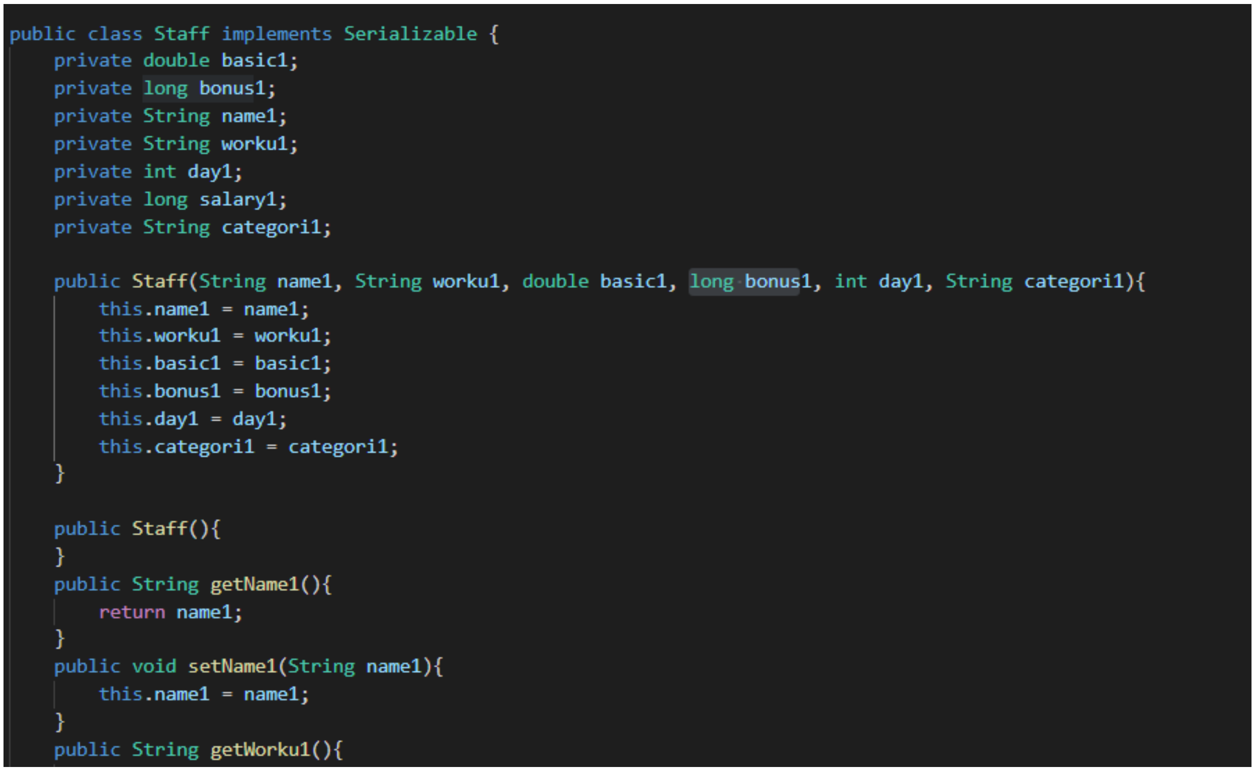
+) long variable name bonus1 and salary1

Method

- Contruct public Staff(String name1, String worku1, double basic1, long bonus1, int day1, String categori1)

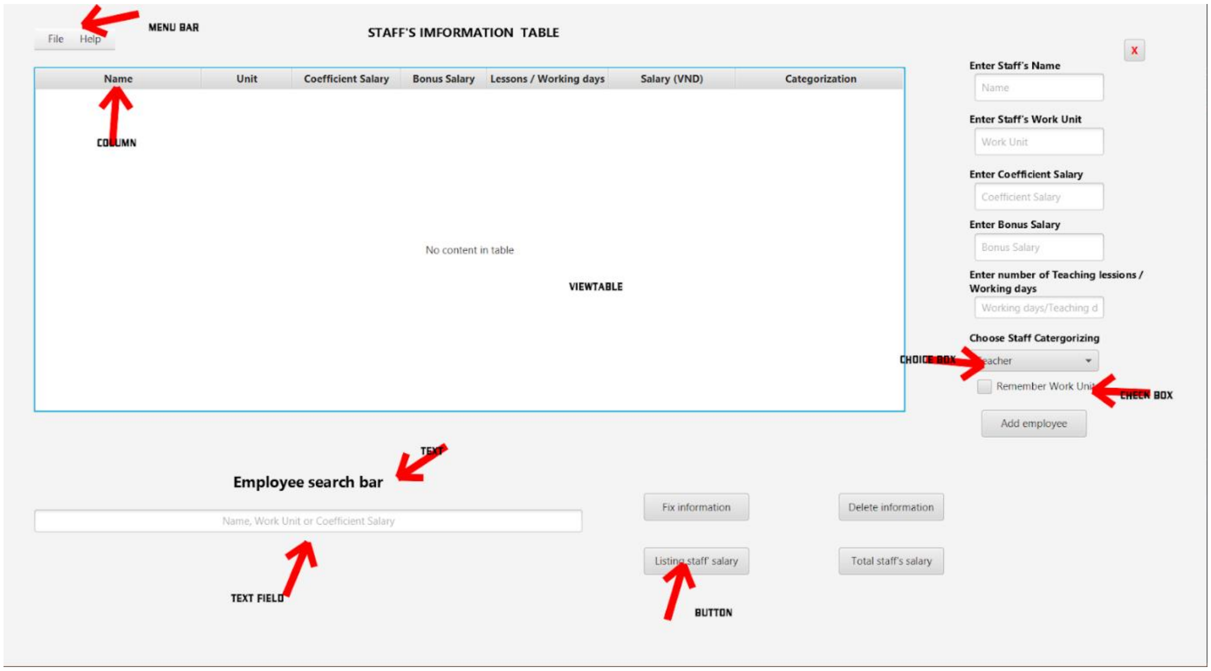
- get(atribute name) () method : return the atribute

- set (variable name) (type value) method : set the value of atribute as the value type in

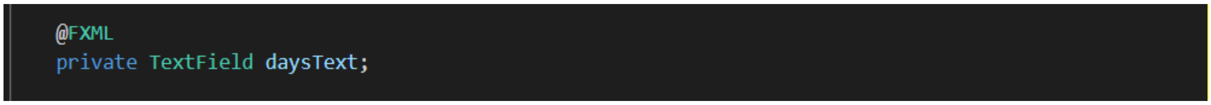


```
public class Staff implements Serializable {
    private double basic1;
    private long bonus1;
    private String name1;
    private String worku1;
    private int day1;
    private long salary1;
    private String categori1;

    public Staff(String name1, String worku1, double basic1, long bonus1, int day1, String categori1){
        this.name1 = name1;
        this.worku1 = worku1;
        this.basic1 = basic1;
        this.bonus1 = bonus1;
        this.day1 = day1;
        this.categori1 = categori1;
    }

    public Staff(){
    }
    public String getName1(){
        return name1;
    }
    public void setName1(String name1){
        this.name1 = name1;
    }
    public String getWorku1(){
```

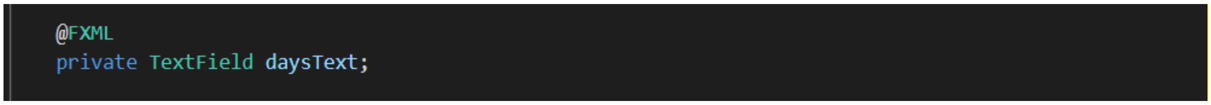
.FXML files



- Every scene you see is the result of an fxml files. In short, we only explain the tableview.fxml because it has the same construction for all fxml files. We have noted them as you can see in the picture with their corresponding functions. Each scene (beside Button) was registered with an ID in each class and applied to their respective fxml file.

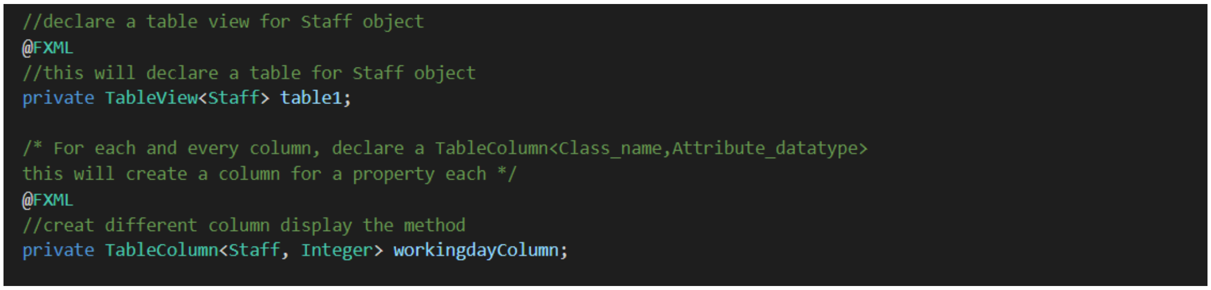


```
@FXML
private TextField daysText;
```



```
@FXML
private TextField daysText;
```

- ID is an identifier which a programmer labels each part with a unique name to differentiate and making it easier during the coding process.

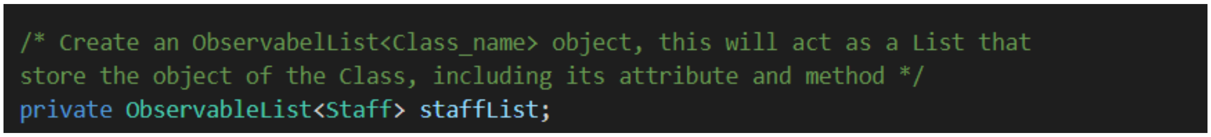


```
//declare a table view for Staff object
@FXML
//this will declare a table for Staff object
private TableView<Staff> table1;

/* For each and every column, declare a TableColumn<Class_name,Attribute_datatype>
this will create a column for a property each */
@FXML
//creat different column display the method
private TableColumn<Staff, Integer> workingdayColumn;
```

- Tableview Declaration will display every value related to Class Staff

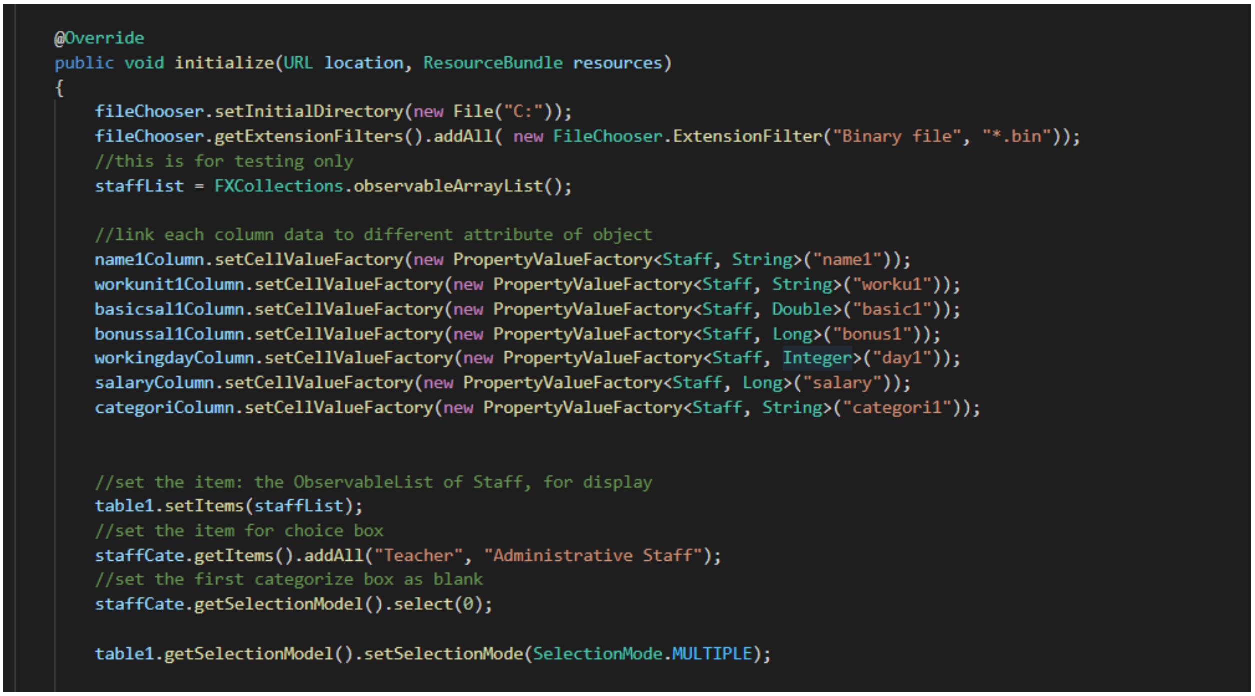
- Tableview Column Declaration will display Class Staff with their specific value.

Class MainSence



```
/* Create an ObservabelList<Class_name> object, this will act as a List that
store the object of the Class, including its attribute and method */
private ObservableList<Staff> staffList;
```

- ObservableList is a list that Tableview always display.



```
@Override
public void initialize(URL location, ResourceBundle resources)
{
    fileChooser.setInitialDirectory(new File("C:"));
    fileChooser.getExtensionFilters().addAll( new FileChooser.ExtensionFilter("Binary file", "*.bin"));
    //this is for testing only
    staffList = FXCollections.observableArrayList();

    //link each column data to different attribute of object
    name1Column.setCellValueFactory(new PropertyValueFactory<Staff, String>("name1"));
    workunit1Column.setCellValueFactory(new PropertyValueFactory<Staff, String>("worku1"));
    basicsal1Column.setCellValueFactory(new PropertyValueFactory<Staff, Double>("basic1"));
    bonussal1Column.setCellValueFactory(new PropertyValueFactory<Staff, Long>("bonus1"));
    workingdayColumn.setCellValueFactory(new PropertyValueFactory<Staff, Integer>("day1"));
    salaryColumn.setCellValueFactory(new PropertyValueFactory<Staff, Long>("salary"));
    categoriColumn.setCellValueFactory(new PropertyValueFactory<Staff, String>("categori1"));


    //set the item: the ObservableList of Staff, for display
    table1.setItems(staffList);
    //set the item for choice box
    staffCate.getItems().addAll("Teacher", "Administrative Staff");
    //set the first categorize box as blank
    staffCate.getSelectionModel().select(0);

    table1.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
```

staffList is called an Array list - **A List that notifies when changes are made.** The FXcollections have to call to identified the observableArrayList.

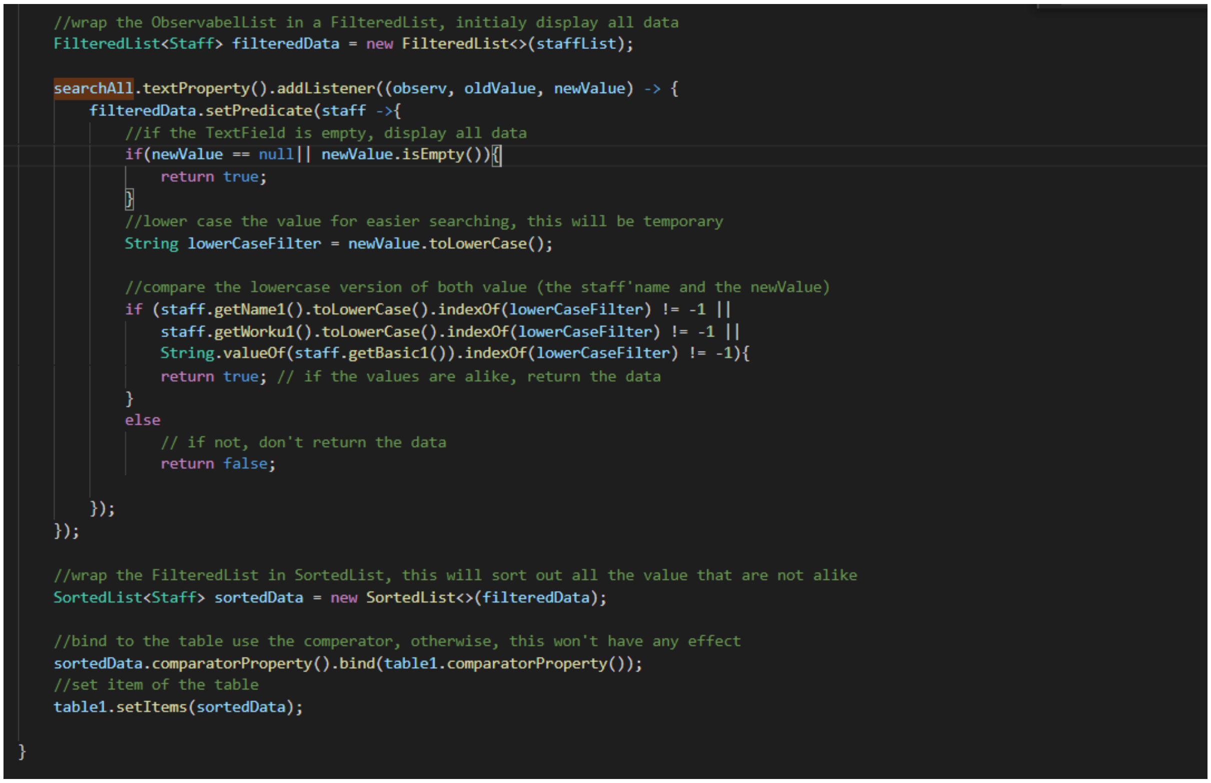
- Declare for each column with their corresponding value in Class Staff.

- Table.setItems(staffList) is where the staffList will be displayed

- Add 2 value: “Teacher” and “administrative” for Choice box to display on GUI.

- table1.getSelectionModel().setSelectionMode(selectionMode.Multiple) improves selection mode from only choosing 1 value as default to choosing multiple value in table1.

Search Method



```
    //wrap the ObservabelList in a FilteredList, initialy display all data
    FilteredList<Staff> filteredData = new FilteredList<>(staffList);

    searchAll.textProperty().addListener((observ, oldValue, newValue) -> {
        filteredData.setPredicate(staff ->{
            //if the TextField is empty, display all data
            if(newValue == null|| newValue.isEmpty()){
                return true;
            }
            //lower case the value for easier searching, this will be temporary
            String lowerCaseFilter = newValue.toLowerCase();

            //compare the lowercase version of both value (the staff'name and the newValue)
            if (staff.getName1().toLowerCase().indexOf(lowerCaseFilter) != -1 ||
                staff.getWorku1().toLowerCase().indexOf(lowerCaseFilter) != -1 ||
                String.valueOf(staff.getBasic1()).indexOf(lowerCaseFilter) != -1){
                return true; // if the values are alike, return the data
            }
            else
                // if not, don't return the data
                return false;

        });
    });

    //wrap the FilteredList in SortedList, this will sort out all the value that are not alike
    SortedList<Staff> sortedData = new SortedList<>(filteredData);

    //bind to the table use the comperator, otherwise, this won't have any effect
    sortedData.comparatorProperty().bind(table1.comparatorProperty());
    //set item of the table
    table1.setItems(sortedData);

}
```

- Create 1 filterList which include the Staff attribute with the information from staffList

- From textField, searchAll, we can extract features of all the information stored tin them with 3 value functions Observ, oldValue và newValue

- If the value we imported in searchAll, or newValue, is empty then the return values will not change, meaning staffList will stay the same.

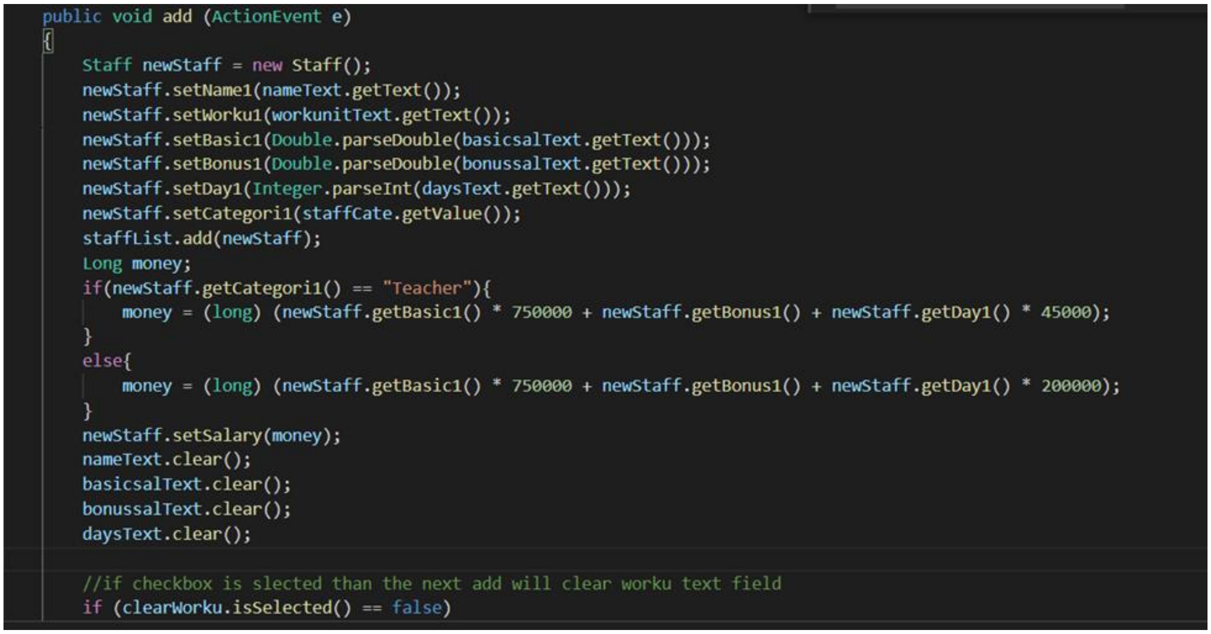
- newValue standards apply to toLowerCase is lowerCaseFilter.

This Java string method converts every character of the particular string into the lower case by using the rules of the default locale.

- Similary, If the value imported in is identical to the value extracted from staff which are getName1, getWorku1, and getBasic1 . It means the staffList Array (staffList !=-1 mean this staffList Array is not an empty set) have the return values true in respond to all the conditions mentioned above.

- else returning values is false which will not be displayed on staffList.

Add method



```
public void add (ActionEvent e)
{
    Staff newStaff = new Staff();
    newStaff.setName1(nameText.getText());
    newStaff.setWorku1(workunitText.getText());
    newStaff.setBasic1(Double.parseDouble(basicsalText.getText()));
    newStaff.setBonus1(Double.parseDouble(bonussalText.getText()));
    newStaff.setDay1(Integer.parseInt(daysText.getText()));
    newStaff.setCategori1(staffCate.getValue());
    staffList.add(newStaff);
    Long money;
    if(newStaff.getCategori1() == "Teacher"){
        money = (long) (newStaff.getBasic1() * 750000 + newStaff.getBonus1() + newStaff.getDay1() * 45000);
    }
    else{
        money = (long) (newStaff.getBasic1() * 750000 + newStaff.getBonus1() + newStaff.getDay1() * 200000);
    }
    newStaff.setSalary(money);
    nameText.clear();
    basicsalText.clear();
    bonussalText.clear();
    daysText.clear();

    //if checkbox is slected than the next add will clear worku text field
    if (clearWorku.isSelected() == false)
```

- Declare newStaff variable as Staff

- Set name1 attribute of newStaff with the value received from textField nameText.

- Set workunit1 attribute of newStaff with the value received from textField workunitText.

- Set basic1 attribute of newStaff with the value received from textField basiccalText.

- Set bonus1 attribute of newStaff with the value received from textField bonussalText.

- Set day1 attribute of newStaff with the value received from textField dayText.

- Set staffcatergori attribute of newStaff with the value received from choiceBox staffCate.

- Declare money variable by type long.

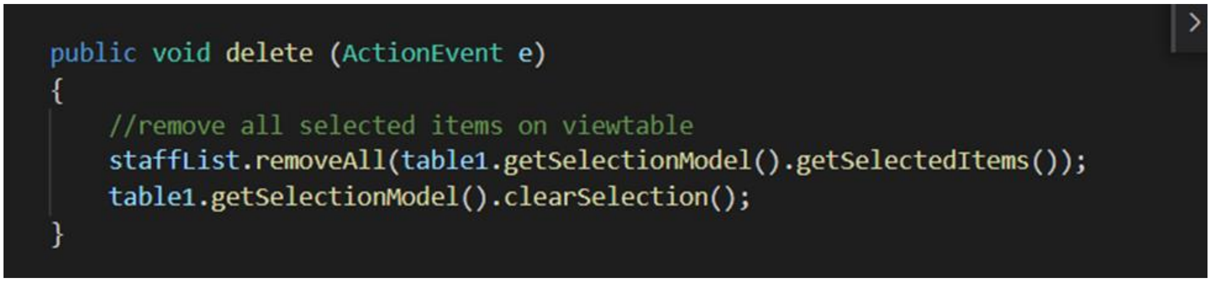
- If the newstaff’s staffcatergori is “Teacher” , put money with cast data type long of basic1 of person * 750000 + bonus1 + day1*45000 . If not then put money with cast data type long of basic1 of person * 750000 + bonus1 + day1*200000.

- Set salary attribute of newStaff with money.

- The next 5 commands are clear values of textFiled.

For the Add method, we use actionEvent. ActionEvent is a code which we apply for a button. This is an event-like action which a person clicks on them to activate.

Delete method

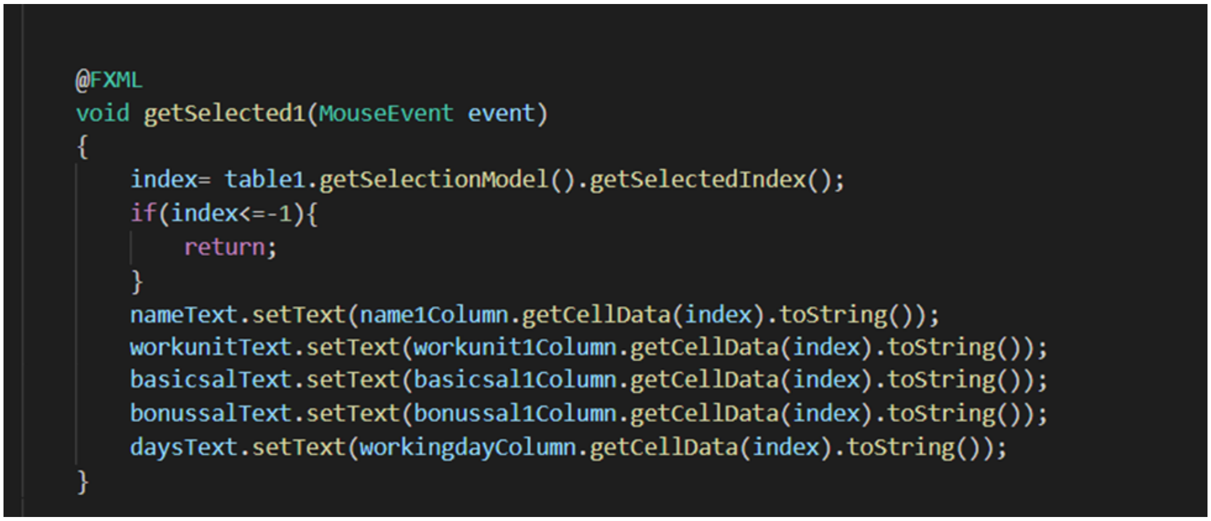


```
public void delete (ActionEvent e)
{
    //remove all selected items on viewtable
    staffList.removeAll(table1.getSelectionModel().getSelectedItems());
    table1.getSelectionModel().clearSelection();
}
```

- Delete button will remove every items selected in staffList of table1.

- Then selection on table 1 will be cleared.

getSlected1 method



```
@FXML
void getSelected1(MouseEvent event)
{
    index= table1.getSelectionModel().getSelectedIndex();
    if(index<=-1){
        return;
    }
    nameText.setText(name1Column.getCellData(index).toString());
    workunitText.setText(workunit1Column.getCellData(index).toString());
    basicsalText.setText(basicsal1Column.getCellData(index).toString());
    bonussalText.setText(bonussal1Column.getCellData(index).toString());
    daysText.setText(workingdayColumn.getCellData(index).toString());
}
```

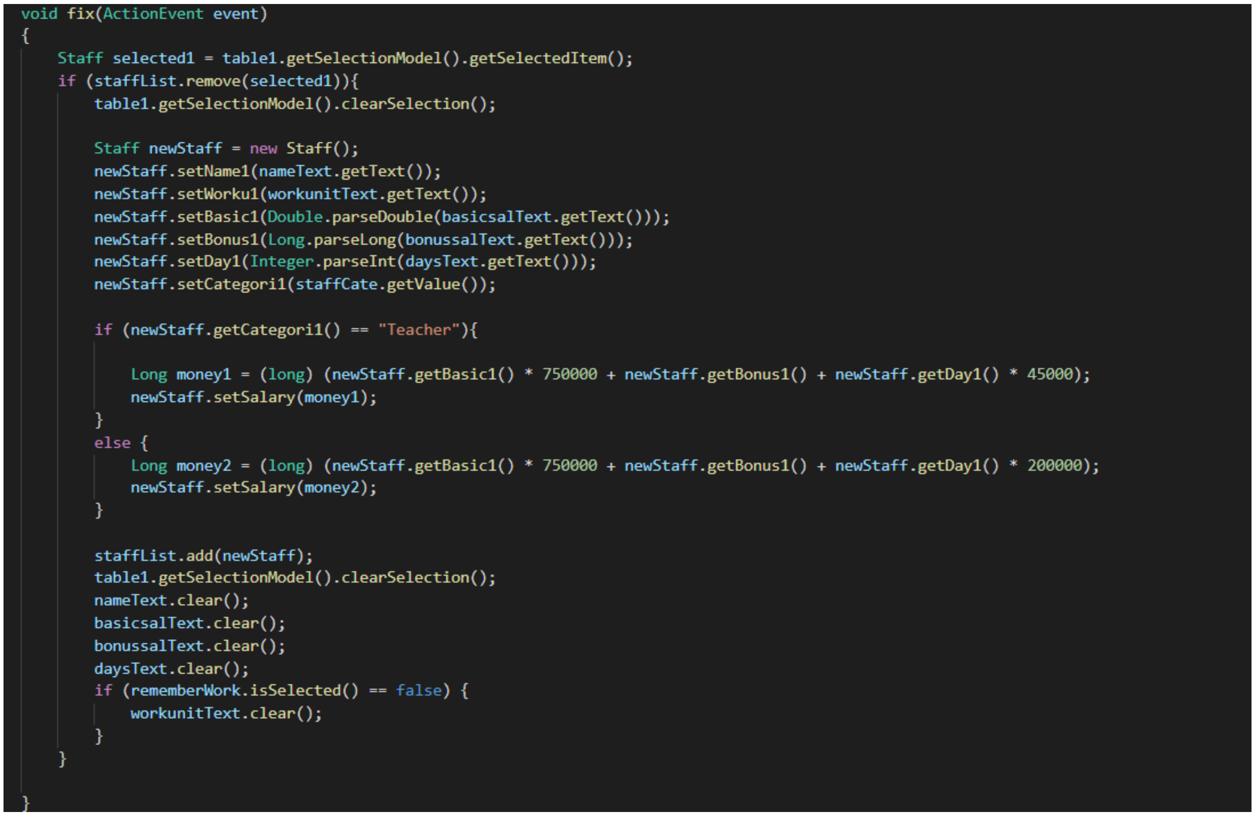
- MouseEvent is an event occurred by clicking the mouse cursor.

- Suppose Index is a model selected in table1 corresponding to in Array.

- If you choose nothing, meaning index <0 or index <=-1 , the return value is nothing or to say, nothing happen.

- With textField, nameText, Workutext,… we will take the text from name1Colunm and cellData respectively - indicating the corresponding value of index in Array.

Fix method



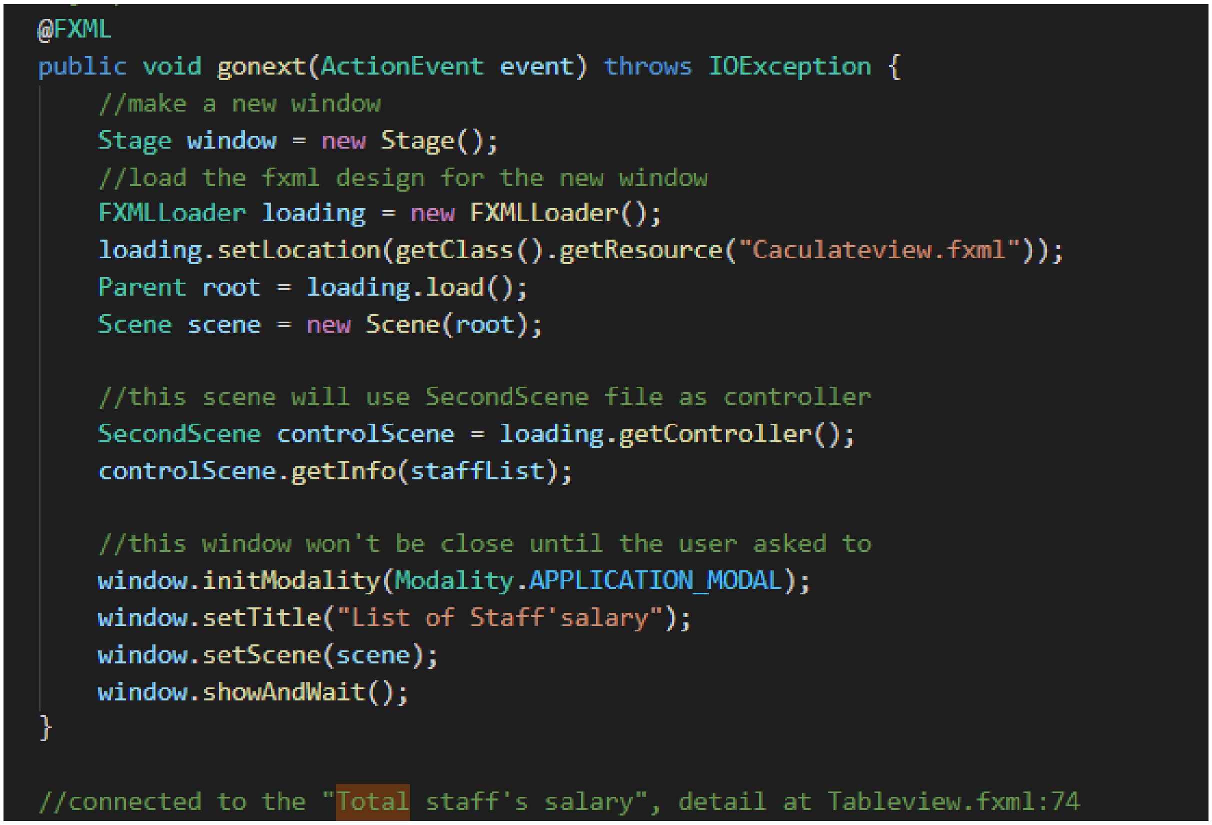
```
void fix(ActionEvent event)
{
    Staff selected1 = table1.getSelectionModel().getSelectedItem();
    if (staffList.remove(selected1)){
        table1.getSelectionModel().clearSelection();

        Staff newStaff = new Staff();
        newStaff.setName1(nameText.getText());
        newStaff.setWorku1(workunitText.getText());
        newStaff.setBasic1(Double.parseDouble(basicsalText.getText()));
        newStaff.setBonus1(Long.parseLong(bonussalText.getText()));
        newStaff.setDay1(Integer.parseInt(daysText.getText()));
        newStaff.setCategori1(staffCate.getValue());

        if (newStaff.getCategori1() == "Teacher"){

            Long money1 = (long) (newStaff.getBasic1() * 750000 + newStaff.getBonus1() + newStaff.getDay1() * 45000);
            newStaff.setSalary(money1);
        }
        else {
            Long money2 = (long) (newStaff.getBasic1() * 750000 + newStaff.getBonus1() + newStaff.getDay1() * 200000);
            newStaff.setSalary(money2);
        }

        staffList.add(newStaff);
        table1.getSelectionModel().clearSelection();
        nameText.clear();
        basicsalText.clear();
        bonussalText.clear();
        daysText.clear();
        if (rememberWork.isSelected() == false) {
            workunitText.clear();
        }
    }

}
```

- The concept of the fix method is that it will start by deleting the selected row and add to staffList the fixed row.

- At the firsts 3 lines, the method starts by getting the item which is selected in the table. Next, it will remove that item out of the staffList, if that item is presented in the staffList, the item will be removed in the staffList and its presence in the table will also be cleared.

- The next section will be the method setting value to the newStaff using the available TextField, and add them into the staffList (from …)). At the end, the selection is cleared, ready for the next task.

List of Staff’s Salary method



```
@FXML
public void gonext(ActionEvent event) throws IOException {
    //make a new window
    Stage window = new Stage();
    //load the fxml design for the new window
    FXMLLoader loading = new FXMLLoader();
    loading.setLocation(getClass().getResource("Caculateview.fxml"));
    Parent root = loading.load();
    Scene scene = new Scene(root);

    //this scene will use SecondScene file as controller
    SecondScene controlScene = loading.getController();
    controlScene.getInfo(staffList);

    //this window won't be close until the user asked to
    window.initModality(Modality.APPLICATION_MODAL);
    window.setTitle("List of Staff'salary");
    window.setScene(scene);
    window.showAndWait();
}

//connected to the "Total staff's salary", detail at Tableview.fxml:74
```

- gonext is an event where the user interacts by clicking button.

- Set scene includes the parent root and execute the loading method.

-The loading method source is Caculateview.fxml.

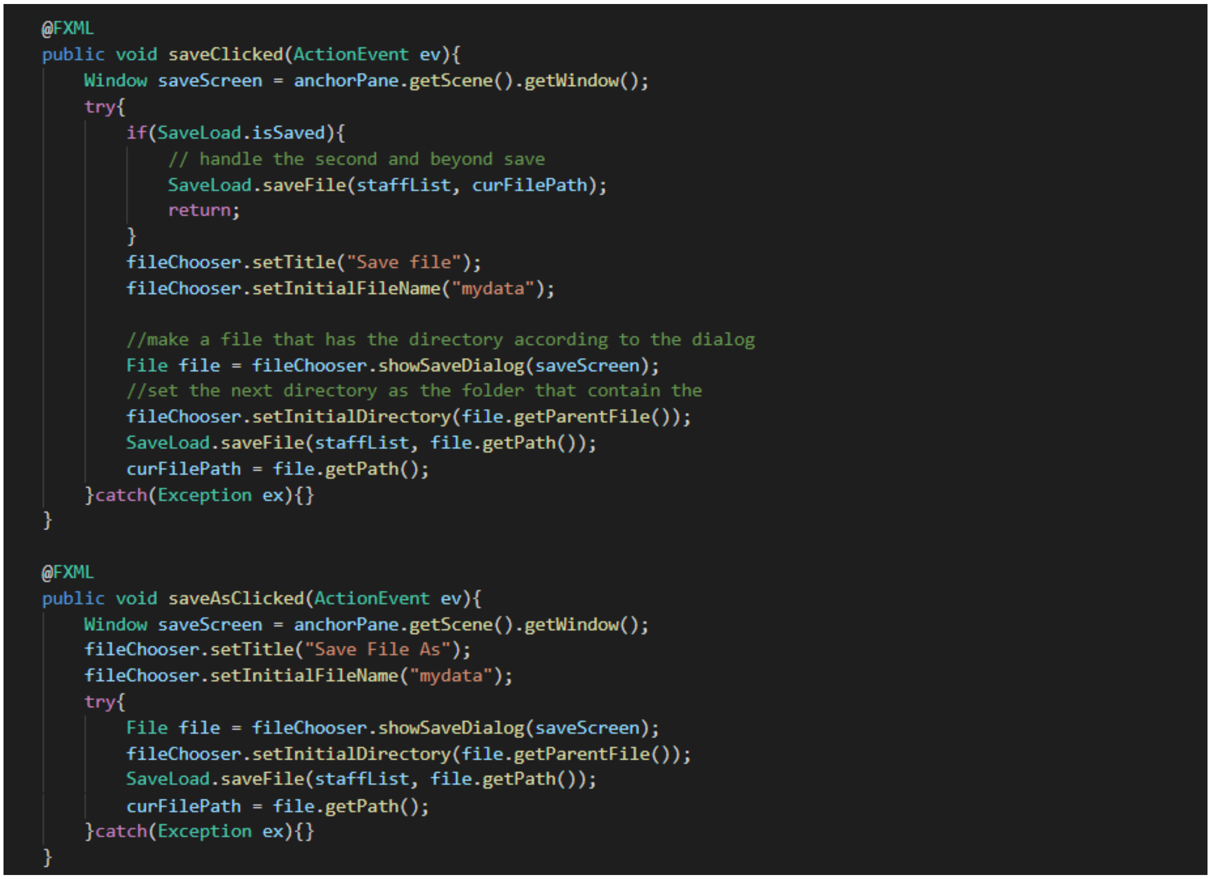
- Scene is set to be root.

- Class SecondScene will get the values function staffList through getInfo.

- Set title for the scene to be List of staff salary.

- Represent scene.

This is the same for gonext 2



```
@FXML
public void saveClicked(ActionEvent ev){
    Window saveScreen = anchorPane.getScene().getWindow();
    try{
        if(SaveLoad.isSaved){
            // handle the second and beyond save
            SaveLoad.saveFile(staffList, curFilePath);
            return;
        }
        fileChooser.setTitle("Save file");
        fileChooser.setInitialFileName("mydata");

        //make a file that has the directory according to the dialog
        File file = fileChooser.showSaveDialog(saveScreen);
        //set the next directory as the folder that contain the
        fileChooser.setInitialDirectory(file.getParentFile());
        SaveLoad.saveFile(staffList, file.getPath());
        curFilePath = file.getPath();
    }catch(Exception ex){}
}

@FXML
public void saveAsClicked(ActionEvent ev){
    Window saveScreen = anchorPane.getScene().getWindow();
    fileChooser.setTitle("Save File As");
    fileChooser.setInitialFileName("mydata");
    try{
        File file = fileChooser.showSaveDialog(saveScreen);
        fileChooser.setInitialDirectory(file.getParentFile());
        SaveLoad.saveFile(staffList, file.getPath());
        curFilePath = file.getPath();
    }catch(Exception ex){}
}
```

Save and Save As method

- **Save method**

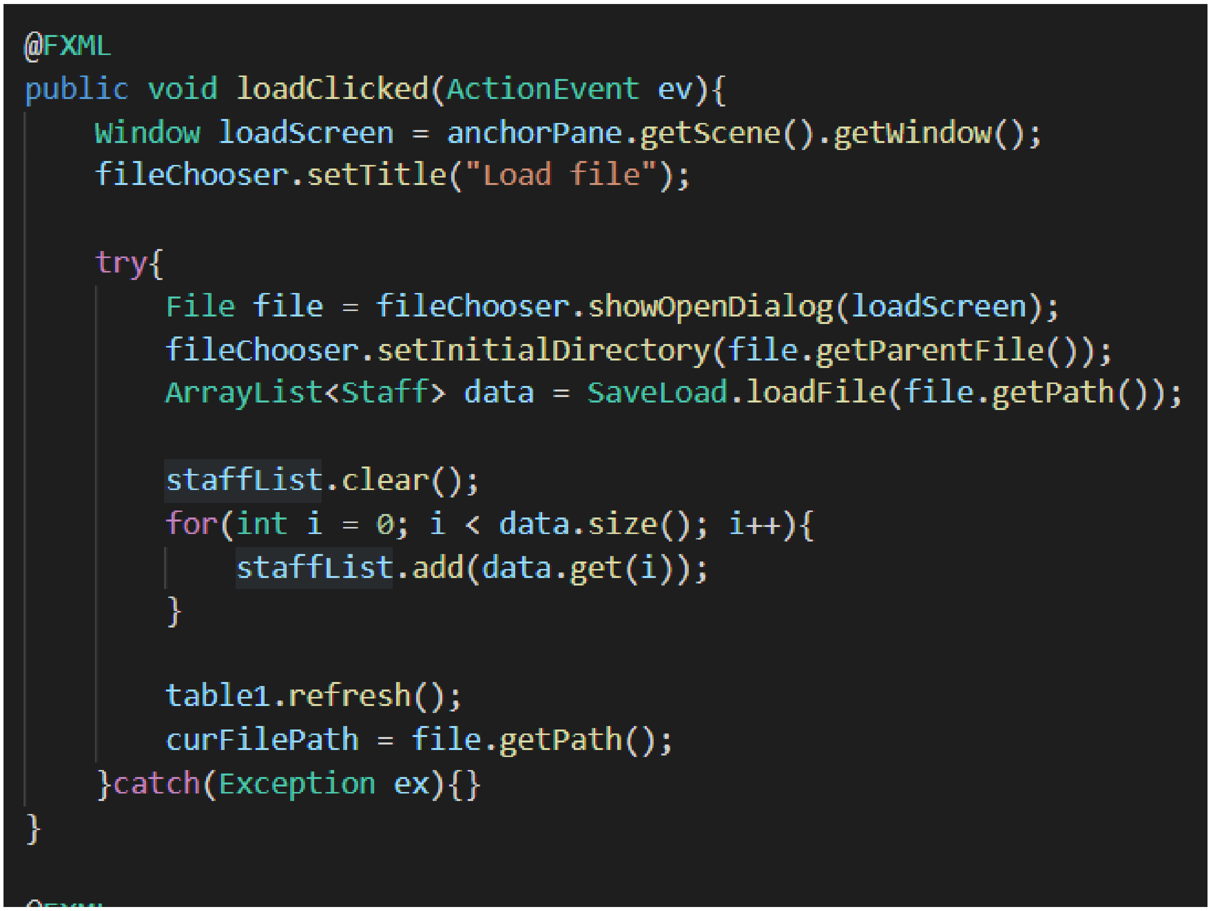
The save method has two ways of functioning: the first time and the times after the first.

For the first time it will: show a dialog by using fileChooser.showSaveDialog, which will show a saving dialog for you to make a file according to the directory you choose. Next, set the initial directory as the directory of the parent file (in short, a folder's directory contains the file you saved), so that the next time you want to open the dialog (could be load or save as dialogs) it will start at that directory. Finally, the saveFile() method from SaveLoad (see more in the SaveLoad analyze) class will write the data to the file and the curFilePath will have the path to the file for later use.

For the next time you save, because the static boolean attribute called isSave is altered from false to true when you used the SaveLoad.saveFile() method (or SaveLoad.loadFile() method; more on that in SaveLoad analysis), the method will jump straight to calling the method SaveLoad.saveFile() using the curFilePath which contain the directory of the data you working with and save it without bother you with another save dialog.

- **Save As method**

It can be said that Save As method is almost the same with the Save method, but whenever it is used, it always generates a save dialog for you to save a new file.

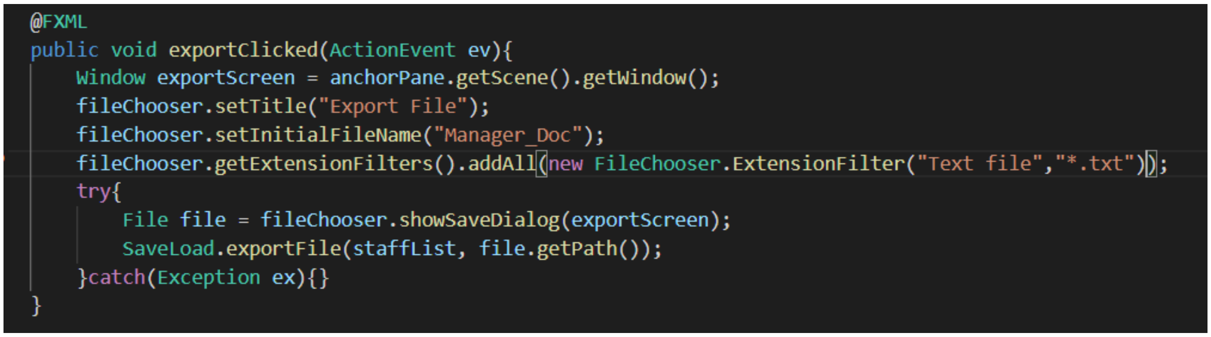


```
@FXML
public void loadClicked(ActionEvent ev){
    Window loadScreen = anchorPane.getScene().getWindow();
    fileChooser.setTitle("Load file");

    try{
        File file = fileChooser.showOpenDialog(loadScreen);
        fileChooser.setInitialDirectory(file.getParentFile());
        ArrayList<Staff> data = SaveLoad.loadFile(file.getPath());

        staffList.clear();
        for(int i = 0; i < data.size(); i++){
            staffList.add(data.get(i));
        }

        table1.refresh();
        curFilePath = file.getPath();
    }catch(Exception ex){}
}
```

Load method

The function of a load method consists of 2 jobs: load the data from the bin to the staffList and update the relevant element (such as the tableview, the binded sortedList..).

The load method starts by open an open’s dialog, where the user will choose the “.bin” file to be worked with. Next, the SaveLoad.loadFile() method will get the data into the directory and return the casted data into an ArrayList<Staff> object (called data). Then, it proceed to erase all the items in staffList and add all the element in data (the ArrayList<> object) in to the staffList through the method staffList.add(). After all is finished, the table1 will be refresh to keep it up to date with the renewed staffList. Finally, curFilePath will have the value of the loaded directory and ready for the next usage.



```
@FXML
public void exportClicked(ActionEvent ev){
    Window exportScreen = anchorPane.getScene().getWindow();
    fileChooser.setTitle("Export File");
    fileChooser.setInitialFileName("Manager_Doc");
    fileChooser.getExtensionFilters().addAll(new FileChooser.ExtensionFilter("Text file","*.txt"));
    try{
        File file = fileChooser.showSaveDialog(exportScreen);
        SaveLoad.exportFile(staffList, file.getPath());
    }catch(Exception ex){}
}
```

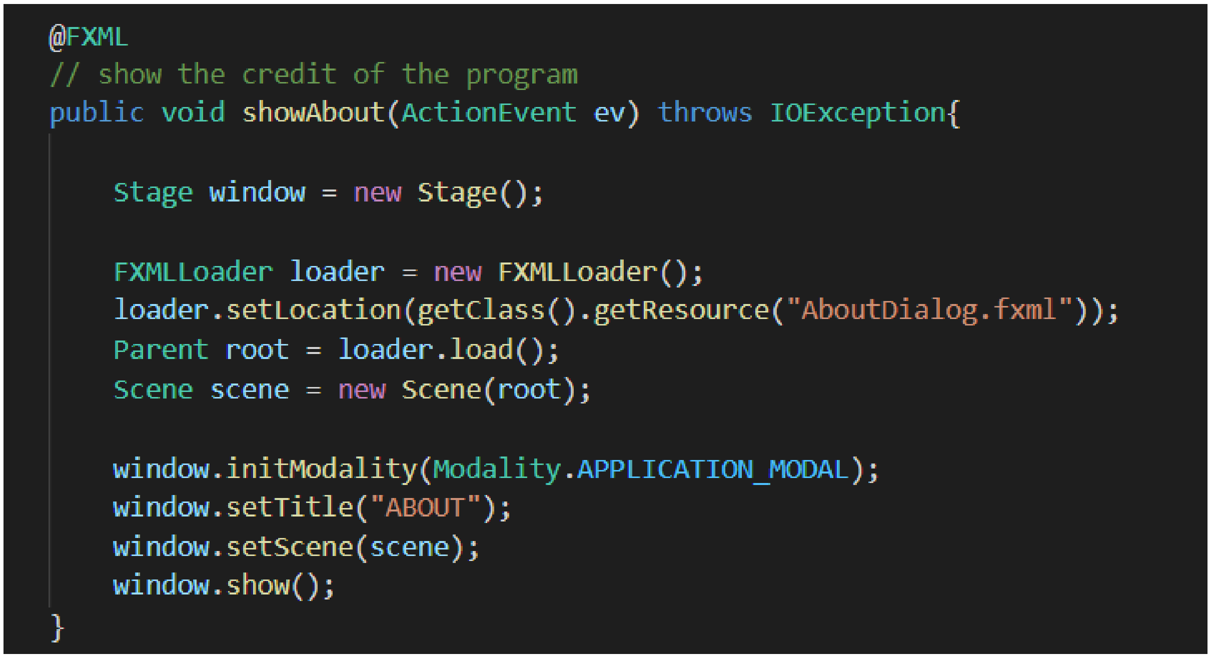
Export method

The export method and the save method are quite alike in term of functioning. However, the difference came from the fact that the save method create a file meant for later use while the export method create a file meant for reading the data from outside of the program.

Take a look at the 6th line, it adds another ExtensionFilter to the save dialog, meaning that aside from the “*.bin” filter, you can choose a “*.txt” for an alternative (keep in mind that it is best to use the “*.txt” alternative).

Beside, at the end of the method, it won’t update the directory to the exported file directory because there won’t be any usage from it.

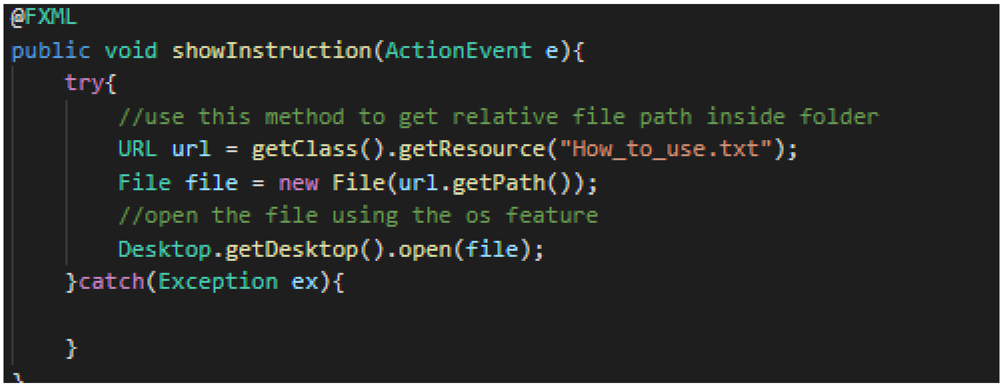
Show About method



```
@FXML
// show the credit of the program
public void showAbout(ActionEvent ev) throws IOException{

    Stage window = new Stage();

    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(getClass().getResource("AboutDialog.fxml"));
    Parent root = loader.load();
    Scene scene = new Scene(root);

    window.initModality(Modality.APPLICATION_MODAL);
    window.setTitle("ABOUT");
    window.setScene(scene);
    window.show();
}
```

Generally speaking, this method is alike the two “gonext” methods. On the other hand, this method only show the user credit of the creator of the program (AboutDialog.fxml) which does not have any controller.

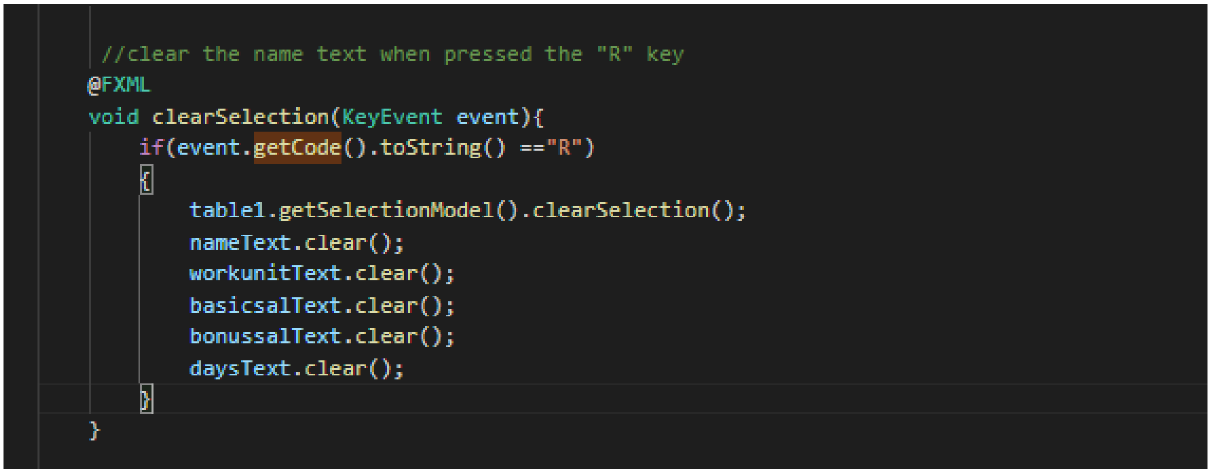
Show Instruction method



```
@FXML
public void showInstruction(ActionEvent e){
    try{
        //use this method to get relative file path inside folder
        URL url = getClass().getResource("How_to_use.txt");
        File file = new File(url.getPath());
        //open the file using the os feature
        Desktop.getDesktop().open(file);
    }catch(Exception ex){

    }
}
```

The method will open a document outside of the program which contains the manual and instruction on how to use the program.

First, it creates an URL (Uniform Resource Locator) object which help to locate the file inside the package. Next, create a File object with the path provided through the URL.getPath() method (which give the absolute path of the document). Finally, use the Desktop.getDesktop().open(file) which will open the file outside of the program.



```
//clear the name text when pressed the "R" key
@FXML
void clearSelection(KeyEvent event){
    if(event.getCode().toString() =="R")
    {
        table1.getSelectionModel().clearSelection();
        nameText.clear();
        workunitText.clear();
        basicsalText.clear();
        bonussalText.clear();
        daysText.clear();
    }
}
```

ClearSelection method

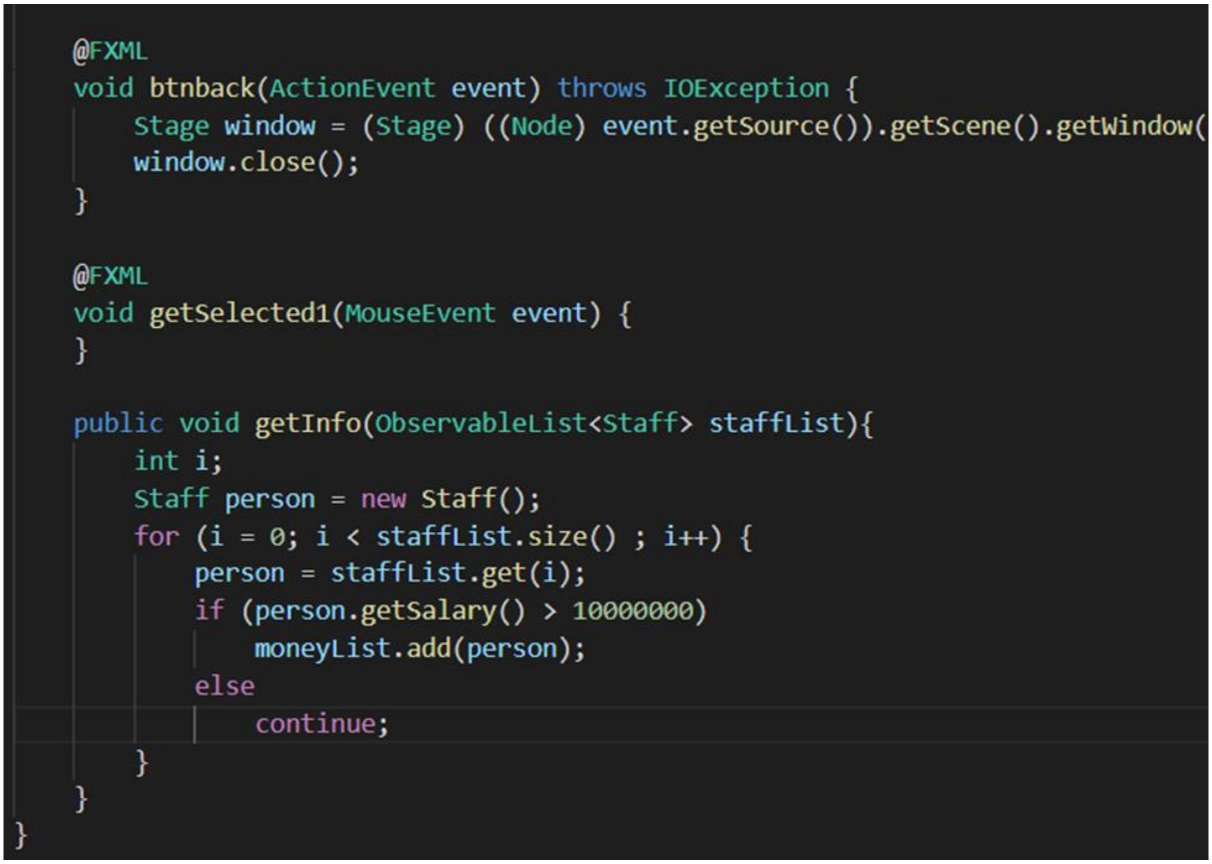
- If the return values of Keyevent in the form of String data type equal to R.

=> Clear selection in the tableview.

=> Clear the value inside nameText , workunitText , basicsalText , bonussalText and daysText field.

Class SecondScene

getInfo method



```
    @FXML
    void btnback(ActionEvent event) throws IOException {
        Stage window = (Stage) ((Node) event.getSource()).getScene().getWindow(
        window.close();
    }

    @FXML
    void getSelected1(MouseEvent event) {
    }

    public void getInfo(ObservableList<Staff> staffList){
        int i;
        Staff person = new Staff();
        for (i = 0; i < staffList.size() ; i++) {
            person = staffList.get(i);
            if (person.getSalary() > 10000000)
                moneyList.add(person);
            else
                continue;
        }
    }
}
```

- Declare i variable type int, person type Staff.

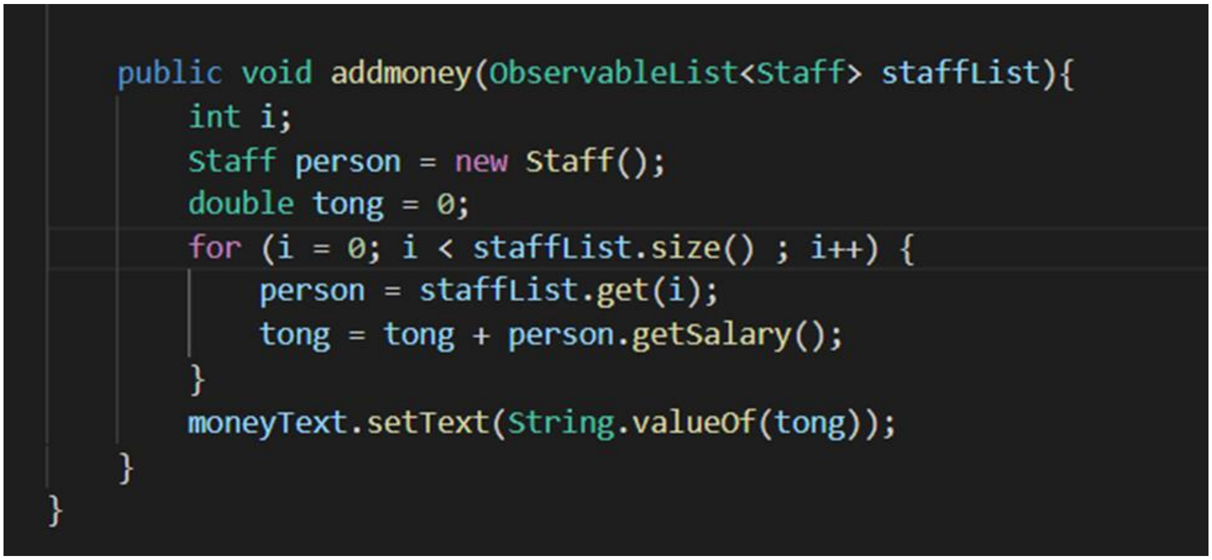
- Make a ‘ for ’ loop from i=0 to the corresponding size of staffList, each loop increase by 1 unit.

+) person takes the value of object no. i in staffList.

+) If the salary attribute ≥ 10000000 , add person into moneyList , if not then we could skip and continute the loop until the condetion cant meet anymore.

Class ThirdScene

addmoney method



```
    public void addmoney(ObservableList<Staff> staffList){
        int i;
        Staff person = new Staff();
        double tong = 0;
        for (i = 0; i < staffList.size() ; i++) {
            person = staffList.get(i);
            tong = tong + person.getSalary();
        }
        moneyText.setText(String.valueOf(tong));
    }
}
```

- Declare variable i type int , person type Staff and tong type double.

- Make a loop from i=0 to the corresponding size of staffList, each loop increase by 1 unit.

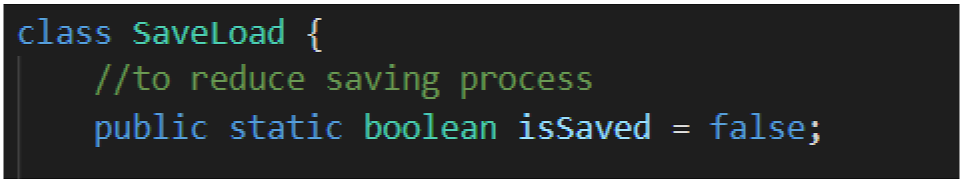
    +) person takes values of object no, i in staffList.

    +) tong = tong + salary value of person.

- The value of textField of moneyText is set to be displayed as String data type of tong.

SaveLoad Class

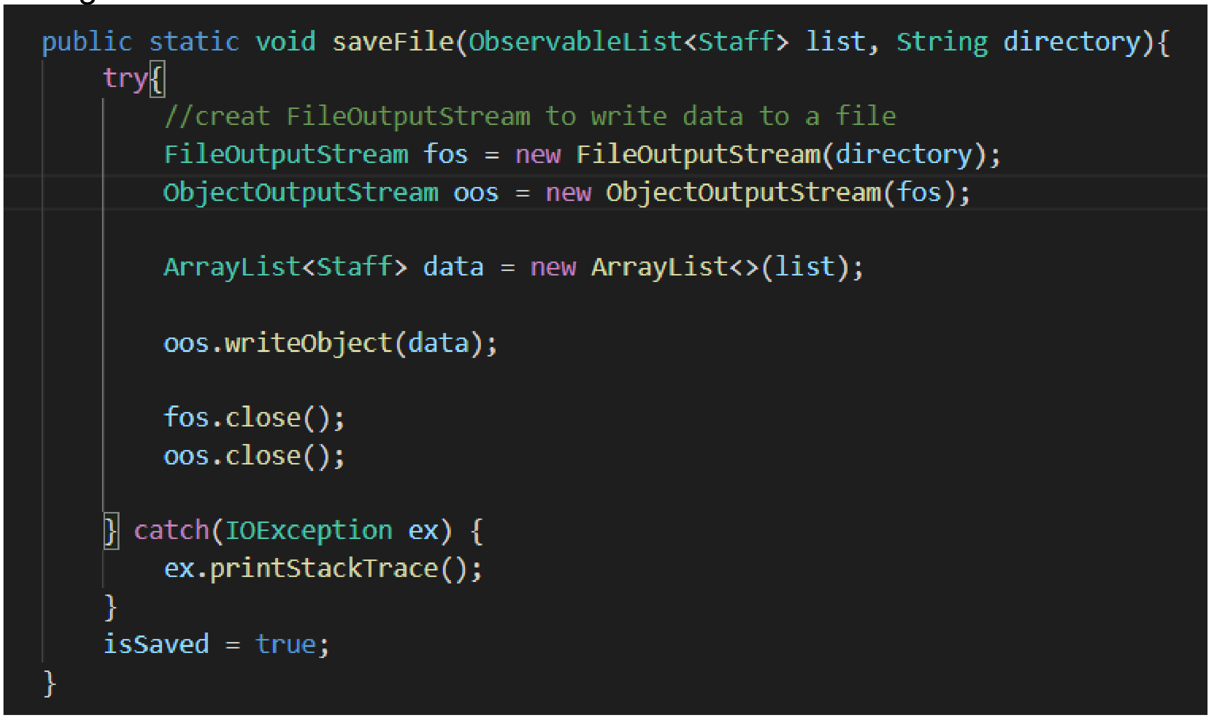
This class is created for the purpose of working with the data file and exporting the final text file contains the information of the staff. Keep in mind that most of the method won’t work properly if the object they work with doesn’t support java.io.Serializable, that’s why the Staff.class implemented Serializable interface.



```
class SaveLoad {
    //to reduce saving process
    public static boolean isSaved = false;
```

Static attribute isSaved

- This static attribute isSaved indicates the status of saving in the program. At the start of the program it will be “false” since there is no save or load action initiated. If one of the two methods saveFile() or loadFile are executed, the isSave will be changed into “true”

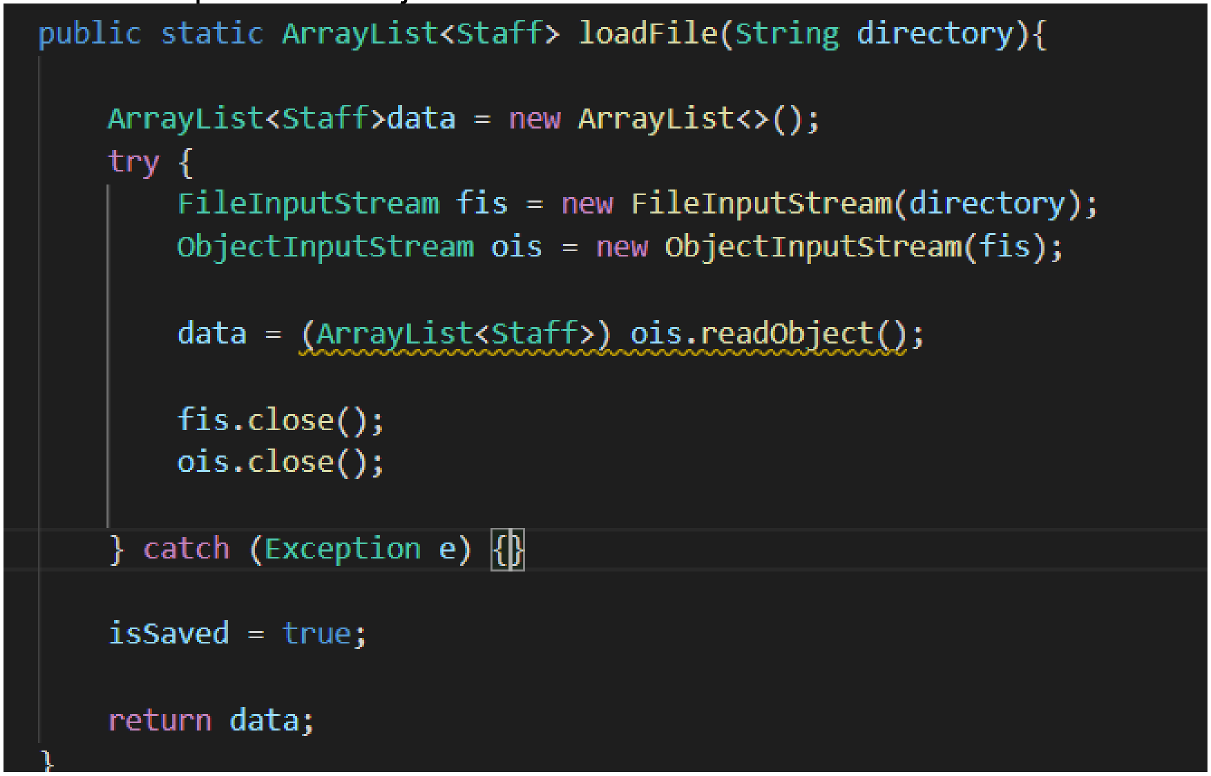


```java
public static void saveFile(ObservableList<Staff> list, String directory){
    try{
        //creat FileOutputStream to write data to a file
        FileOutputStream fos = new FileOutputStream(directory);
        ObjectOutputStream oos = new ObjectOutputStream(fos);

        ArrayList<Staff> data = new ArrayList<>(list);

        oos.writeObject(data);

        fos.close();
        oos.close();

    } catch(IOException ex) {
        ex.printStackTrace();
    }
    isSaved = true;
}
```

Save File method

- The two parameters in the saveFile() method are the ObservableList<Staff> object, which is the ObservableList user wanted to save the data, and the String directory which is the directory of the file user chosen to save.

- First, the method creates two objects: the FileOutputStream fos (which will help writing raw bytes data to a file which has the directory) and the ObjectOutputStream oos (which will help write the object data into the file through the OutputStream). Next, it creates a ArrayList<Staff> data which each of its element represent the ObservabelList’s items. Then, use the method ObjectOutputStream.writeObject to write the objects into the file. Finally, close both the OutputStream object and set the isSaved as “true”

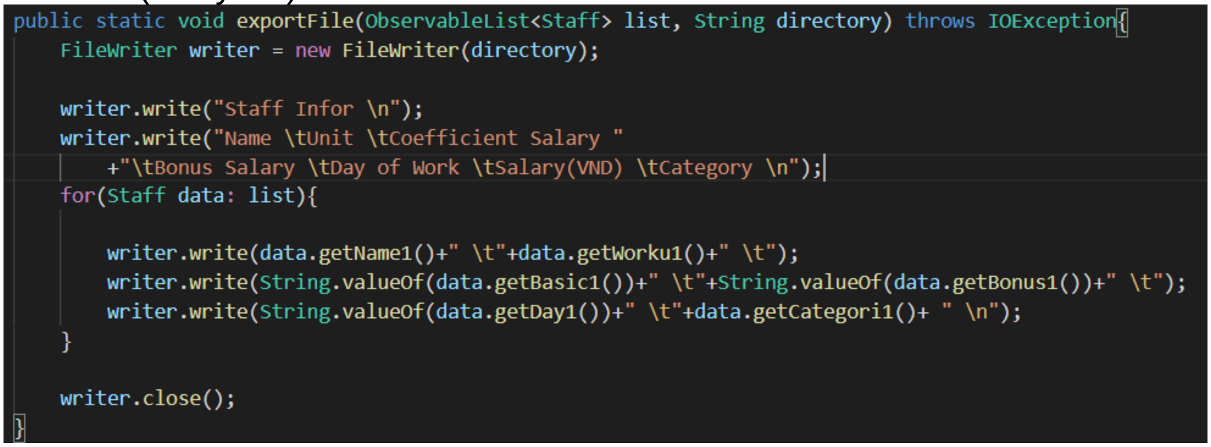


```
public static ArrayList<Staff> loadFile(String directory){

    ArrayList<Staff>data = new ArrayList<>();
    try {
        FileInputStream fis = new FileInputStream(directory);
        ObjectInputStream ois = new ObjectInputStream(fis);

        data = (ArrayList<Staff>) ois.readObject();

        fis.close();
        ois.close();

    } catch (Exception e) {}

    isSaved = true;

    return data;
}
```

Load File method

- This method returns the ArrayList<> object that contains the data from the previous save file. The method has one parameter: the String directory of the file user wishes to load/open.

- First, the method creates three objects: the empty ArrayList data, the FileInputStream fis (which will obtain the file data from file system) and the ObjectInputStream ois (which deserialize the previous written using the ObjectOutputStream). Next, cast the object which was read by the ObjectInputStream.readObject() method into the (ArrayList<Staff>) object, then set the data (ArrayList) to that. Finally, close two InputStream objects and return the data (ArrayList).



```java
public static void exportFile(ObservableList<Staff> list, String directory) throws IOException{
    FileWriter writer = new FileWriter(directory);

    writer.write("Staff Infor \n");
    writer.write("Name \tUnit \tCoefficient Salary "
        +"\tBonus Salary \tDay of Work \tSalary(VND) \tCategory \n");
    for(Staff data: list){

        writer.write(data.getName1()+" \t"+data.getWorku1()+" \t");
        writer.write(String.valueOf(data.getBasic1())+" \t"+String.valueOf(data.getBonus1())+" \t");
        writer.write(String.valueOf(data.getDay1())+" \t"+data.getCategori1()+ " \n");
    }

    writer.close();
}
```

Export method

- Unlike the other two methods that work with the raw bytes data. This method works with character-types data and saves them in the file system. There are two parameters: the ObservableList and the String directory.

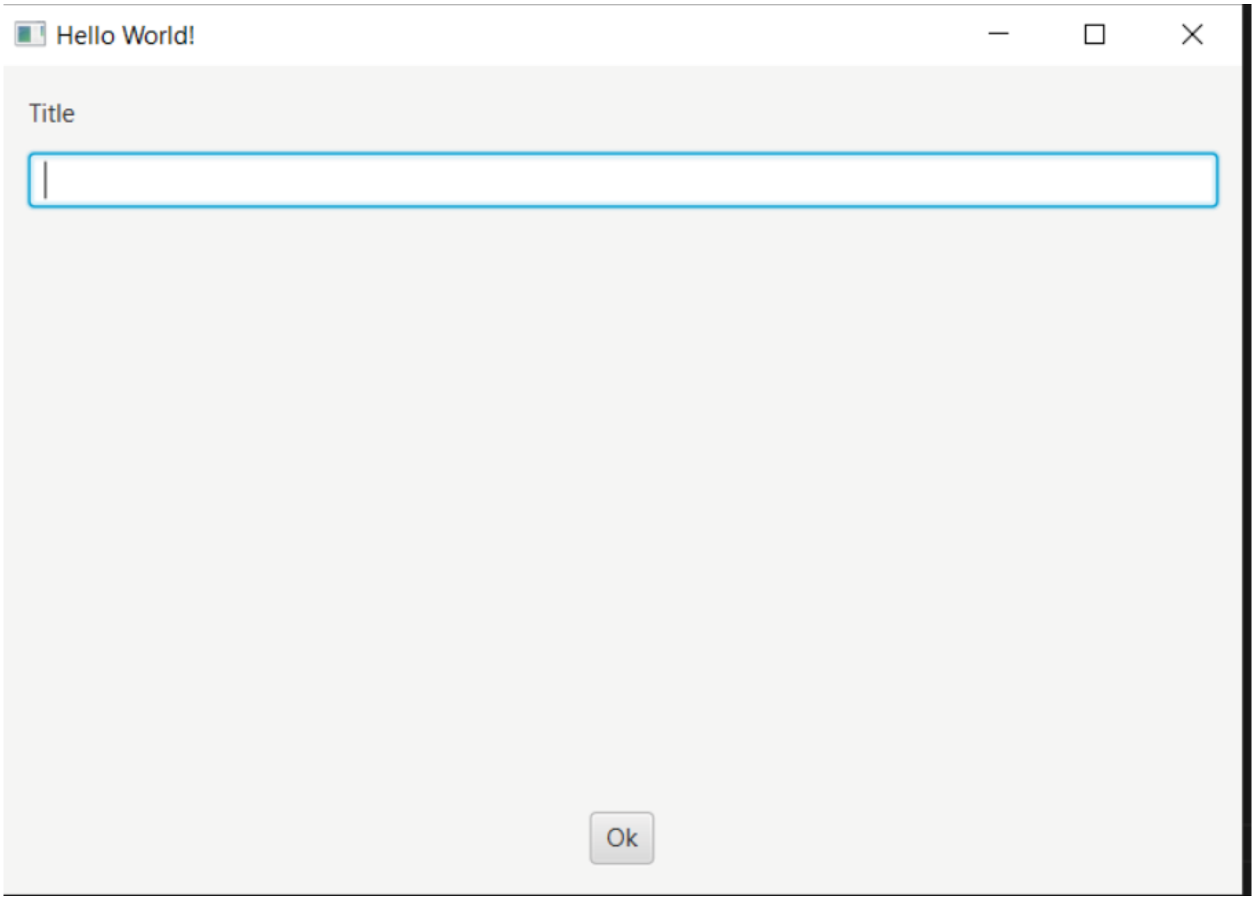
- First it creates a FileWriter object named “writer”, which helps write text to the file of the directory. Next, using the Writer.write() method which will write a String to the file, it formats the character file, starts with the head line (from 4th to 7th line) and then simultaneously writes the rest of the data.

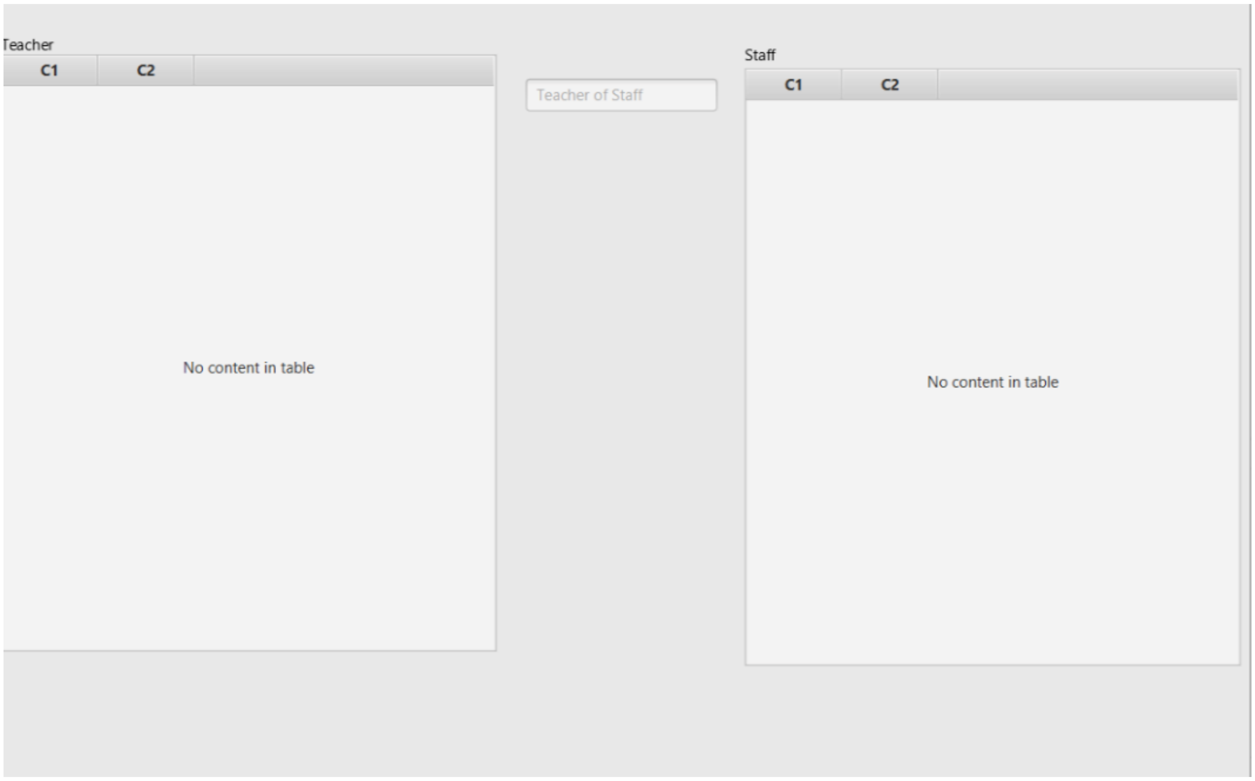
Close the “writer” object and finish.

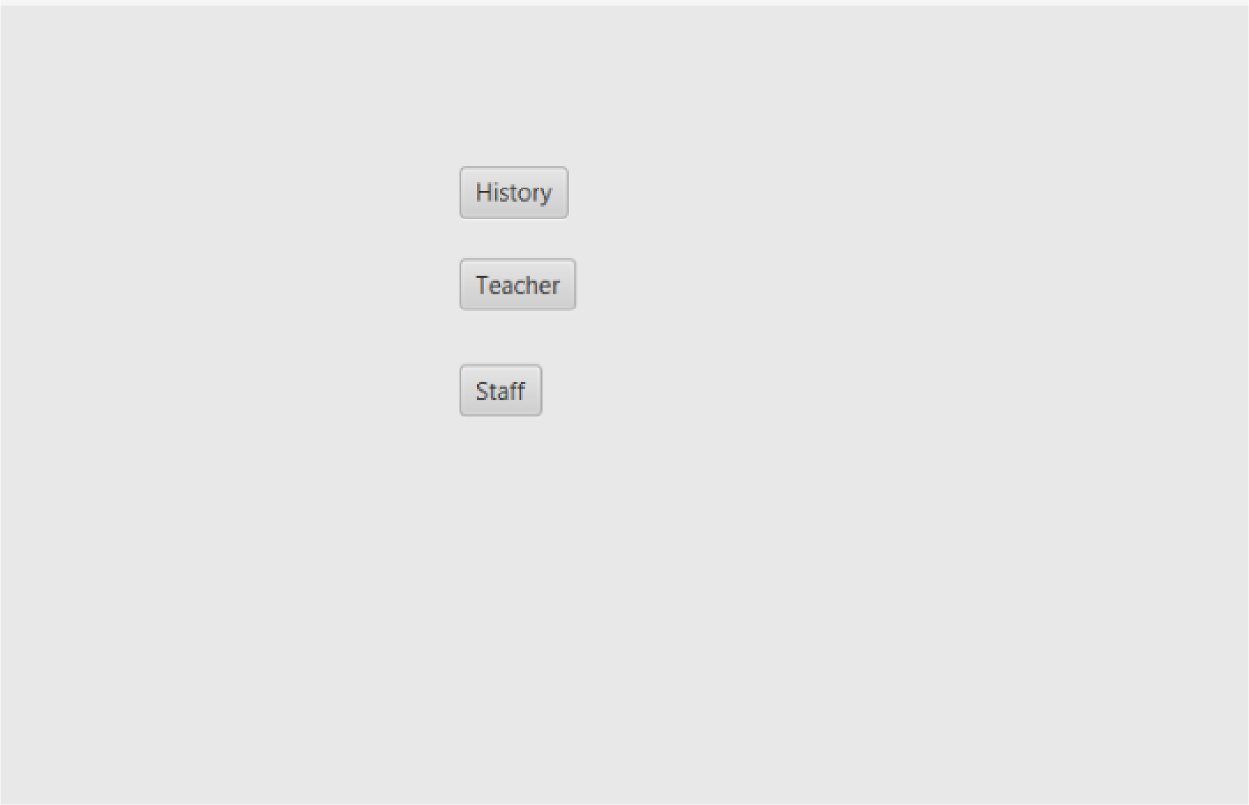
V.Problems

This is one of the biggest university projects we have had so far, so many problems occurred during our working process. JavaFX and GUI is something very new for us so it took us a hard time to get acquainted with. Before we used the Scene Builder we had had to code every scenes, little by little which took us a lot of time and effort just to have a glimpse of what was going right, understood how does it function,…

(some pictures in the early process)



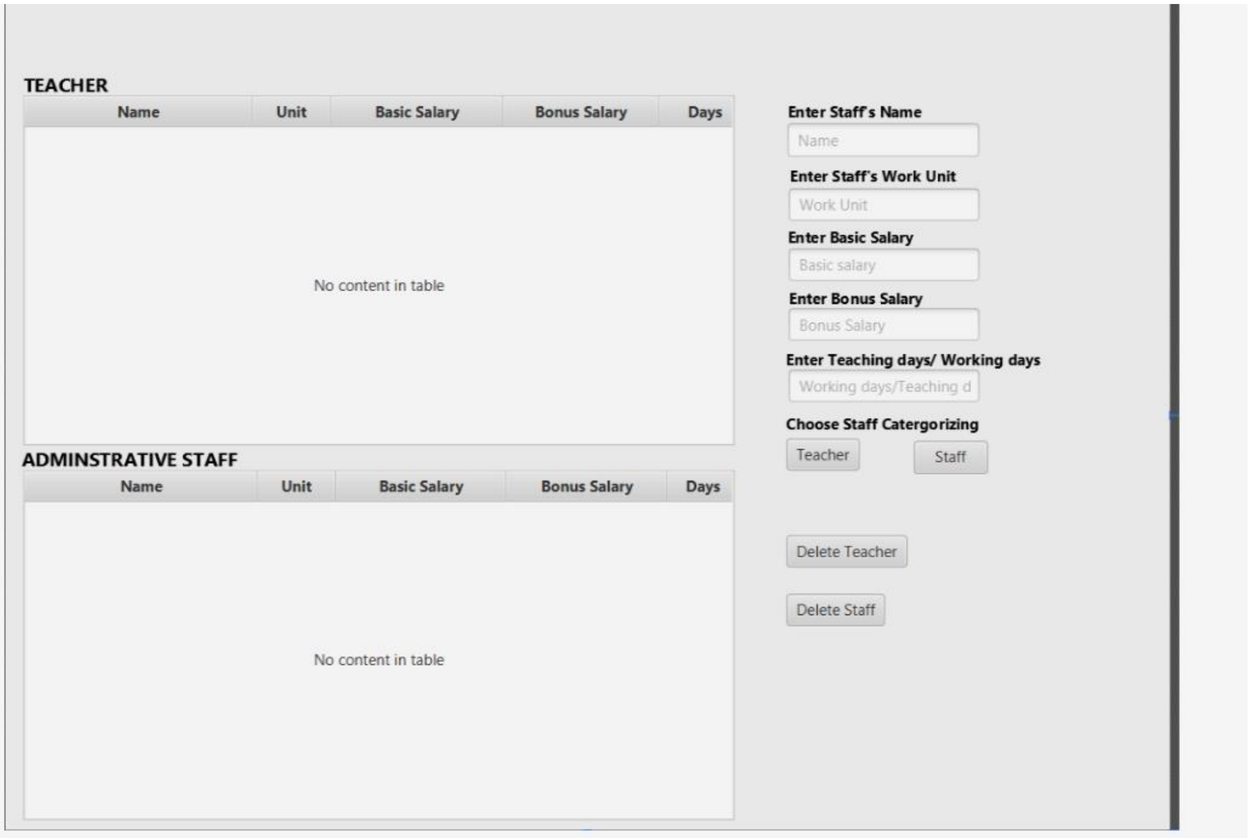




However, thourgh hardworking and dedication, we have managed to do it.

Another problem is plans don't always work in the way we wanted them to. Especially the designing.

Let’s take a look at this picture:



TEACHER

Name	Unit	Basic Salary	Bonus Salary	Days

No content in table

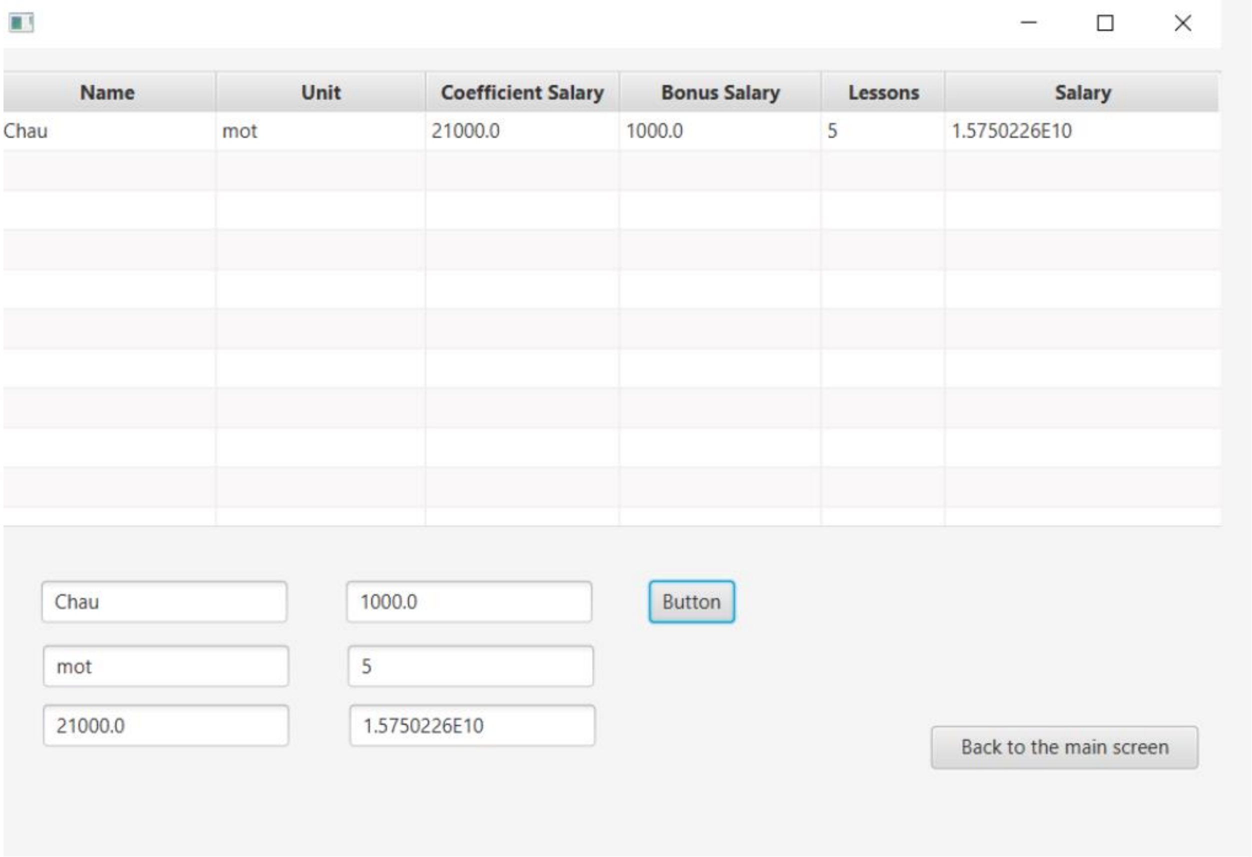
ADMINSTRATIVE STAFF

Name	Unit	Basic Salary	Bonus Salary	Days

No content in table

(old Tableview.fxml - Mainscene)

This is one of the first displays we made. So we planned to make 2 table: One will contain teacher object and the other is administrative staff. The problems took place when Nghia was in the SecondScene coding process. We found out that in order to calculate the Salary we have to run the info in text field before loading to the calculate tableview and only 1 staff can be added.

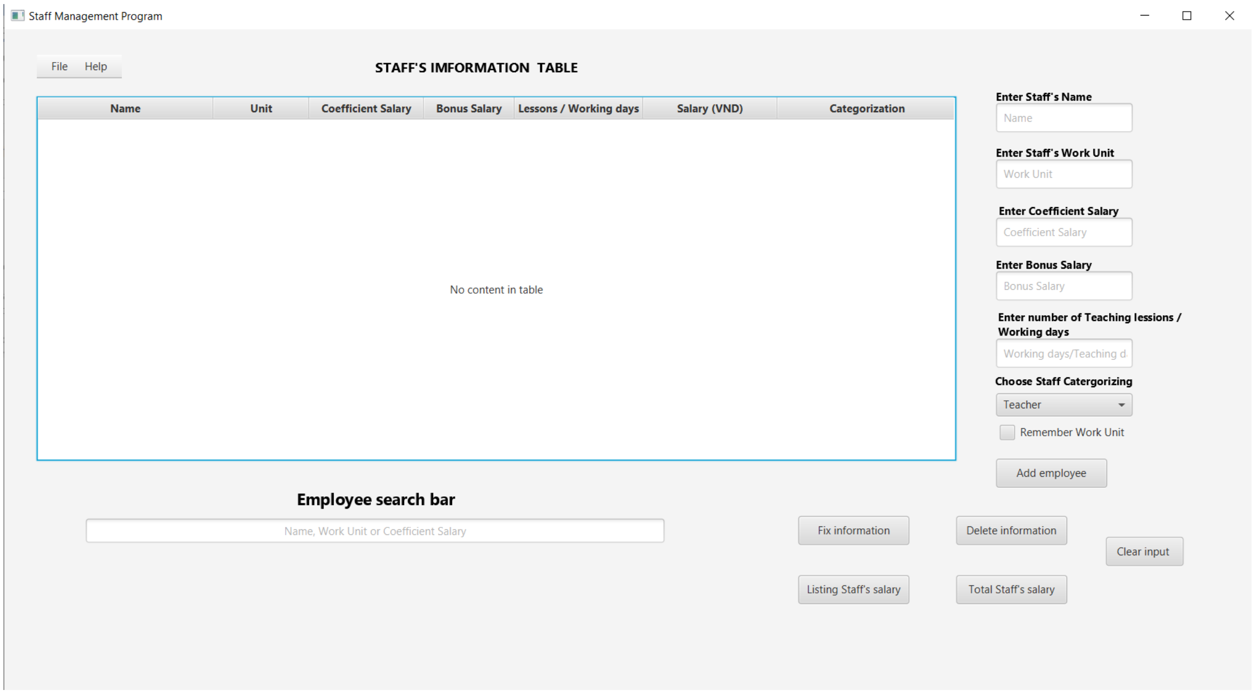


Name	Unit	Coefficient Salary	Bonus Salary	Lessons	Salary
Chau	mot	21000.0	1000.0	5	1.5750226E10

Chau
1000.0
mot
5
21000.0
1.5750226E10

(old Caculateview.fxml - SecondScene)

After a long conversation on the next Friday, we finally decided to re-code for the second times and re-design everything from scratch, which lead to the improved version you see today.

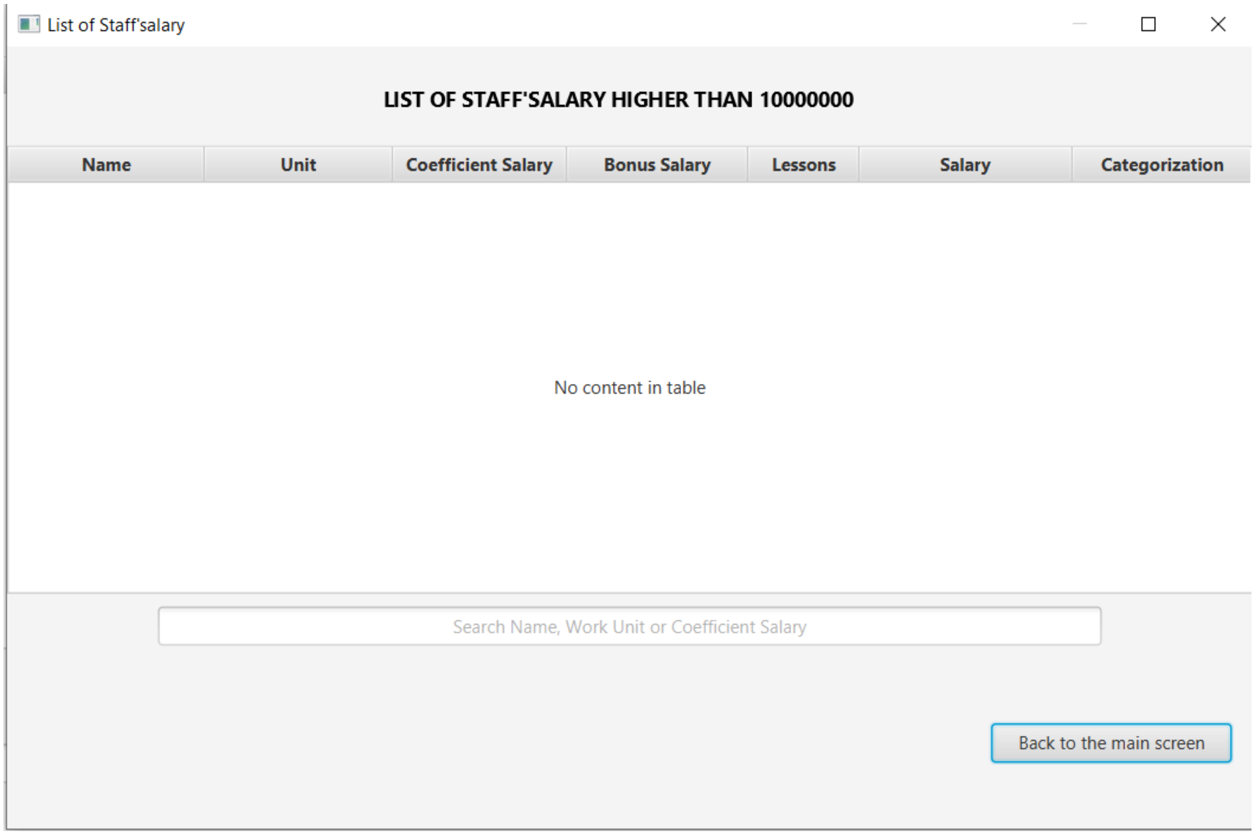


Staff Management Program

STAFF'S IMFORMATION TABLE

Name	Unit	Coefficient Salary	Bonus Salary	Lessons / Working days	Salary (VND)	Categorization

No content in table

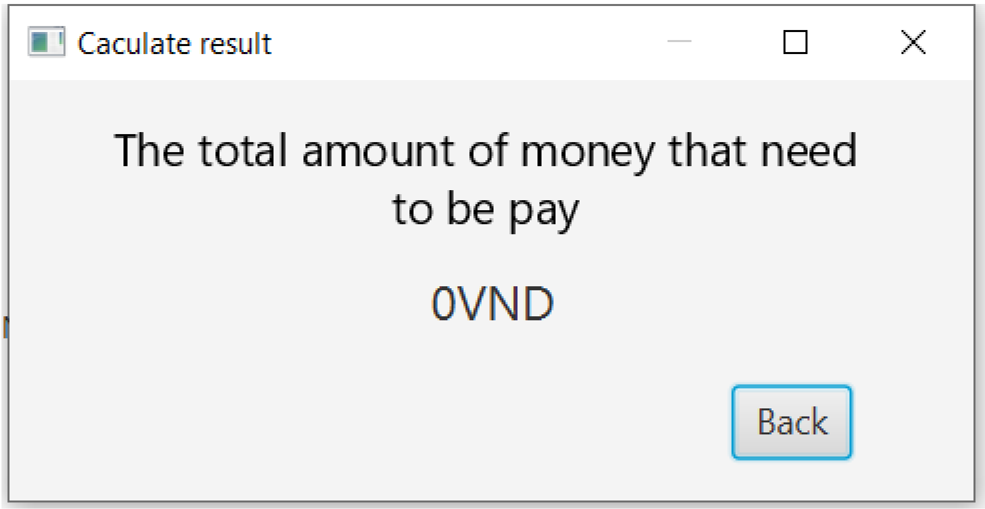


List of Staff'salary

LIST OF STAFF'SALARY HIGHER THAN 10000000

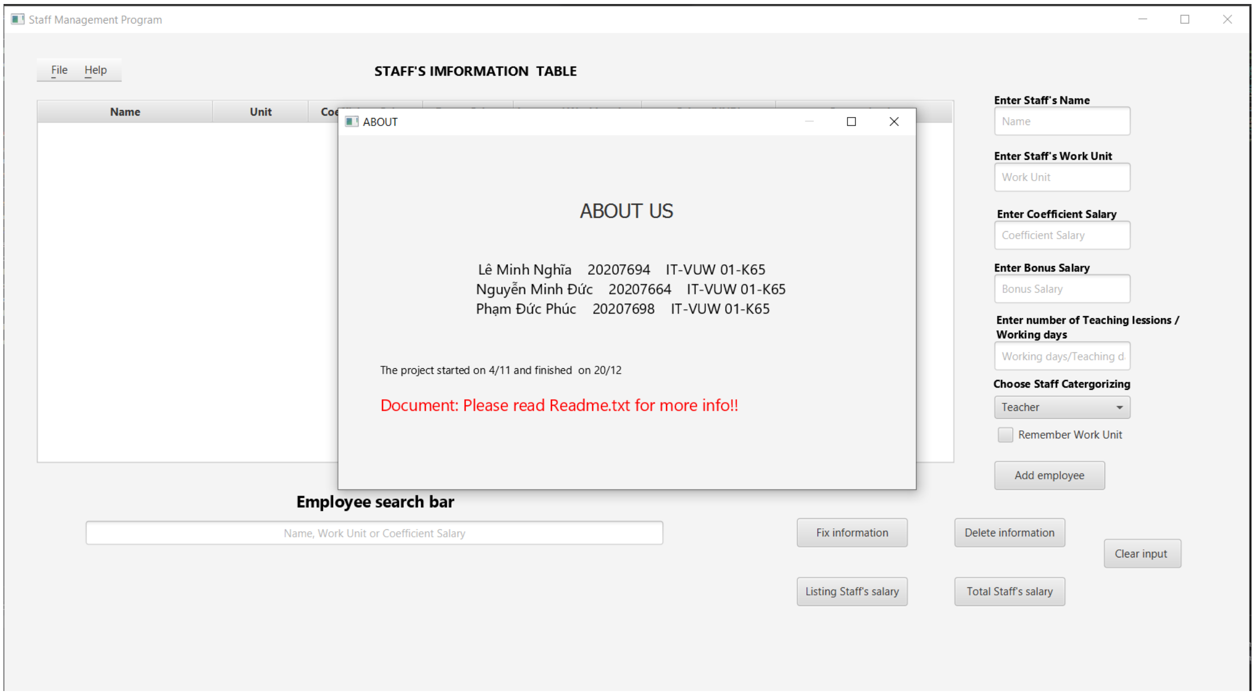
Name	Unit	Coefficient Salary	Bonus Salary	Lessons	Salary	Categorization

No content in table



The total amount of money that need to be pay

0VND



ABOUT US

Lê Minh Nghĩa 20207694 IT-VUW 01-K65
Nguyễn Minh Đức 20207664 IT-VUW 01-K65
Phạm Đức Phúc 20207698 IT-VUW 01-K65

The project started on 4/11 and finished on 20/12

Document: Please read Readme.txt for more info!!

Because the mainscene contains many functions so a lot of small bugs we also have to fix and change every week. And before the day deadline comes, we still have to fix it.

Beside those coding problems, another problem we had to face were preparation and communication. Because everything was exchanged through the internet due to the affects of Covid-19, sometimes we misunderstood each other's ideas. Although each Friday we had a meeting call, but we still couldn’t solve this problem completely.

Still, the version you see today is not the idea version according to our plan, but i think we had done it, at least to an acceptable level.

As for each member of our teams, we have different problems as well

Nguyen Minh Duc: Just for the Mainscene I have to code it for 5 to 6 times, to test every functions and every possible ways to code with javafx library. From pure coding to using Scene builder program just to know how it works took me ton of times and effort. Lots of bugs during the coding process made my code look like a mess.

Le Minh Nghia: This is one of the most stressful but informative experiences I have . First time i learn about and work with SceneBulider and probably not the last time.

Pham Duc Phuc: After all, this project is quite the challenge, especially during the Covid seasons where all three of us have to do it without any actual meeting outside of Facebook or Team. To be frank, in the beginning of the project, I was the burden of the team due to my slow learning habits, but the moment I fully grasped the concept of the project and started contributing, I found it fascinating and interesting.

Final Thought.

Although this project is quite hard and sometimes make us tremble, it helped us realize the limitation of a human person. None of us can do this project alone. (I could do exceptionally good at designing but if we are talking about coding, Nghia would definitely be my best choice for a teammate. He assisted me during the debug coding process as best as possible and also rearranged them to look more refined. Phúc is also a keeper, although he was slow at first but after just a while he caught up with the pace and outperformed himself and made us speechless with such flawless results.

Like a saying: “Two heads are better than one”, maybe when we work together as a team like this would we recognized our human limitations and surpass all of them. Something that I could not have done (especially the coding part) was finised with the help of someone else. It was thanks to this project could I had met with such a good opportunity to work with such talented people and improve on my own behalf. Despite being a University project, this project suits to be the stepping stone for me to get to an even greater, greater height.