**JAVA THEORY**

**DIGITAL ASSIGNMENT-1**

**JAVAFX GUI PROJECT**

**NAME- SAURABH SINGH**

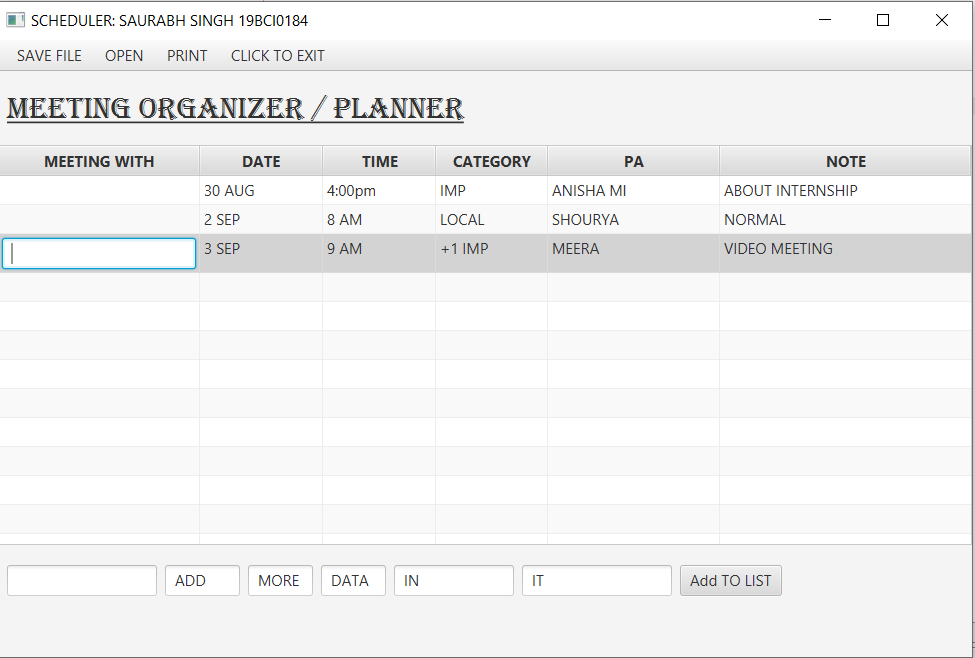
**ID- 19BCI0184**

**GUI NAME- SCHEDULER**

**(MEETING ORGANIZER / PLANNER)**

**TEACHER- PRIYA G**

**FRONTPAGE-**

****

**DESCRIPTION OF ‘SCHEDULER’ :-**

THIS ‘**SCHEDULER**’ GUI MADE IN JAVAFX ALLOWS US TO:

* ORGANIZE MEETINGS/ PLAN MEETINGS
* ADD DATA / ORGANIZE DATA
* MAINTAIN TIME TABLES
* OPEN TO SEE EXISTING MEETINGS / EDIT MEETINGS
* SAVE THE MEETINGS
* PRINT THE PAGE OF PLANNED
* EXIT APP

**SOFTWARE USED:**

* JAVA SDK-14
* JavaFX SDK-11.0.2
* NETBEANS 8.2

**CONCEPTS USED:-**

JAVAFX-

JavaFX is a Java library used to develop variety of applications . The applications built in JavaFX, can run on multiple platforms including Web, Mobile and Desktops.

FEATURES-

**Java Library:** consists of many classes and interfaces that are written in Java.

**Built in UI controls** : Built in UI component are just enough to develop a full featured application.

**SCENE:**

The JavaFX Scene object is the root of the GUI. The JavaFX Scene contains all the visual JavaFX GUI components inside it and is represented by the class javafx.scene.Scene. A Scene object has to be set on a JavaFX Stage to be visible.

**STAGE:**

A JavaFX Stage, javafx.stage.Stage, represents a window in GUI. Inside a JavaFX Stage you can insert a JavaFX Scene which represents the content displayed inside a window - inside a Stage.

**JavaFX Label :**

A component that is used to define a simple text on the screen.

*javafx.scene.control.Label*, used to describe the purpose of the other components to the user.

**Button:**

A component that controls the function of the application and is used to create labelled button.

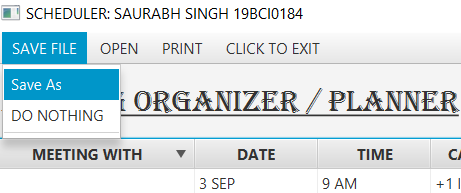
**TextField:**

Used to get the input from the user in the form of text. javafx.scene.control.TextField represents TextField.

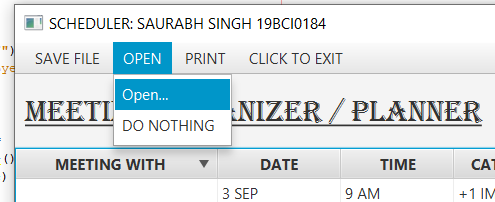
**Menu:**

Implement menus. In this GUI , in the menubar there are various menus.

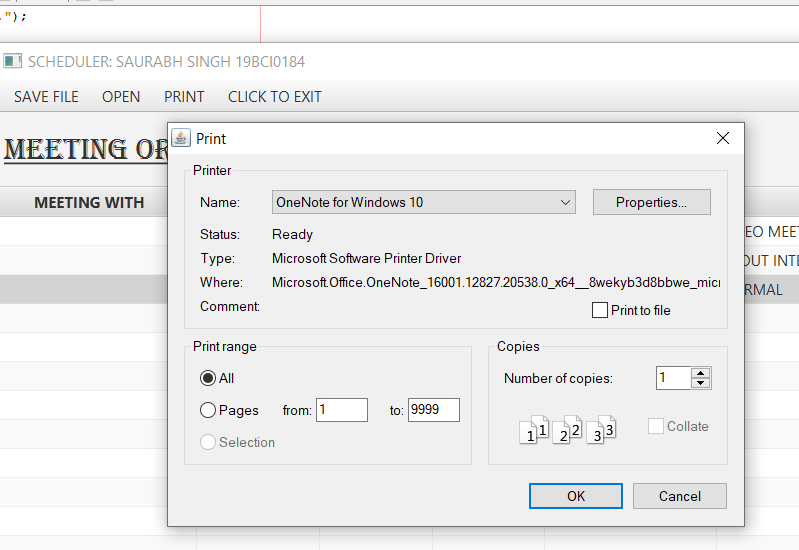
* SAVE- saves the file.



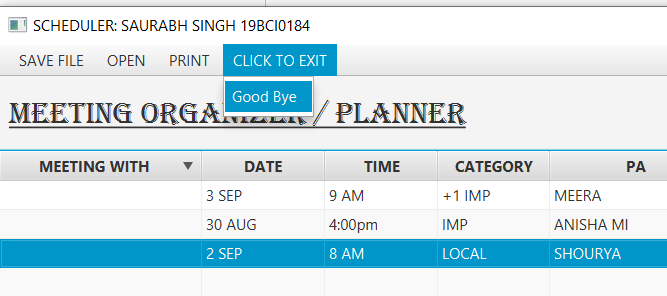
* OPEN- open an existing file.



* PRINT- print the present page.



* EXIT- exit the window.



In this I also implemented drop down menus by using menuitems as you can see in above pics.

* SAVE
* DO NOTHING
* GOOD BYE

These are implemented in this GUI using Event Handler.

**Event and Event handling:**

In JavaFX, An event is occurred whenever the user interacts with the application nodes. They tell us something happened at user’s end.

Eg- Mouse clicks to implement or taking input from user using keyword etc.

**TWO TYPES:**

1. **Foreground events** are generated as consequence of a personinteracting with the graphical components in a GUI (clicking a button, moving the mouse, etc.).
2. **Background events** are those that require the interaction ofthe end user. For example, hardware or software failure, timer expiry.

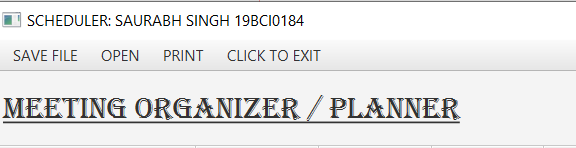
In this GUI, we are taking input of meeting data from user and when clicking on add button it adds the data in the table. We can also edit the data in table directly.

**setOnAction()**

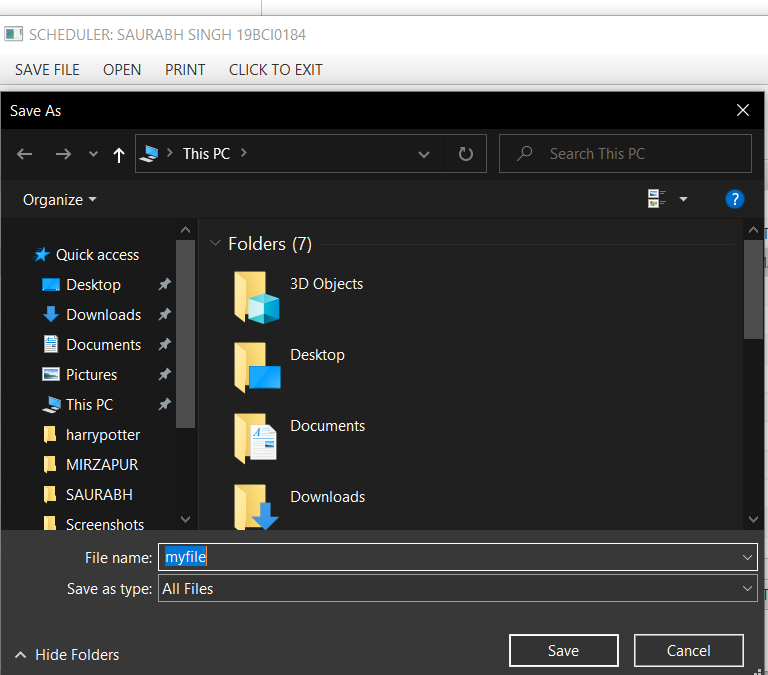
The EventHandler registered with the setOnAction() method is called when the Play button is clicked and it is set to rotate the rectangle on the screen.

**SOME PICS OF OPERATION:**

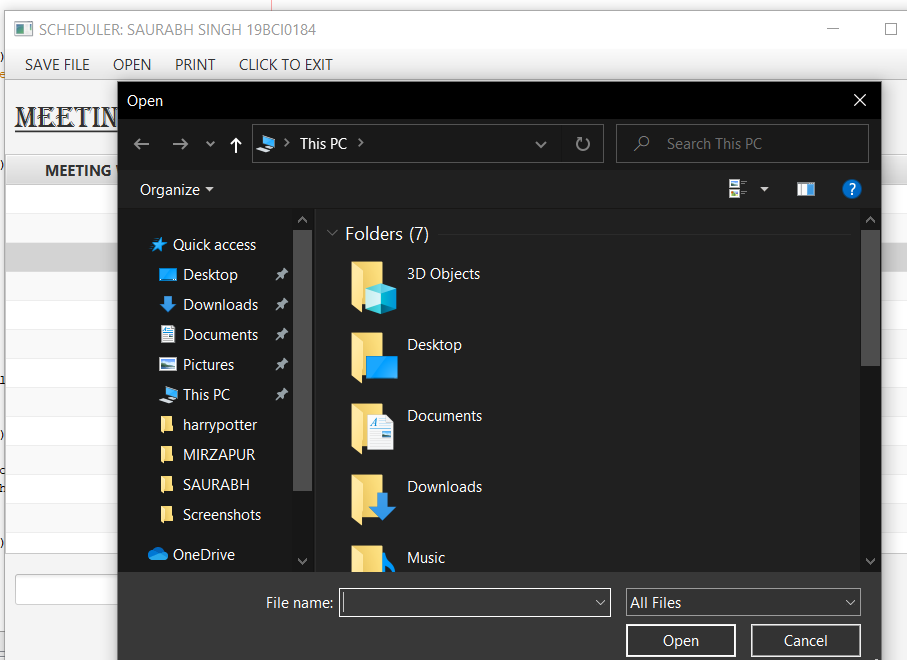
**MENUS:**

****

**SAVE:**

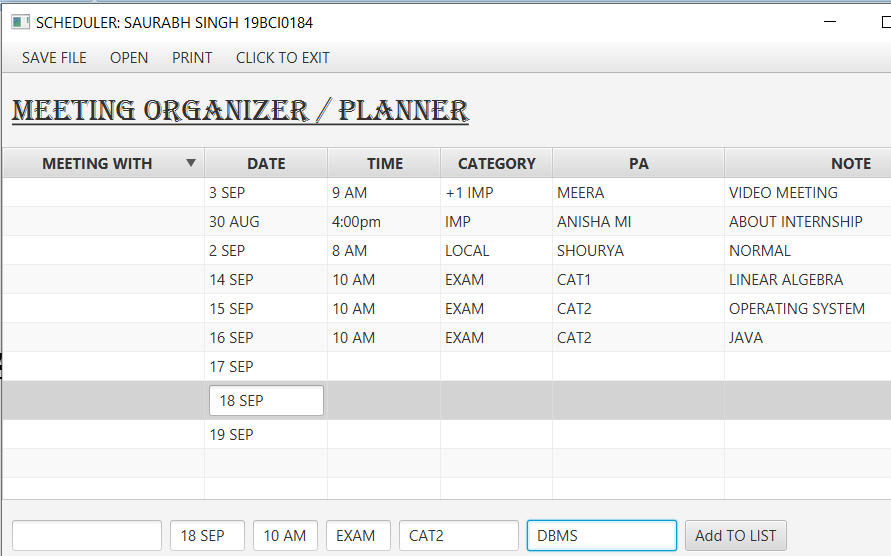
****

**OPEN:**

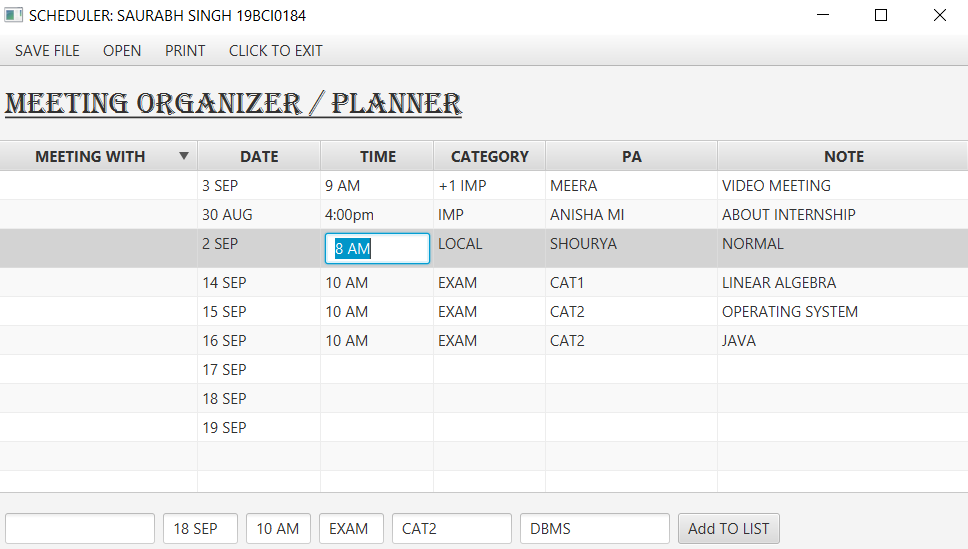
****

**ADD DATA:**

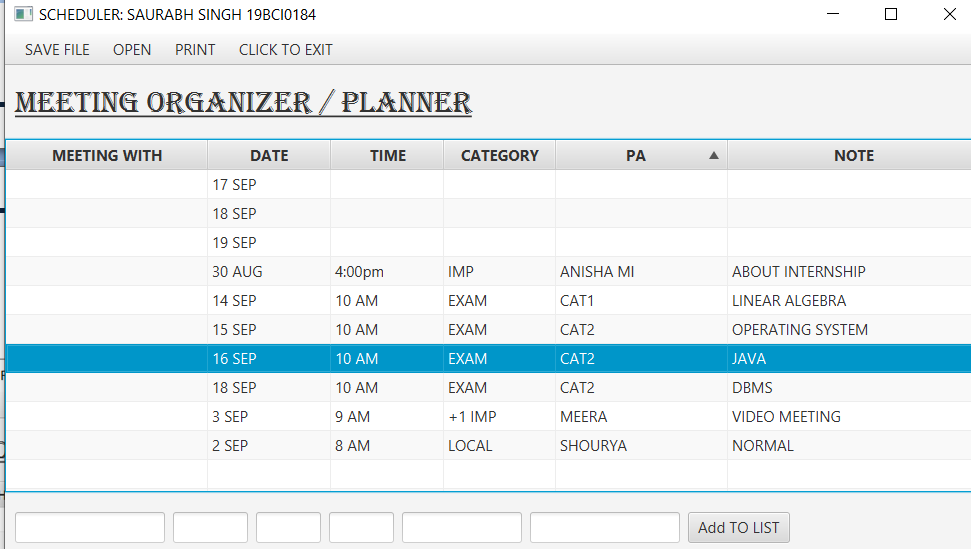
WE CAN ADD DATA BY ENTERING DATA ACCORDING TO TABLE AND THE BY CLICKING THE “ADD TO LIST” BUTTON.



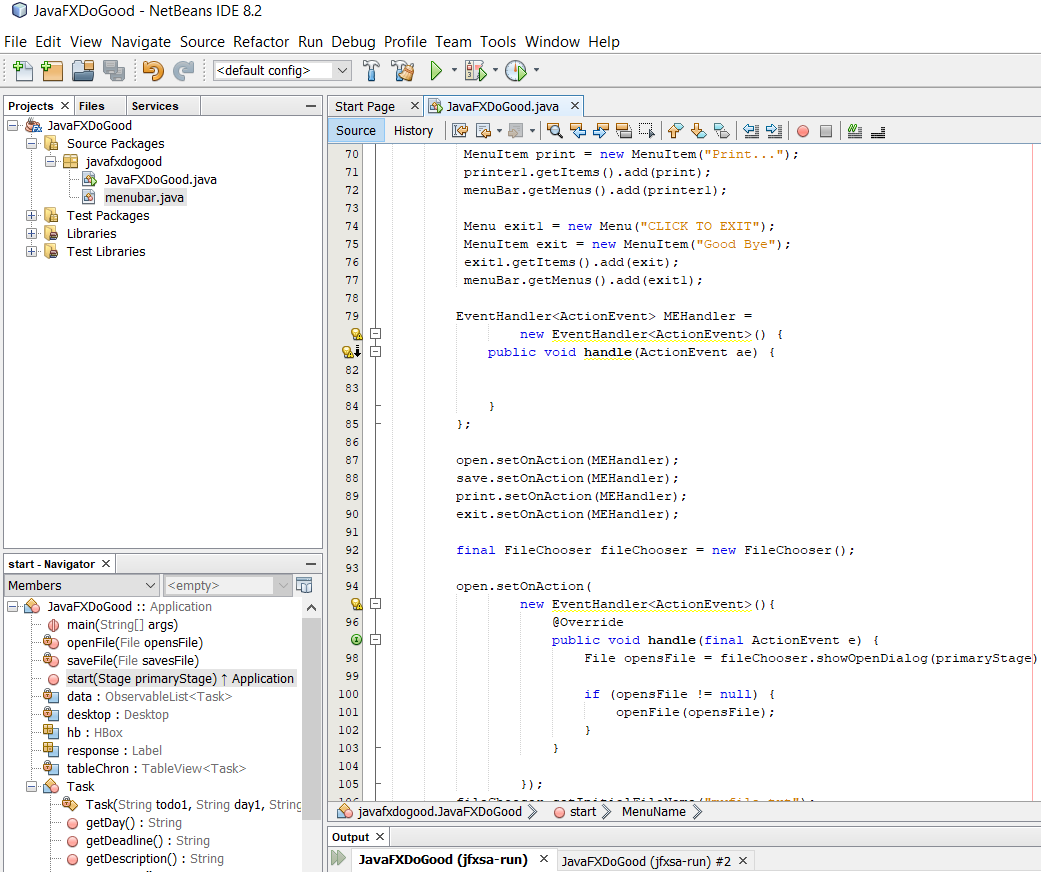
**EDITING DATA:**

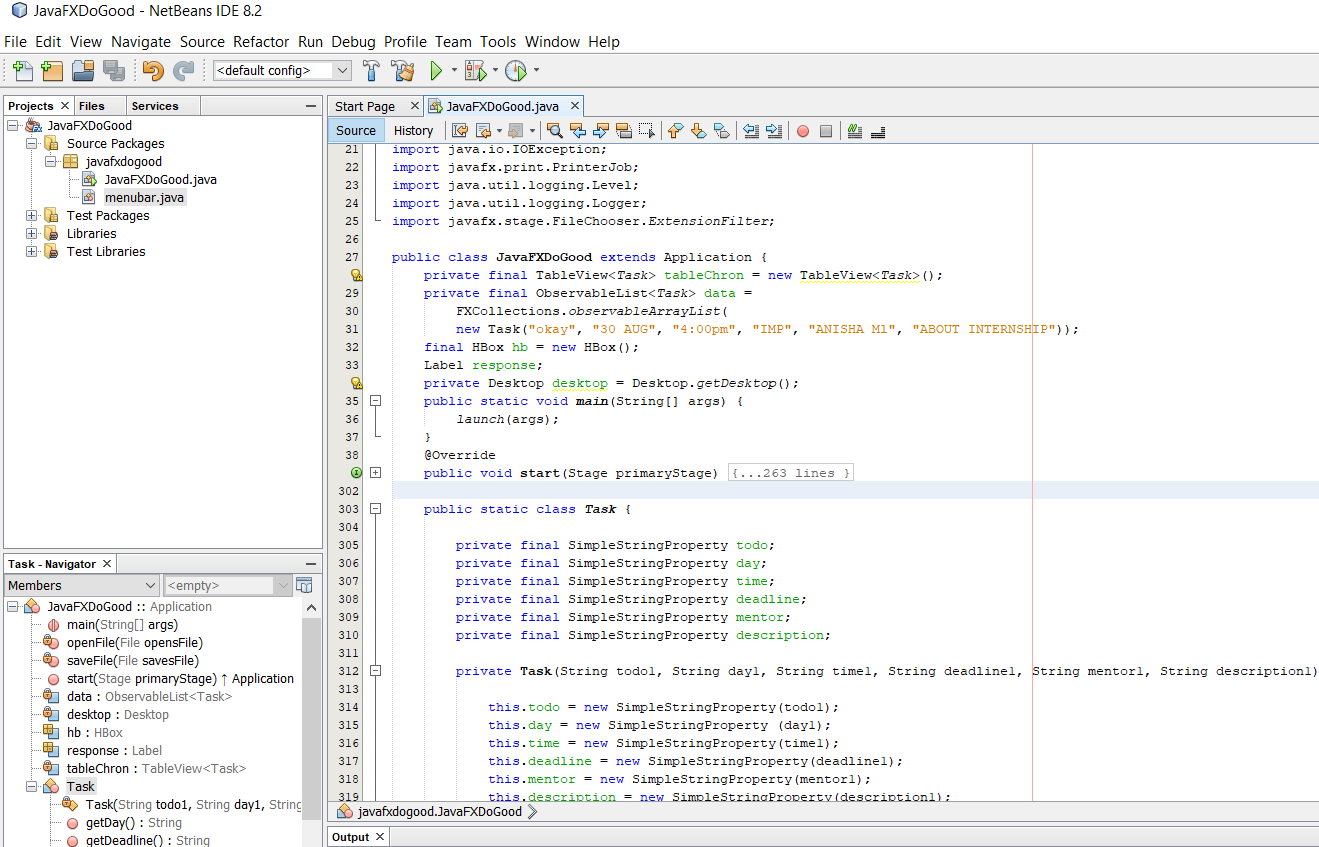
****

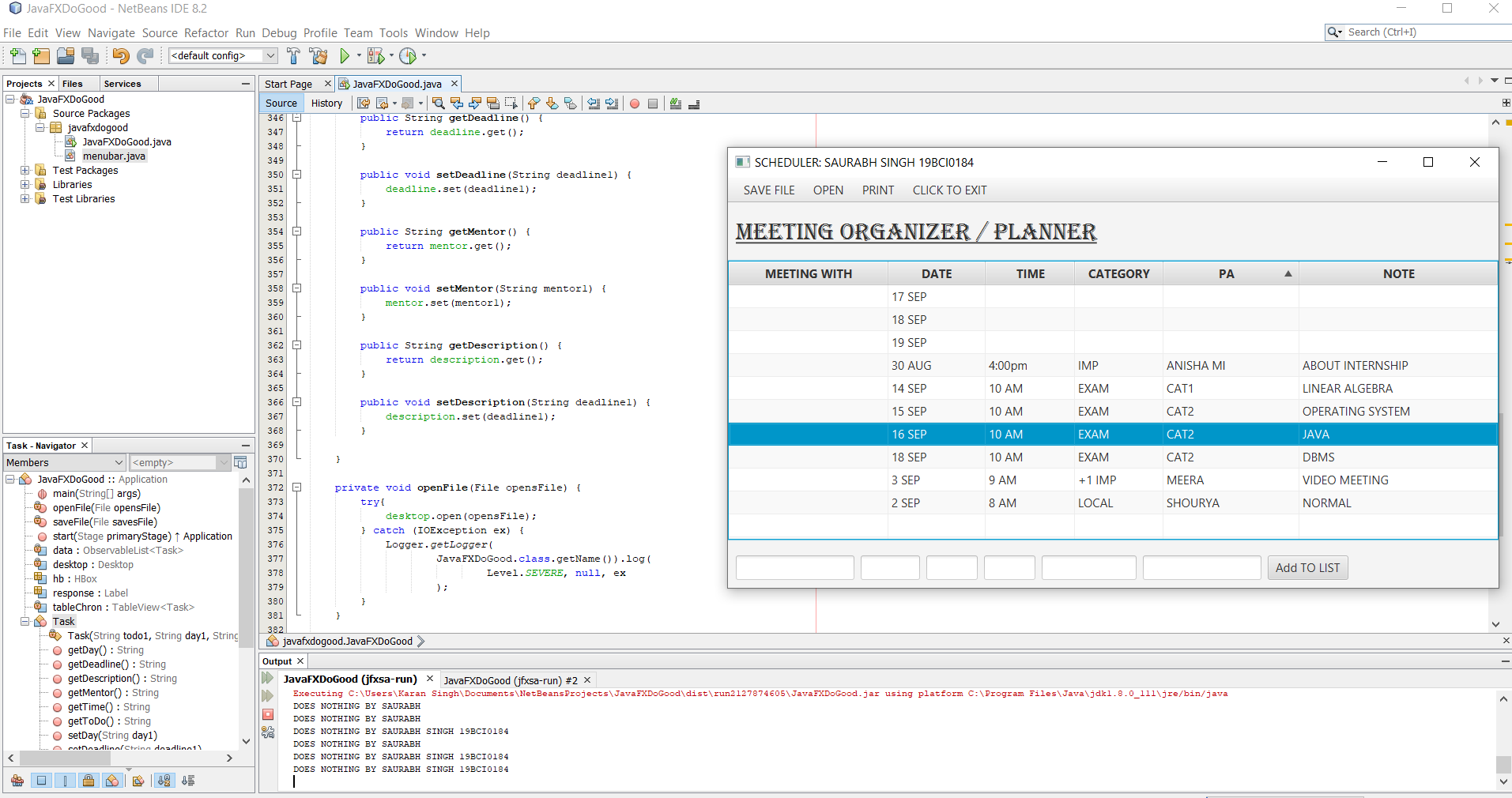
**ORGANIZE DATA:**

****

**SOME CODE SNIPPETS:**

****

****

****

**THANKYOU MAAM**

**SAURABH SINGH 19BCI0184**

I AM UPLOADING THE CODE AND FILES ON MY GITHUB I’D IN JAVAFX FOLDER TOO AND THE CODE IS ALSO GIVEN BELOW.

**GITHUB I’D-**  https://github.com/baazis

**CODE OF SCHEDULER GUI:-**

package javafxdogood;

import javafx.application.\*;

import javafx.beans.property.SimpleStringProperty;

import javafx.geometry.Insets;

import javafx.scene.layout.\*;

import javafx.scene.text.Font;

import javafx.scene.\*;

import javafx.scene.control.\*;

import javafx.stage.\*;

import javafx.scene.control.Menu;

import javafx.scene.control.MenuBar;

import javafx.scene.control.MenuItem;

import javafx.collections.ObservableList;

import javafx.collections.FXCollections;

import javafx.event.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.control.cell.TextFieldTableCell;

import java.awt.Desktop;

import java.io.File;

import java.io.IOException;

import javafx.print.PrinterJob;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.stage.FileChooser.ExtensionFilter;

public class JavaFXDoGood extends Application {

private final TableView<Task> tableChron = new TableView<Task>();

private final ObservableList<Task> data =

FXCollections.observableArrayList(

new Task("okay", "30 AUG", "4:00pm", "IMP", "ANISHA Ml", "ABOUT INTERNSHIP"));

final HBox hb = new HBox();

Label response;

private Desktop desktop = Desktop.getDesktop();

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) {

VBox vBox1 = new VBox();

Scene scene1 = new Scene(vBox1);

vBox1.setSpacing(10);

response = new Label("Menu");

MenuBar menuBar = new MenuBar();

Menu fileMenu = new Menu("SAVE FILE");

MenuItem save = new MenuItem("Save As");

MenuItem SAVE1 = new MenuItem("DO NOTHING");

SeparatorMenuItem separator = new SeparatorMenuItem();

SAVE1.setOnAction((e)-> {

System.out.println("DOES NOTHING BY SAURABH");

});

fileMenu.getItems().add(save);

fileMenu.getItems().add(SAVE1);

fileMenu.getItems().add(separator);

menuBar.getMenus().add(fileMenu);

Menu MenuName = new Menu("OPEN");

MenuItem open = new MenuItem("Open...");

MenuItem MenuItem1 = new MenuItem("DO NOTHING");

MenuItem1.setOnAction((e)-> {

System.out.println("DOES NOTHING BY SAURABH SINGH 19BCI0184");

});

MenuName.getItems().add(open);

MenuName.getItems().add(MenuItem1);

menuBar.getMenus().add(MenuName);

Menu printer1 = new Menu("PRINT");

MenuItem print = new MenuItem("Print...");

printer1.getItems().add(print);

menuBar.getMenus().add(printer1);

Menu exit1 = new Menu("CLICK TO EXIT");

MenuItem exit = new MenuItem("Good Bye");

exit1.getItems().add(exit);

menuBar.getMenus().add(exit1);

EventHandler<ActionEvent> MEHandler =

new EventHandler<ActionEvent>() {

public void handle(ActionEvent ae) {

}

};

open.setOnAction(MEHandler);

save.setOnAction(MEHandler);

print.setOnAction(MEHandler);

exit.setOnAction(MEHandler);

final FileChooser fileChooser = new FileChooser();

open.setOnAction(

new EventHandler<ActionEvent>(){

@Override

public void handle(final ActionEvent e) {

File opensFile = fileChooser.showOpenDialog(primaryStage);

if (opensFile != null) {

openFile(opensFile);

}

}

});

fileChooser.setInitialFileName("myfile.txt");

fileChooser.getExtensionFilters().addAll(new ExtensionFilter("All Files", "\*.\*"));

save.setOnAction(

new EventHandler<ActionEvent>(){

@Override

public void handle(ActionEvent e) {

File savesFile = fileChooser.showSaveDialog(primaryStage);

if (savesFile != null) {

saveFile(savesFile);

}

}

});

exit.setOnAction(new EventHandler<ActionEvent>() {

public void handle(ActionEvent event) {

Platform.exit();

}

});

print.setOnAction(

new EventHandler<ActionEvent>(){

@Override

public void handle(ActionEvent e) {

PrinterJob job = PrinterJob.createPrinterJob();

if (job != null) {

boolean showPrintDialog = job.showPrintDialog(primaryStage.getOwner());

if(showPrintDialog){

job.endJob();

}

}

}

});

final Label label = new Label("MEETING ORGANIZER");

label.setFont(new Font("ALGERIAN", 30));

label.setPadding(new Insets(10, 10, 10, 10));

label.setUnderline(true);

tableChron.setEditable(true);

TableColumn taskCol = new TableColumn("MEETING WITH");

TableColumn dayCol = new TableColumn("DATE");

TableColumn timeCol = new TableColumn("TIME");

TableColumn deadlineCol = new TableColumn("CATEGORY");

TableColumn mentorCol = new TableColumn("PA");

TableColumn descriptionCol = new TableColumn("NOTE");

taskCol.setCellValueFactory(

new PropertyValueFactory<Task, String>("Task"));

taskCol.setCellFactory(TextFieldTableCell.forTableColumn());

taskCol.setOnEditCommit(

new EventHandler<TableColumn.CellEditEvent<Task, String>>() {

@Override

public void handle(TableColumn.CellEditEvent<Task, String> t) {

((Task) t.getTableView().getItems().get(

t.getTablePosition().getRow())

).setToDo(t.getNewValue());

}

}

);

dayCol.setCellValueFactory(

new PropertyValueFactory<Task, String>("Day"));

dayCol.setCellFactory(TextFieldTableCell.forTableColumn());

dayCol.setOnEditCommit(

new EventHandler<TableColumn.CellEditEvent<Task, String>>() {

@Override

public void handle(TableColumn.CellEditEvent<Task, String> t) {

((Task) t.getTableView().getItems().get(

t.getTablePosition().getRow())

).setDay(t.getNewValue());

}

}

);

timeCol.setCellValueFactory(

new PropertyValueFactory<Task, String>("Time"));

timeCol.setCellFactory(TextFieldTableCell.forTableColumn());

timeCol.setOnEditCommit(

new EventHandler<TableColumn.CellEditEvent<Task, String>>() {

@Override

public void handle(TableColumn.CellEditEvent<Task, String> t) {

((Task) t.getTableView().getItems().get(

t.getTablePosition().getRow())

).setTime(t.getNewValue());

}

}

);

deadlineCol.setCellValueFactory(

new PropertyValueFactory<Task, String>("Deadline"));

deadlineCol.setCellFactory(TextFieldTableCell.forTableColumn());

deadlineCol.setOnEditCommit(

new EventHandler<TableColumn.CellEditEvent<Task, String>>() {

@Override

public void handle(TableColumn.CellEditEvent<Task, String> t) {

((Task) t.getTableView().getItems().get(

t.getTablePosition().getRow())

).setDeadline(t.getNewValue());

}

}

);

mentorCol.setCellValueFactory(

new PropertyValueFactory<Task, String>("Mentor"));

mentorCol.setCellFactory(TextFieldTableCell.forTableColumn());

mentorCol.setOnEditCommit(

new EventHandler<TableColumn.CellEditEvent<Task, String>>() {

@Override

public void handle(TableColumn.CellEditEvent<Task, String> t) {

((Task) t.getTableView().getItems().get(

t.getTablePosition().getRow())

).setMentor(t.getNewValue());

}

}

);

descriptionCol.setCellValueFactory(

new PropertyValueFactory<Task, String>("Description"));

descriptionCol.setCellFactory(TextFieldTableCell.forTableColumn());

descriptionCol.setOnEditCommit(

new EventHandler<TableColumn.CellEditEvent<Task, String>>() {

@Override

public void handle(TableColumn.CellEditEvent<Task, String> t) {

((Task) t.getTableView().getItems().get(

t.getTablePosition().getRow())

).setDescription(t.getNewValue());

}

}

);

tableChron.setItems(data);

tableChron.getColumns().addAll(taskCol, dayCol, timeCol, deadlineCol, mentorCol, descriptionCol);

tableChron.setColumnResizePolicy(TableView.CONSTRAINED\_RESIZE\_POLICY);

taskCol.setMinWidth(150);

dayCol.setMinWidth(70);

timeCol.setMinWidth(60);

deadlineCol.setMinWidth(60);

mentorCol.setMinWidth(120);

descriptionCol.setMinWidth(200);

HBox hbox1 = new HBox();

hbox1.setSpacing(8);

hbox1.setPadding(new Insets(10, 10, 10, 10));

TextField addToDo = new TextField();

TextField addDay = new TextField();

TextField addTime = new TextField();

TextField addDeadline = new TextField();

TextField addMentor = new TextField();

TextField addDescription = new TextField();

addToDo.setText("NAME");

addDay.setText("ENTER DATE");

addTime.setText("Time");

addDeadline.setText("IMP?");

addMentor.setText("ORGANIZER NAME");

addDescription.setText("Add specific descriptions");

addToDo.setPrefWidth(150);

addDay.setPrefWidth(75);

addTime.setPrefWidth(65);

addDeadline.setPrefWidth(65);

addMentor.setPrefWidth(120);

addDescription.setPrefWidth(150);

Button btnAdd = new Button("Add TO LIST");

btnAdd.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent e) {

data.add(new Task(

addToDo.getText(),

addDay.getText(),

addTime.getText(),

addDeadline.getText(),

addMentor.getText(),

addDescription.getText()));

addToDo.clear();

addDay.clear();

addTime.clear();

addDeadline.clear();

addMentor.clear();

addDescription.clear();

}

});

hbox1.getChildren().addAll(addToDo, addDay, addTime, addDeadline, addMentor, addDescription, btnAdd);

vBox1.getChildren().addAll(menuBar, label, tableChron, hbox1);

primaryStage.setWidth(750);

primaryStage.setHeight(500);

primaryStage.setTitle("SCHEDULER: SAURABH SINGH 19BCI0184");

primaryStage.setScene (scene1);

primaryStage.show();

}

public static class Task {

private final SimpleStringProperty todo;

private final SimpleStringProperty day;

private final SimpleStringProperty time;

private final SimpleStringProperty deadline;

private final SimpleStringProperty mentor;

private final SimpleStringProperty description;

private Task(String todo1, String day1, String time1, String deadline1, String mentor1, String description1){

this.todo = new SimpleStringProperty(todo1);

this.day = new SimpleStringProperty (day1);

this.time = new SimpleStringProperty(time1);

this.deadline = new SimpleStringProperty(deadline1);

this.mentor = new SimpleStringProperty(mentor1);

this.description = new SimpleStringProperty(description1);

}

public String getToDo() {

return todo.get();

}

public void setToDo(String todo1) {

todo.set(todo1);

}

public String getDay() {

return day.get();

}

public void setDay(String day1) {

day.set(day1);

}

public String getTime() {

return time.get();

}

public void setTime(String time1) {

time.set(time1); }

public String getDeadline() {

return deadline.get();

}

public void setDeadline(String deadline1) {

deadline.set(deadline1);

}

public String getMentor() {

return mentor.get();

}

public void setMentor(String mentor1) {

mentor.set(mentor1);

}

public String getDescription() {

return description.get();

}

public void setDescription(String deadline1) {

description.set(deadline1);

}

}

private void openFile(File opensFile) {

try{

desktop.open(opensFile);

} catch (IOException ex) {

Logger.getLogger(

JavaFXDoGood.class.getName()).log(

Level.SEVERE, null, ex

);

}

}

private void saveFile(File savesFile) {

try{

desktop.open(savesFile);

} catch (IOException ex) {

Logger.getLogger(

JavaFXDoGood.class.getName()).log(

Level.SEVERE, null, ex

);

}

}}

THANKYOU MAAM

SAURABH SINGH

19BCI0184