```
package Controllers;

import Models.DatabaseModel;
import Models.Employee;
import Views.AvailabilityView;
import Views.LoginView;

import java.sql.Date;

/**
 * Created by Kuba on 2015-11-28.
 */
public class AvailabilityController {
    DatabaseModel databaseModel= new
DatabaseModel();
    AvailabilityView availabilityView;



    public void goToAvailabilityView(){
        availabilityView= new
AvailabilityView();

availabilityView.createAvailabilityView();


    }

    public void addAvailability(Date dates,
String shiftType, int empId) {
```

```
databaseModel.addAvailability(dates,shiftTy
pe,empId);
    }

    public void removeAvailability(Date
dates, String shiftType, int empId) {

databaseModel.removeAvailability(dates,shif
tType,empId);
    }
    public boolean ifAvailable(Date dates,
String shiftType, int empId) {
        return
databaseModel.ifAvailable(dates,shiftType,e
mpId);
    }

}

package Controllers;

import Models.DatabaseModel;
import Views.CreateEmployeeView;

/**
 * Created by Kuba on 2015-12-01.
 */
public class CreateEmployeeController {
```

```
    CreateEmployeeView createEmployeeView=
new CreateEmployeeView();

    public void goToCreateEmployee(){

createEmployeeView.createEmployee();

    }

    public void
saveEmployeeToDatabase(boolean ifManager,
String fullName, String address, String
city, String nationality, String
mobileNumber, String postcode,

String login, String password, String cpr,
String picture){
        DatabaseModel databaseModel= new
DatabaseModel();

databaseModel.saveEmployeeInfoToDatabase(if
Manager,fullName,address,city,nationality,m
obileNumber,postcode,login,password,cpr,pic
ture);

    }
}


package Controllers;
```

```
import Models.DatabaseModel;
import Models.Employee;
import Views.EditProfileView;

/**
 * Created by Kuba on 2015-11-29.
 */
public class EditProfileController {

    EditProfileView editProfileView;

    public void goToEditProfileView(){
        editProfileView= new
EditProfileView();

editProfileView.createEditProfileView();

    }


    public Employee getEmployeeInfo(int
empID){
        DatabaseModel databaseModel= new
DatabaseModel();
        return
databaseModel.getEmployeeInfoFromDatabase(e
mpID);
    }

    public void
updateEmployeeInfoToDatabase(boolean
```

```
ifManager, String fullName, String address,
String city, String nationality, String
mobileNumber, String postcode,

String login, String password, String cpr,
String picture, int empId){
        DatabaseModel databaseModel= new
DatabaseModel();


databaseModel.updateEmployeeInfoToDatabase(
ifManager,  fullName,  address,  city,
nationality,  mobileNumber,  postcode,
                  login,  password,  cpr,
picture, empId);

    }
}


package Controllers;

import Models.DatabaseModel;
import Models.Employee;
import Views.LoginView;

/**
 * Created by Kuba on 2015-11-28.
 */
public class LoginController {
```

```
    LoginView loginView= new LoginView();


    public void goBackToLoginView(){
        loginView.createLoginView();
    }

    public Employee
getAccesFromWithEmployeeInfo(String login,
String password){
        DatabaseModel databaseModel= new
DatabaseModel();
        return
databaseModel.getAccesRetrieveEmployee(logi
n,password);


    }
    public Employee passLoggedEmployee(){
        return
loginView.passLoggedEmployee();

    }
```

```
}


package Controllers;

import Models.DatabaseModel;
import Models.Employee;
import Views.ManagerConfirmationView;

import java.util.ArrayList;

/**
 * Created by Kuba on 2015-12-08.
 */
public class ManagerConfirmationController
{



    ManagerConfirmationView
managerConfirmationView= new
ManagerConfirmationView();
    DatabaseModel databaseModel= new
DatabaseModel();

    public void
goToManagerConfirmationView(){

managerConfirmationView.availabilitiesView(
);
    }
```

```
    public ArrayList<Employee>
getEmployeesForShiftConfirmation(String
date, String shiftType){

        return
databaseModel.getEmployeesForShiftConfirmat
ion(date, shiftType);
    }

    public void updateConfirmedRow(int
availabilityId, boolean shiftType)
    {

databaseModel.updateConfirmedRow(availabili
tyId, shiftType);
    }


}


package Controllers;


import Models.ShiftTypes;

/**
 * Created by Bob on 12/7/2015.
 */
public class ShiftTypesController {
```

```
    public ShiftTypes[] getShiftTypes() {
        ShiftTypes[] shiftTypes = new
ShiftTypes[3];
         shiftTypes = ShiftTypes.values();

        return shiftTypes;
    }
}


package Controllers;

import Models.DatabaseModel;
import Models.Employee;
import Views.UltimateScheduleView;
import java.util.ArrayList;

/**
 * Created by Kuba on 2015-11-29.
 */
public class UltimateScheduleController {

    UltimateScheduleView
ultimateScheduleView;
    DatabaseModel databaseModel= new
DatabaseModel();

    public void goToUltimateScheduleView(){
        ultimateScheduleView= new
UltimateScheduleView();
```

```
ultimateScheduleView.createUltimateSchedule
View();


    }
    public ArrayList<Employee>
getEmployeesForUltimateSchedule(String
date, String shiftType, boolean confirmed){

        return
databaseModel.getEmployeesForUltimateSchedu
le(date,shiftType,confirmed);

    }
    public boolean ifYouAreBooked(String
date,String shiftType ,int empId, boolean
confirmed)
    {

        return
databaseModel.ifYouAreBooked(date,
shiftType, empId, confirmed);

    }


}


package Models;
```

```
import Views.LoginView;

import java.lang.Boolean;import
java.lang.String;import
java.lang.System;import java.sql.*;
import java.sql.Connection;import
java.sql.DriverManager;import
java.sql.PreparedStatement;import
java.sql.ResultSet;import
java.sql.SQLException;import
java.util.ArrayList;

/**
 * Created by Kuba on 2015-12-01.
 */
public class DatabaseModel {
    Connection connection= null;
    public DatabaseModel() {

        String DB_URL = "jdbc:mysql://
localhost/northmediadatabase";
        String USER = "root";
        String PASS = "root";
        try {
            connection =
DriverManager.getConnection(DB_URL,USER,PAS
S);
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
    }

    public Employee
getAccesRetrieveEmployee(String login,
String password){
        String sql= "SELECT * FROM
employees WHERE login= ? AND password= ?";
        Employee employee = null;
        try{
            PreparedStatement
preparedStatement =
connection.prepareStatement(sql);
            preparedStatement.setString(1,
login);
            preparedStatement.setString(2,
password);
            ResultSet resultSet =
preparedStatement.executeQuery();
            if (resultSet.next()) {
                int empId=
resultSet.getInt(1);
                Boolean ifManager =
resultSet.getBoolean(2);
                String fullName=
resultSet.getString(3);
                String address=
resultSet.getString(4);
                String city=
resultSet.getString(5);
                String nationality=
resultSet.getString(6);
```

```
                String mobileNumber=
resultSet.getString(7);
                String postcode=
resultSet.getString(8);
                String sLogin=
resultSet.getString(9);
                String sPassword=
resultSet.getString(10);
                String cpr=
resultSet.getString(11);
                String picture=
resultSet.getString(12);
                employee= new
Employee(empId,ifManager,fullName,address,c
ity,nationality,mobileNumber,postcode,sLogi
n,sPassword,cpr,picture);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }


        return employee;}



    public void
saveEmployeeInfoToDatabase(boolean
ifManager, String fullName, String address,
String city, String nationality, String
```

```
mobileNumber, String postcode,

String login, String password, String cpr,
String picture){
        String sql= "INSERT INTO employees
VALUES(null,?,?,?,?,?,?,?,?,?,?,?)";
        try {
        PreparedStatement
preparedStatement=
connection.prepareStatement(sql);
        preparedStatement.setBoolean(1,
ifManager);
            preparedStatement.setString(2,
fullName);
            preparedStatement.setString(3,
address);
            preparedStatement.setString(4,
city);
            preparedStatement.setString(5,
nationality);
            preparedStatement.setString(6,
mobileNumber);
            preparedStatement.setString(7,
postcode);
            preparedStatement.setString(8,
login);
            preparedStatement.setString(9,
password);
            preparedStatement.setString(10,
cpr);
            preparedStatement.setString(11,
```

```
picture);
            int numberOfRows=
preparedStatement.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }


    }

    public Employee
getEmployeeInfoFromDatabase(int employeeId)
{
        String sql= "SELECT * FROM
employees WHERE empId= ?";
        Employee employee= null;
        try {
            PreparedStatement
preparedStatement=
connection.prepareStatement(sql);
            preparedStatement.setInt(1,
employeeId);
            ResultSet resultSet =
preparedStatement.executeQuery();
            while (resultSet.next()){
                int empId=
resultSet.getInt(1);
                Boolean ifManager =
resultSet.getBoolean(2);
                String fullName=
```

```
resultSet.getString(3);
                String address=
resultSet.getString(4);
                String city=
resultSet.getString(5);
                String nationality=
resultSet.getString(6);
                String mobileNumber=
resultSet.getString(7);
                String postcode=
resultSet.getString(8);
                String login=
resultSet.getString(9);
                String password=
resultSet.getString(10);
                String cpr=
resultSet.getString(11);
                String picture=
resultSet.getString(12);
                employee= new
Employee(empId,ifManager,fullName,address,c
ity,nationality,mobileNumber,postcode,login
,password,cpr,picture);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return employee;
    }

    public void
```

```
updateEmployeeInfoToDatabase(boolean
ifManager, String fullName, String address,
String city, String nationality, String
mobileNumber, String postcode,

String login, String password, String cpr,
String picture, int empId){
        String sql= "UPDATE employees SET
ifManager=?, fullName=?, address=?,
city= ?, nationality=?, mobileNumber=?,
postcode=?, login=?, password=?, cpr=?,
empPicture=? " +
                "WHERE empId=?";
        try {
            PreparedStatement
preparedStatement=
connection.prepareStatement(sql);
            preparedStatement.setBoolean(1,
ifManager);
            preparedStatement.setString(2,
fullName);
            preparedStatement.setString(3,
address);
            preparedStatement.setString(4,
city);
            preparedStatement.setString(5,
nationality);
            preparedStatement.setString(6,
mobileNumber);
            preparedStatement.setString(7,
postcode);
```

```
            preparedStatement.setString(8,
login);
            preparedStatement.setString(9,
password);
            preparedStatement.setString(10,
cpr);
            preparedStatement.setString(11,
picture);

preparedStatement.setInt(12,empId);

preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }


    }




    public boolean ifYouAreBooked(String
date,String shiftType ,int empId, boolean
confirmed)
    {
        String sql = "SELECT * FROM
availabilities WHERE empId = ? AND
confirmed = ? AND date = ? AND shiftType
= ?";
        try {
```

```
            PreparedStatement
preparedStatement =
connection.prepareStatement(sql);
            preparedStatement.setInt(1,
empId);
            preparedStatement.setBoolean(2,
confirmed);
            preparedStatement.setString(3,
date);
            preparedStatement.setString(4,
shiftType);
            ResultSet resultSet =
preparedStatement.executeQuery();
            if (resultSet.next())
            {
                return true;
                //change the button
            }
        }catch (SQLException e)
        {
            System.out.println("Something
is bad");
        }return false;
    }
    public ArrayList<Employee>
getEmployeesForUltimateSchedule(String
date, String shiftType, boolean confirmed){
        ArrayList<Employee> employees= new
ArrayList<>();

        String sql= "SELECT
```

```
employees.empId, employees.fullname,
employees.mobileNumber, employees.postcode
FROM employees " +
                "JOIN availabilities " +
                "ON availabilities.empId =
employees.empId " +
                "WHERE
availabilities.date= ? AND
availabilities.shiftType= ? AND
availabilities.confirmed= ?";

        try {
            PreparedStatement
preparedStatement =
connection.prepareStatement(sql);
            preparedStatement.setString(1,
date);
            preparedStatement.setString(2,
shiftType);
            preparedStatement.setBoolean(3,
confirmed);

            ResultSet resultSet =
preparedStatement.executeQuery();

            while(resultSet.next()){
                String sFullName=
resultSet.getString("employees.fullname");
                String sMobileNumber=
resultSet.getString("employees.mobileNumber
");
```

```
                String sPostCode=
resultSet.getString("employees.postcode");

                employees.add(new
Employee(sFullName, sMobileNumber,
sPostCode));

            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return employees;
    }

    public void addAvailability(Date dates,
String shiftType, int empId) {
        // sql - INSERT INTO
`northmediadatabase`.`availabilities`
(`availabilityId`, `dates`, `shiftType`,
`empId`) VALUES ('3', '2015-12-08', 'E',
'1');

        String sql = "INSERT INTO
availabilities (date, shiftType, empId) " +
"VALUES (?,?,?)";
        try
        {
            PreparedStatement
preparedStatement =
connection.prepareStatement(sql);
```

```
            preparedStatement.setDate(1,
dates);
            preparedStatement.setString(2,
shiftType);
            preparedStatement.setInt(3,
empId);

preparedStatement.executeUpdate();

        } catch (SQLException e)
        {
            e.printStackTrace();
        }
    }

    public void removeAvailability(Date
dates, String shiftType, int empId) {

        String sql = "DELETE FROM
availabilities WHERE date = ? AND shiftType
= ? AND empId = ?";

        try
        {
            PreparedStatement
preparedStatement =
connection.prepareStatement(sql);
            preparedStatement.setDate(1,
dates);
            preparedStatement.setString(2,
shiftType);
```

```
            preparedStatement.setInt(3,
empId);
            preparedStatement.execute();


        } catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
    public void updateConfirmedRow(int
availabilityId, boolean shiftType)
    {//try this
        String shiftTypeVal;
        if (shiftType)// same as
(shiftType==true)
            shiftTypeVal="1";
        else
            shiftTypeVal="0";
        //ManagerConfirmationController
managerConfirmationController= new
ManagerConfirmationController();
        //ArrayList<Employee>
employeesForShiftConfirmation= new
ArrayList<>();
        //
employeesForShiftConfirmation=managerConfir
mationController.getEmployeesForShiftConfir
mation(date, shiftType);
        String sql66 = "UPDATE
availabilities SET confirmed = ? where
```

```
availabilityId=?";


        try{
            PreparedStatement
preparedStatement =
connection.prepareStatement(sql66);
            preparedStatement.setString(1,
shiftTypeVal);
            preparedStatement.setInt(2,
availabilityId);
            int numberOfRows =
preparedStatement.executeUpdate();
        }
        catch (SQLException e)
        {
            System.out.println("Wrong!");

        }
    }

    public ArrayList<Employee>
getEmployeesForShiftConfirmation(String
date, String shiftType){
        ArrayList<Employee> employees= new
ArrayList<>();

        String sql= "SELECT
availabilities.availabilityId,
employees.fullName, employees.mobileNumber,
employees.postcode,availabilities.confirmed
```

```
FROM employees " +
                "JOIN availabilities " +
                "ON availabilities.empId =
employees.empId " +
                "WHERE
availabilities.date= ? AND
availabilities.shiftType=?";

        try {
            PreparedStatement
preparedStatement =
connection.prepareStatement(sql);
            preparedStatement.setString(1,
date);

preparedStatement.setString(2,shiftType);
            ResultSet resultSet =
preparedStatement.executeQuery();
            while(resultSet.next()){
                int sAvailabilityId=
resultSet.getInt(1);/// we ll do that later
                String sFullName=
resultSet.getString(2);
                String sMobileNumber=
resultSet.getString(3);
                String sPostCode=
resultSet.getString(4);
                boolean sConfirmed=
resultSet.getBoolean(5);
                employees.add(new
Employee(sAvailabilityId,sFullName,sMobileN
```

```
umber,sPostCode, sConfirmed));

            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return employees;
    }




    public boolean ifAvailable(Date dates,
String shiftType, int empId) {
        boolean out = true;

        String sql = "SELECT confirmed FROM
availabilities WHERE date = ? AND shiftType
= ? AND empId = ?";
        try {
            PreparedStatement
preparedStatement =
connection.prepareStatement(sql);
            preparedStatement.setDate(1,
dates);
            preparedStatement.setString(2,
shiftType);
            preparedStatement.setInt(3,
empId);
```

```
            //
preparedStatement.executeQuery();
            ResultSet resultSet =
preparedStatement.executeQuery();


            if(resultSet.next() == false) {
                out = false;
            }
        } catch (SQLException e){
            e.printStackTrace();
        }catch (NullPointerException e){
            e.printStackTrace(); // if
employee is working - then the record is
not null, its present.
        }

        return out;
    }

}


package Models;

import java.awt.*;import
java.lang.Boolean;import java.lang.String;

/**
 * Created by Kuba on 2015-12-01.
 */
```

```
public class Employee {

    public boolean getIfManager() {
        return ifManager;
    }

    public void setIfManager(boolean
ifManager) {
        this.ifManager = ifManager;
    }

    public String getFullName() {
        return fullName;
    }

    public void setFullName(String
fullName) {
        this.fullName = fullName;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address)
{
        this.address = address;
    }

    public String getCity() {
        return city;
```

```
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getNationality() {
        return nationality;
    }

    public void setNationality(String
nationality) {
        this.nationality = nationality;
    }

    public String getMobileNumber() {
        return mobileNumber;
    }

    public void setMobileNumber(String
mobileNumber) {
        this.mobileNumber = mobileNumber;
    }

    public String getPostcode() {
        return postcode;
    }

    public void setPostcode(String
postcode) {
        this.postcode = postcode;
```

```
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String
password) {
        this.password = password;
    }

    public String getCpr() {
        return cpr;
    }

    public void setCpr(String cpr) {
        this.cpr = cpr;
    }

    public String getPicture() {
        return picture;
    }
```

```
    public void setPicture(String picture)
{
        this.picture = picture;
    }


    int empId;

    public int getEmpId() {
        return empId;
    }

    public void setEmpId(int empId) {
        this.empId = empId;
    }

    boolean ifManager; // later boolean
            String fullName;
    String address;
    String city;
    String nationality;
            String mobileNumber;
    String postcode;
    String login;
    String password;
    String cpr;
    String picture;
    int availabilityId;

    public boolean isIfManager() {
        return ifManager;
```

```
    }

    public int getAvailabilityId() {
        return availabilityId;
    }

    public void setAvailabilityId(int
availabilityId) {
        this.availabilityId =
availabilityId;
    }

    public Boolean getAvailability() {
        return availability;
    }

    public void setAvailability(Boolean
availability) {
        this.availability = availability;
    }

    Boolean availability;



    public Employee(int empId, boolean
ifManager, String fullName, String address,
String city, String nationality, String
mobileNumber, String postcode, String
login, String password, String cpr, String
picture) {
```

```
        this.empId=empId;
        this.ifManager = ifManager;
        this.fullName = fullName;
        this.address = address;
        this.city = city;
        this.nationality = nationality;
        this.mobileNumber = mobileNumber;
        this.postcode = postcode;
        this.login = login;
        this.password = password;
        this.cpr = cpr;
        this.picture = picture;

    }

    public Employee(String fullName, String
mobileNumber, String postcode) {
        this.fullName = fullName;
        this.mobileNumber = mobileNumber;
        this.postcode = postcode;
    }

    public Employee(String fullName, String
mobileNumber, String postcode,boolean
availability) {
        this.fullName = fullName;
        this.mobileNumber = mobileNumber;
        this.postcode = postcode;
        this.availability= availability;
    }
    /*public Employee(int empId,String
```

```
fullName, String mobileNumber, String
postcode,boolean availability) {
        this.empId=empId;
        this.fullName = fullName;
        this.mobileNumber = mobileNumber;
        this.postcode = postcode;
        this.availability= availability;
    }*/

    public Employee(int
availabilityId,String fullName, String
mobileNumber, String postcode,boolean
availability) {
        this.availabilityId =
availabilityId;
        this.fullName = fullName;
        this.mobileNumber = mobileNumber;
        this.postcode = postcode;
        this.availability= availability;
    }
}


package Models;

/**
 * Created by Bob on 12/7/2015.
 */
public enum ShiftTypes {
    N, D, E;
}
```

```
package Views;

import Controllers.*;
import Models.ShiftTypes;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.geometry.VPos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.*;
import javafx.stage.Stage;

import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;

/**
 * Created by Grzegorz on 2015-11-28.
 */
public class AvailabilityView {
    AvailabilityController
availabilityController= new
AvailabilityController();
    EditProfileController
editProfileController= new
EditProfileController();
    LoginController loginController= new
```

```
LoginController();
    public void hideWindow(){
        availabilityViewStage.hide();
    }


    Scene availabilityViewScene;
    Stage availabilityViewStage;


    /**
     * Gregory - availability view
     */
    ShiftTypesController
shiftTypesController = new
ShiftTypesController();

    Calendar cal;
    SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy-MM-dd");
    ArrayList<Label> labels= new
ArrayList<>();
    ArrayList<Button> buttons = new
ArrayList<>();

    HBox hBox;
    Label weekLabel;
    Button incrementByOneButton;
    Button decrementByOneButton;
    int[] weekDays = {2,3,4,5,6,7,1}; //
because Sunday has number 1, we swap
```

```
Monday(2) on that place – project
requirements


    Button editEmployee= new Button("Edit
employee");

    GridPane gridPane;
    BorderPane borderPane;


    int weekOfYear;

    public void createAvailabilityView(){

        cal = Calendar.getInstance();  //
getting the current date

cal.setFirstDayOfWeek(Calendar.MONDAY); //
setting the Monday to be the first day of
the week instead of Sunday
        weekOfYear =
cal.get(Calendar.WEEK_OF_YEAR); // getting
the current week number

        gridPane = new GridPane(); //
creating a grid pane for the calendar and
all the functional buttons
        gridPane.setVgap(8); // setting
spacing between grid rows
        gridPane.setHgap(8);// setting
```

```
spacing between grid columns
        gridPane.setPadding(new
Insets(10,10,10,10));

        hBox = new HBox();
        decrementByOneButton = new
Button("<");
        incrementByOneButton = new
Button(">");
        weekLabel = new Label("WEEK: " +
weekOfYear);
        Label ifLoggedInLabel = new
Label();

        /**
         * Logged as
         * */

        String loginCredit =
loginController.passLoggedEmployee().getFul
lName();
        ifLoggedInLabel.setText("Welcome: "
+ loginCredit);

gridPane.setConstraints(ifLoggedInLabel, 0,
0, 3, 2, HPos.LEFT, VPos.TOP);

gridPane.getChildren().add(ifLoggedInLabel)
;

        HBox switchWeeks= new HBox();
```

```
switchWeeks.getChildren().addAll(decrementB
yOneButton, weekLabel,
incrementByOneButton);

gridPane.setConstraints(switchWeeks, 3, 1,
3, 2, HPos.LEFT, VPos.CENTER);


gridPane.getChildren().add(switchWeeks);

decrementByOneButton.setMaxWidth(Double.MAX
_VALUE);

incrementByOneButton.setMaxWidth(Double.MAX
_VALUE);


decrementByOneButton.setAlignment(Pos.CENTE
R_RIGHT);


incrementByOneButton.setOnAction(event1 ->
{
            incrementByOneButtonAction();
        });


decrementByOneButton.setOnAction(event1 ->
{
            decrementByOneButtonAction();
```

```
        });


        editEmployee.setOnAction(event -> {

editProfileController.goToEditProfileView()
;
        });
        Button ultimateViewButton= new
Button("Ultimate View");

ultimateViewButton.setOnAction(event1 -> {
            hideWindow();
            UltimateScheduleController
ultimateScheduleController = new
UltimateScheduleController();

ultimateScheduleController.goToUltimateSche
duleView();

        });



        Button logOutButton = new
Button("Log out");
        logOutButton.setOnAction(event1 ->
{
            loginController = new
LoginController();
            hideWindow();
```

```
loginController.goBackToLoginView();

        });

gridPane.getChildren().addAll(logOutButton,
ultimateViewButton, editEmployee);



gridPane.setConstraints(ultimateViewButton,
7,8,3,1, HPos.LEFT, VPos.TOP);

gridPane.setConstraints(logOutButton, 0,
8,2,1, HPos.CENTER, VPos.TOP);

gridPane.setConstraints(editEmployee, 3, 8,
2, 1, HPos.LEFT, VPos.TOP);


        createLabels();  // CREATING LABELS
AND SETTING THEM TO GRID AT ROW 1 POSITIONS
1-7
        createButtons();



gridPane.getColumnConstraints().add(new
ColumnConstraints(240));

gridPane.getColumnConstraints().add(new
```

```
ColumnConstraints(56));

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));


gridPane.getRowConstraints().addAll(new
RowConstraints(90));

gridPane.getRowConstraints().addAll(new
RowConstraints(30));

gridPane.getRowConstraints().addAll(new
RowConstraints(10));

gridPane.getRowConstraints().addAll(new
```

```
RowConstraints(40));

gridPane.getRowConstraints().addAll(new
RowConstraints(63));

gridPane.getRowConstraints().addAll(new
RowConstraints(63));

gridPane.getRowConstraints().addAll(new
RowConstraints(63));

gridPane.getRowConstraints().addAll(new
RowConstraints(40));

        borderPane = new BorderPane();
        borderPane.setTop(hBox);
        borderPane.setCenter(gridPane);
        availabilityViewStage = new
Stage();
        availabilityViewScene = new
Scene(borderPane, 878, 600);



availabilityViewScene.getStylesheets().add(
"backGround.css");
        borderPane.setId("background");


availabilityViewStage.setScene(availability
ViewScene);
```

```
        availabilityViewStage.show();


    }




    /**
     * Buttons methods
     */

    public void  createButtons() {  //
probably for model
        ShiftTypes[] shiftTypes=
shiftTypesController.getShiftTypes(); //
getting values from Enum of ShiftTypes (via
Controller)

        for(int j=0; j<3; j++) {
            for(int i=0; i<7; i++) {
                String dates =
labels.get(i).getText();
                String id = dates +
shiftTypes[j]; // creating buttons Id
                Button button = new
Button();

button.setMaxWidth(Double.MAX_VALUE);

button.setMaxHeight(Double.MAX_VALUE);
```

```
                button.setId(id);
                Date sqlDate =
Date.valueOf(dates); // asigning a date of
sql type Date
                String shiftType =
shiftTypes[j].toString();
                boolean booked =
availabilityController.ifAvailable(sqlDate,
shiftTypes[j].toString(),
loginController.passLoggedEmployee().getEmp
Id());

                if(booked == true) {

                    button.setStyle("-fx-
base: #00b300"); // green


                }

                button.setOnAction(event ->
{
                    // check
if(ifAvailable() != true) -> button is
gray(person haven't clicked 'available for
work' yet
                    if
(availabilityController.ifAvailable(sqlDate
, shiftType,
loginController.passLoggedEmployee().getEmp
Id()) != true) { // then add availability
```

```
                        button.setStyle("-
fx-base: #00b300"); // green


availabilityController.addAvailability(sqlD
ate, shiftType,
loginController.passLoggedEmployee().getEmp
Id()); // adding entry to availabilities
table

                    } else {

availabilityController.removeAvailability(s
qlDate, shiftType,
loginController.passLoggedEmployee().getEmp
Id()); // remove entry
                        button.setStyle("-
fx-base: #e3e3e3"); // gray

                    }
                    // send an update
request for the particular button

                });
                buttons.add(button);

gridPane.getChildren().add(button);

gridPane.setConstraints(button, i + 1, j +
4);
            }
```

```
        }
    }



    public void
incrementByOneButtonAction() { //main motor
to changing the properties of the buttons
of the calendar x shift
        if(weekOfYear < 52) {  // checking
if the week is smaller than 52
            weekOfYear = weekOfYear + 1;//
incrementing the week of the year
            updateDateLabels(labels); //
updating the date labels
            updateButtons();// updating the
button information


        }
    }
    public void
decrementByOneButtonAction() {//main motor
to changing the properties of the buttons
of the calendar x shift
        if(weekOfYear > 1) {// checking if
the week is smaller than 52
            weekOfYear = weekOfYear - 1; //
decrementing the week of the year
            updateDateLabels(labels);//
```

```
updating the date labels
            updateButtons();// updating the
button information


        }
    }

    /**
     *
     * Labels methods
     *
     */

    public void updateButtons() {

        String[] shiftTypes= {"N","D","E"};
        int indexButton = 0;

        for(int j=0; j<3; j++) {
            for (int i = 0; i < 7; i++) {

                String id =
labels.get(i).getText() + shiftTypes[j];
                Button button =
buttons.get(indexButton);

                String date =
id.substring(0,10);
                String shiftType =
id.substring(10, 11);
```

```
                Date sqlDate =
Date.valueOf(date);
                boolean booked =
availabilityController.ifAvailable(sqlDate,
shiftType,
loginController.passLoggedEmployee().getEmp
Id());
                if(booked == true) {

                    button.setStyle("-fx-
base: #00b300"); // green



                }
                else{
                    button.setStyle("-fx-
base: #e3e3e3"); // gray
                }
                //  button.setStyle("-fx-
font-size: 20px");
                button.setId(id);
                button.setOnAction(event ->
{



                    // check
if(ifAvailable() != true) -> button is
gray(person haven't clicked 'available for
work' yet

if(availabilityController.ifAvailable(sqlDa
```

```
te, shiftType,
loginController.passLoggedEmployee().getEmp
Id() ) != true) { // then add availability
                            button.setStyle("-
fx-base: #00b300"); // green


availabilityController.addAvailability(sqlD
ate, shiftType,
loginController.passLoggedEmployee().getEmp
Id()); // adding entry to availabilities
table

                    }
                    else {

availabilityController.removeAvailability(s
qlDate, shiftType,
loginController.passLoggedEmployee().getEmp
Id()); // remove entry
                            button.setStyle("-
fx-base: #e3e3e3"); // gray

                    }




                    //
updateCustomTable(date, shiftType);
                });
```

```
                indexButton++;
            }
        }
    }


    public void createLabels (){


        for(int i =0; i<7; i++) {    // 7 times we create a label, from Monday label to Sunday label

            cal.set(Calendar.WEEK_OF_YEAR, cal.get(Calendar.WEEK_OF_YEAR)); // setting a calendar to a desired week
            cal.set(Calendar.DAY_OF_WEEK, weekDays[i]);

            String date= simpleDateFormat.format(cal.getTime());// labels Id format (date)
            Label label = new Label(); // creating a label for a date
            label.setText(date);// setting date to label
            label.setStyle("-fx-font-size: 8px");// setting size of the label
            labels.add(label);// adding labels to array list of labels
```

```
gridPane.getChildren().add(label); // put
label to grid
            gridPane.setConstraints(label,
i + 1, 3); // position label at column
1...7, row 3
        }


    }

    public void updateDateLabels
(ArrayList<Label> labels) {
        weekLabel.setText("WEEK: " +
weekOfYear);  // setting the week label to
a week of interest
        cal = Calendar.getInstance(); //
getting the current date

cal.setFirstDayOfWeek(Calendar.MONDAY);//
setting the Monday to be the first day of
the week instead of Sunday


        for (int i = 0; i < labels.size();
i++) {
            cal.set(Calendar.WEEK_OF_YEAR,
weekOfYear);
            cal.set(Calendar.DAY_OF_WEEK,
weekDays[i]);
```

```
            String date =
simpleDateFormat.format(cal.getTime()); //
setting a string for a desired date
            labels.get(i).setText(date); //
setting the name of the label to desired
date


        }
    }




}

package Views;

import
Controllers.CreateEmployeeController;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
```

```
/**
 * Created by Kuba on 2015-12-01.
 */
public class CreateEmployeeView {
    VBox createEmployeeBox= new VBox();
    TextField fullNameTextField = new
TextField();
    Label fullNameLabel= new Label("Full
name");
    HBox fullNameHbox= new HBox();
    TextField cityTextField= new
TextField();
    Label cityLabel= new Label("City");
    HBox cityHbox= new HBox();
    TextField mobileTextField= new
TextField();
    Label mobileLabel= new Label("Mobile");
    HBox mobileHbox= new HBox();
    TextField addressTextField = new
TextField();
    Label addressLabel= new
Label("Address");
    HBox addressHbox= new HBox();
    TextField postCodeTextField= new
TextField();
    Label postCodeLabe= new Label("Post
code");
    HBox postCodeHbox= new HBox();
    TextField nationTextField= new
TextField();
```

```
    Label nationLabel= new Label("Nation");
    HBox nationHbox= new HBox();
    TextField loginTextField= new
TextField();
    Label loginLabel= new Label("Login");
    HBox loginHBox= new HBox();
    PasswordField passwordTextField= new
PasswordField();
    Label passwordLabel= new
Label("Password");
    HBox passwordHbox= new HBox();
    TextField cprTextField= new
TextField();
    Label cprLabel= new Label("Cpr");
    HBox cprHbox= new HBox();
    CheckBox ifManagerCheckBox= new
CheckBox();
    Label ifManagerLabel= new
Label("Ticked= Manager/ Unticked=
Employee");
    HBox ifManagerHbox= new HBox();
    ImageView pictureImageView= new
ImageView();
    Label pictureLabel= new Label();
    HBox pictureHbox= new HBox();
    Button createEmployeeButton= new
Button("Create account");
    Button cancelButton= new
Button("Cancel");
    HBox confirmCancelBox= new HBox();
    Scene createEmployeeScene= new
```

```
Scene(createEmployeeBox, 300, 400);
    Stage createEmployeeStage= new Stage();
    CreateEmployeeController
createEmployeeController;

    public void createEmployee(){

        functionsOfCreateEmp();
    }

    public void functionsOfCreateEmp() {
        createEmployeeController = new
CreateEmployeeController();

createEmployeeButton.setOnAction(event1 ->
{
            // again problem with many
errors non stop
            Boolean ifManager =
ifManagerCheckBox.selectedProperty().get();
            String fullName =
fullNameTextField.getText();
            String city =
cityTextField.getText();
            String mobile =
mobileTextField.getText();
            String address =
addressTextField.getText();
            String postcode =
postCodeTextField.getText();
            String nation =
```

```
nationTextField.getText();
            String login =
loginTextField.getText();
            String password =
passwordTextField.getText();
            String cpr =
cprTextField.getText();
            query(ifManager, fullName,
city, mobile, address, postcode, nation,
login, password, cpr);
            hideWindow();

        });

        cancelButton.setOnAction(event1 ->
{
            hideWindow();
        });


fullNameHbox.getChildren().addAll(fullNameT
extField, fullNameLabel);

cityHbox.getChildren().addAll(cityTextField
,cityLabel);

mobileHbox.getChildren().addAll(mobileTextF
ield,mobileLabel);

addressHbox.getChildren().addAll(addressTex
tField,addressLabel);
```

```
postCodeHbox.getChildren().addAll(postCodeT
extField, postCodeLabe);

nationHbox.getChildren().addAll(nationTextF
ield, nationLabel);

ifManagerHbox.getChildren().addAll(ifManage
rCheckBox,ifManagerLabel);

loginHBox.getChildren().addAll(loginTextFie
ld,loginLabel);

passwordHbox.getChildren().addAll(passwordT
extField,passwordLabel);

cprHbox.getChildren().addAll(cprTextField,
cprLabel);

confirmCancelBox.getChildren().addAll(creat
eEmployeeButton, cancelButton);

        String url= "picture.jpg";



        pictureImageView= new ImageView(new
Image(String.valueOf(getClass().getResource
(url))));

pictureImageView.setOnMouseClicked(event ->
```

```
{
            FileChooser fileChooser = new
FileChooser();
            fileChooser.setTitle("Open
Resource File");
            FileChooser.ExtensionFilter
extFilter = new
FileChooser.ExtensionFilter("Image Files",
"*.jpg", "*.jpeg");

fileChooser.getExtensionFilters().add(extFi
lter);
            File file =
fileChooser.showOpenDialog(createEmployeeSt
age.getScene().getWindow());

            if (file != null) {
                String fileDi =
file.getAbsolutePath();
                //Image newImage = new
Image(fileDi);
                System.out.println(fileDi);

            }
        });

pictureHbox.getChildren().addAll(pictureIma
geView, pictureLabel);


createEmployeeBox.getChildren().addAll(full
```

```
NameHbox,cityHbox,mobileHbox,addressHbox,po
stCodeHbox,nationHbox, loginHBox,
passwordHbox, cprHbox, ifManagerHbox,
pictureHbox,confirmCancelBox);
        createEmployeeBox.setStyle("-fx-
background-color: #d3e5e7");

createEmployeeStage.setScene(createEmployee
Scene);

        createEmployeeStage.show();

    }

    public void query(Boolean ifManager,
String fullName, String city, String
mobile, String address, String postcode,
String nation, String login, String
password, String cpr) { //

createEmployeeController.saveEmployeeToData
base(ifManager,fullName,address,
city,nation,mobile,postcode,login,
password,cpr,null);
    }

    public void hideWindow(){
        createEmployeeStage.hide();
    }
}
```

```
package Views;

import Controllers.AvailabilityController;
import
Controllers.CreateEmployeeController;
import Controllers.EditProfileController;
import Controllers.LoginController;
import Models.Employee;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

/**
 * Created by Kuba on 2015-11-29.
 */
public class EditProfileView extends
CreateEmployeeView{
    LoginController loginController = new
LoginController();
    EditProfileController
editProfileController= new
EditProfileController();



    public void createEditProfileView(){
        createEmployeeButton.setText("Save
changes");
    createEmployee();
```

```
    }

    public void createEmployee(){
        EditProfileController
editProfileController= new
EditProfileController();
        Employee employeeInfo =
editProfileController.getEmployeeInfo(login
Controller.passLoggedEmployee().getEmpId())
;

fullNameTextField.setText(employeeInfo.getF
ullName());

cityTextField.setText(employeeInfo.getCity(
));

mobileTextField.setText(employeeInfo.getMob
ileNumber());

addressTextField.setText(employeeInfo.getAd
dress());

postCodeTextField.setText(employeeInfo.getP
ostcode());

nationTextField.setText(employeeInfo.getNat
ionality());
```

```
loginTextField.setText(employeeInfo.getLogi
n());

passwordTextField.setText(employeeInfo.getP
assword());

cprTextField.setText(employeeInfo.getCpr())
;

ifManagerCheckBox.setSelected(employeeInfo.
getIfManager());

        functionsOfCreateEmp();
    }

    public void query(Boolean ifManager,
String fullName, String city, String
mobile, String address, String postcode,
String nation, String login, String
password, String cpr) {

editProfileController.updateEmployeeInfoToD
atabase(ifManager, fullName, address, city,
nation, mobile, postcode, login, password,
cpr, null,
loginController.passLoggedEmployee().getEmp
Id());
    }

}
```

```
package Views;
// created by Jakub
import Controllers.AvailabilityController;
import Controllers.LoginController;
import
Controllers.ManagerConfirmationController;
import Models.Employee;
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class LoginView extends Application{
    AvailabilityController
availabilityController= new
AvailabilityController();
    ManagerConfirmationController
managerConfirmationController= new
ManagerConfirmationController();
    LoginController loginController;
    BorderPane border;
    TextField loginTextField;
    PasswordField passwordTextField;
    Button loginButton;
    Scene loginScene;
```

```
    Stage loginStage;
    public static Employee employee;

    public static void main(String[] args)
{
        launch(args);
    }


    @Override
    public void start(Stage primaryStage)
throws Exception {

        createLoginView();
    }

    public void createLoginView() {
        border = new BorderPane();
        border.setId("loginBackGround");



        loginTextField= new TextField();
        loginTextField.setText("emp"); //
so we dont have to write it every time

loginTextField.setPromptText("Login");
        loginTextField.setId("login");
        passwordTextField= new
PasswordField();
        passwordTextField.setId("login");
```

```
        passwordTextField.setText("1");//
so we dont have to write it every time

passwordTextField.setPromptText("Password")
;

        loginButton= new Button("Login");
        loginButton.setOnAction(event -> {
            loginController = new
LoginController(); // when i put it in
global scope it runs over and over again
            employee =
loginController.getAccesFromWithEmployeeInf
o(loginTextField.getText(),
passwordTextField.getText());
            if ((employee != null)) {
                hideWindow();
                if
(employee.getIfManager()) {

managerConfirmationController.goToManagerCo
nfirmationView();
                } else {

availabilityController.goToAvailabilityView
();
                }
            }
        });

        VBox centerVBox = new VBox();
```

```
centerVBox.getChildren().addAll(loginTextFi
eld, passwordTextField, loginButton);

centerVBox.setId("loginBackGround");

centerVBox.setAlignment(Pos.CENTER);



        //    border.setMargin(centerVBox,
new Insets(300,0,0,300));

        loginScene= new Scene(centerVBox,
640, 480);

loginScene.getStylesheets().addAll(this.get
Class().getResource("loginStyle.css").toExt
ernalForm());
        //
loginScene.getStylesheets().add(getClass().
getResource("/
loginStyle.css").toExternalForm());

        loginStage= new Stage();
        loginStage.setResizable(false);
        loginStage.setScene(loginScene);
        loginStage.show();
    }
    public void hideWindow(){
```

```
        loginStage.hide();
    }

    public Employee passLoggedEmployee(){
        return employee;

    }
}


package Views;

import
Controllers.CreateEmployeeController;
import Controllers.LoginController;
import
Controllers.ManagerConfirmationController;
import
Controllers.UltimateScheduleController;
import Models.Employee;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.geometry.VPos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
```

```
/**
 * Created by Edgaras on 12/1/2015.
 */

public class ManagerConfirmationView{
    LoginController loginController;
    SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy-MM-dd");
    Calendar cal;
    ArrayList<Label> labels= new
ArrayList<>();
    ArrayList<Button> buttons = new
ArrayList<>();
    UltimateScheduleController
ultimateScheduleController = new
UltimateScheduleController();
    String[] shiftTypes= {"N","D","E"};
    int[] weekDays = {2,3,4,5,6,7,1}; //
because Sunday has number 1, we swap
Monday(2) on that place - project
requirements


    Label weekLabel;
    Button incrementByOneButton;
    Button decrementByOneButton;
    Button logOutButton;
    Button createEmployeeButton;
```

```
    GridPane gridPane;
    GridPane customTableGridPane= new
GridPane();

    BorderPane borderPane= new
BorderPane();
    Scene scene;
    Stage stage;

    int weekOfYear;


    public VBox createCustomTable() {

        VBox customTableView = new
VBox(); // VBox containing Hbox with custom
table column names and GridPane with
        // all the entries

        HBox customTableHbox= new HBox();//
HBox containing all the column names for
the custom table
        Label customTableLabel= new
Label("Employees         ");//
        Label customTableLabel1 = new
Label("Phone   ");//
        Label customTableLable2 = new
Label("Post");// creating column labels
        Label customTableLable3 = new
Label("Y/N");//
```

```
        customTableHbox.setPadding(new
Insets(186,42,30,0));// Setting padding so
the title column Hbox is
        // exactly matching our background
picture (same with grid pane of entries)

        customTableHbox.setSpacing(10);  //
setting spacing between column labels

customTableHbox.getChildren().addAll(custom
TableLabel, customTableLabel1,
customTableLable2,customTableLable3);



customTableView.getChildren().addAll(custom
TableHbox, customTableGridPane);// putting
the column titles
        //and entries GridPane in a
matching


        return customTableView;
    }



    public void updateCustomTable(String
date, String shiftType){
```

```
customTableGridPane.getChildren().clear();
        ManagerConfirmationController
managerConfirmationController= new
ManagerConfirmationController();
        ArrayList<Employee>
employeesForShiftConfirmation= new
ArrayList<>();

employeesForShiftConfirmation=managerConfir
mationController.getEmployeesForShiftConfir
mation(date,shiftType);

        for (int i=0;
i<employeesForShiftConfirmation.size(); i+
+){
            Label employeesNameLabel= new
Label(employeesForShiftConfirmation.get(i).
getFullName());
            employeesNameLabel.setStyle("-
fx-font-size: 10px");
            Label employeesPhoneLabel = new
Label(employeesForShiftConfirmation.get(i).
getMobileNumber());
            employeesPhoneLabel.setStyle("-
fx-font-size: 10px");
            Label employeesPostLabel = new
Label(employeesForShiftConfirmation.get(i).
getPostcode());
            employeesPostLabel.setStyle("-
fx-font-size: 10px");
```

```
            CheckBox cb= new CheckBox();

cb.setId(String.valueOf(employeesForShiftCo
nfirmation.get(i).getAvailabilityId()));


cb.setSelected(employeesForShiftConfirmatio
n.get(i).getAvailability());
            cb.setOnAction(event -> {
                int avalID =
Integer.parseInt(cb.getId());
                boolean shiftTypeTemp =
cb.isSelected();

managerConfirmationController.updateConfirm
edRow(avalID, shiftTypeTemp);


            });

customTableGridPane.setConstraints(cb, 3,
i);

customTableGridPane.setConstraints(employee
sNameLabel,0,i);

customTableGridPane.setConstraints(employee
sPhoneLabel,1, i);

customTableGridPane.setConstraints(employee
sPostLabel, 2, i);
```

```
customTableGridPane.getChildren().addAll(em
ployeesNameLabel, employeesPhoneLabel,
employeesPostLabel, cb);



        }



    }


    public void availabilitiesView() {
        cal = Calendar.getInstance();  //
getting the current date

cal.setFirstDayOfWeek(Calendar.MONDAY); //
setting the Monday to be the first day of
the week instead of Sunday
        weekOfYear =
cal.get(Calendar.WEEK_OF_YEAR); // getting
the current week number

        gridPane = new GridPane(); //
creating a grid pane for the calendar and
all the functional buttons
        gridPane.setVgap(8); // setting
spacing between grid rows
```

```
        gridPane.setHgap(8);// setting
spacing between grid columns



        decrementByOneButton = new
Button("<");//   creating control buttons
for the calendar

decrementByOneButton.setOnAction(event1 ->
{
            decrementByOneButtonAction();
        });
        incrementByOneButton = new
Button(">");

incrementByOneButton.setOnAction(event1 ->
{
            incrementByOneButtonAction();
        });



decrementByOneButton.setMaxWidth(Double.MAX
_VALUE);//  creating label displaying week
number for the calendar

incrementByOneButton.setMaxWidth(Double.MAX
_VALUE);//setting buttons size to match the
grid field dimensions
        weekLabel = new Label("WEEK: " +
weekOfYear);
```

```
gridPane.getChildren().addAll(decrementByOn
eButton, weekLabel, incrementByOneButton);

gridPane.setConstraints(decrementByOneButto
n, 3, 1);//
        gridPane.setConstraints(weekLabel,
4, 1);           ////  setting all the
calendar nodes in grid

gridPane.setConstraints(incrementByOneButto
n, 5, 1);//



decrementByOneButton.setAlignment(Pos.CENTE
R_RIGHT);// setting the alignment so all
the elements look consistent




        logOutButton = new Button("Log
out");
        logOutButton.setOnAction(event1 ->
{
            hideWindow();
```

```
            loginController = new
LoginController();

loginController.goBackToLoginView();

        });

        Button ultimateScheduleButton= new
Button("Ultimate schedule");

ultimateScheduleButton.setOnAction(event1 -
> {
            hideWindow();

ultimateScheduleController.goToUltimateSche
duleView();
        });



        createEmployeeButton = new
Button("Create employee");

createEmployeeButton.setOnAction(event -> {
            CreateEmployeeController
createEmployeeController = new
CreateEmployeeController();

createEmployeeController.goToCreateEmployee
();
```

```
        });
        loginController= new
LoginController();
        String loginCredit =
loginController.passLoggedEmployee().getFul
lName();
        Label ifLoggedInLabel =new
Label("Welcome: " + loginCredit);

gridPane.setConstraints(ifLoggedInLabel, 0,
0, 3, 2, HPos.LEFT, VPos.TOP);

gridPane.getChildren().add(ifLoggedInLabel)
;



gridPane.setConstraints(ultimateScheduleBut
ton, 6, 12, 4, 1, HPos.RIGHT,
VPos.CENTER);//

gridPane.setConstraints(logOutButton, 0,
12, 3, 1);// setting buttons in the right
position in grid

gridPane.setConstraints(createEmployeeButto
n, 3, 12,3,1);//


gridPane.getChildren().addAll(logOutButton,
createEmployeeButton,
```

```
ultimateScheduleButton);


gridPane.getColumnConstraints().add(new
ColumnConstraints(130));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//setting columns
and rows constraints for the grid

gridPane.getRowConstraints().addAll(new
RowConstraints(90));//
```

```
gridPane.getRowConstraints().addAll(new
RowConstraints(30));//

gridPane.getRowConstraints().addAll(new
RowConstraints(30));//

gridPane.getRowConstraints().addAll(new
RowConstraints(33));//

gridPane.getRowConstraints().addAll(new
RowConstraints(63));//

gridPane.getRowConstraints().addAll(new
RowConstraints(63));//

gridPane.getRowConstraints().addAll(new
RowConstraints(63));//


customTableGridPane.getColumnConstraints().
add(new ColumnConstraints(100));//

customTableGridPane.getColumnConstraints().
add(new ColumnConstraints(60));//setting
columns constrains for the

customTableGridPane.getColumnConstraints().
add(new ColumnConstraints(30));//custom
table grid

customTableGridPane.getColumnConstraints().
```

```
add(new ColumnConstraints(5));//



        createLabels();  // CREATING LABELS
AND SETTING THEM TO GRID AT ROW 1 POSITIONS
1-7

        createButtons();  // CREATING
BUTTONS AND SETTING THEM TO GRID



        borderPane.setCenter(gridPane);

borderPane.setRight(createCustomTable());

        borderPane.setId("pane"); //setting
the backGround image

        scene = new Scene(borderPane, 878,
600);

scene.getStylesheets().add("backGround.css"
);  // using the CSS stylesheet

        stage = new Stage();

        stage.setScene(scene);
        stage.show();
```

```
    }

    public void hideWindow() {
        stage.hide();
    }

    /**
     * Buttons methods
     */

    public void  createButtons() {


        for(int j=0; j<shiftTypes.length; j
++) { // for every shift type
            for(int i=0; i<7; i++) { //
there is seven days of the week

                String id =
labels.get(i).getText() + shiftTypes[j]; //
creating buttons ID which consist of
                //date and shift type
                String date =
id.substring(0,10);//
                String shiftType =
id.substring(10, 11);// spliting the id so
we can use it in the parameters of the
query
                Button button = new
Button(); // creating a button for a
specific day and shift
```

```
                button.setId(id);
                button.setOnAction(event ->
{
                    updateCustomTable(date,
shiftType); // updating the information for
the buttons
                });

button.setMaxWidth(Double.MAX_VALUE);//

button.setMaxHeight(Double.MAX_VALUE);//
setting buttons size to match the grid
field dimensions


                buttons.add(button); //
adding a button to array list of buttons

gridPane.getChildren().add(button); //
adding a button to the grid calendar x
shift grid pane

gridPane.setConstraints(button, i + 1, j +
4); // dynamically setting the button in
the right position
                // in grid pane
            }
        }
    }
```

```
    public void updateButtons() {
        int indexButton = 0;// index which
is used to get a specific item from an
array list of labels for buttons

        for(int j=0; j<3; j++) { // for
every shift type
            for (int i = 0; i < 7; i++) {//
there is seven days of the week

                String id =
labels.get(i).getText() + shiftTypes[j];//
creating buttons ID which consist of
                //date and shift type
                String date =
id.substring(0,10);//
                String shiftType =
id.substring(10, 11);// splitting the id so
we can use it in the parameters of the
query
                Button button =
buttons.get(indexButton); // get a button
of a given index
                button.setId(id); //setting
a new id to the button (with updated date
and shift type)
                button.setOnAction(event ->
{
                    updateCustomTable(date,
shiftType);// update custom table
                });
```

```
                indexButton++;// in each
loop setting the pointer to next element
(incrementing the pointer)
            }
        }
    }

    public void
incrementByOneButtonAction() { //main motor
to changing the properties of the buttons
of the calendar x shift
        if(weekOfYear < 52) {  // checking
if the week is smaller than 52
            weekOfYear = weekOfYear + 1;//
incrementing the week of the year
            updateDateLabels(labels); //
updating the date labels
            updateButtons();// updating the
button information

customTableGridPane.getChildren().clear();
// clearing all the entries in custom table

        }
    }
    public void
decrementByOneButtonAction() {//main motor
to changing the properties of the buttons
of the calendar x shift
        if(weekOfYear > 1) {// checking if
the week is smaller than 52
```

```
            weekOfYear = weekOfYear - 1; //
decrementing the week of the year
            updateDateLabels(labels);//
updating the date labels
            updateButtons();// updating the
button information

customTableGridPane.getChildren().clear();/
/ clearing all the entries in custom table

        }
    }

    /**
     *
     * Labels methods
     *
     */
    public void createLabels (){

        //int[] weekDays =
{2,3,4,5,6,7,1}; // because Sunday has
number 1, we swap Monday(2) on that place -
project requirements

        for(int i =0; i<7; i++) {    // 7
times we create a label, from Monday label
to Sunday label

            cal.set(Calendar.WEEK_OF_YEAR,
cal.get(Calendar.WEEK_OF_YEAR)); // setting
```

```
a calendar to a desired week
            cal.set(Calendar.DAY_OF_WEEK,
weekDays[i]);

            String date=
simpleDateFormat.format(cal.getTime());//
labels Id format (date)
            Label label = new Label(); //
creating a label for a date
            label.setText(date);// setting
date to label
            label.setStyle("-fx-font-size:
8px");// setting size of the label
            labels.add(label);// adding
labels to array list of labels


gridPane.getChildren().add(label); // put
label to grid
            gridPane.setConstraints(label,
i + 1, 3); // position label at column
1...7, row 3
        }


    }

    public void updateDateLabels
(ArrayList<Label> labels) {
        weekLabel.setText("WEEK: " +
weekOfYear);  // setting the week label to
```

```
a week of interest
        cal = Calendar.getInstance(); //
getting the current date

cal.setFirstDayOfWeek(Calendar.MONDAY);//
setting the Monday to be the first day of
the week instead of Sunday


        for (int i = 0; i < labels.size();
i++) {
            cal.set(Calendar.WEEK_OF_YEAR,
weekOfYear);
            cal.set(Calendar.DAY_OF_WEEK,
weekDays[i]);
            String date =
simpleDateFormat.format(cal.getTime()); //
setting a string for a desired date
            labels.get(i).setText(date); //
setting the name of the label to desired
date


        }
    }



}

package Views;
```

```
import Controllers.AvailabilityController;
import Controllers.LoginController;
import
Controllers.ManagerConfirmationController;
import
Controllers.UltimateScheduleController;
import Models.Employee;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.geometry.VPos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;


/**
 * Created by tadas on 12/1/2015.
 */
public class UltimateScheduleView{
    SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy-MM-dd"); // sdf
    Calendar cal;
    String[] shiftTypes= {"N","D","E"};
    ArrayList<Label> labels= new
ArrayList<>();
```

```
    ArrayList<Button> buttons = new
ArrayList<>(); // array list of calendar x
shift buttons
    LoginController loginController= new
LoginController();
    ManagerConfirmationController
managerConfirmationController= new
ManagerConfirmationController();

    UltimateScheduleController
ultimateScheduleController= new
UltimateScheduleController();
    AvailabilityController
availabilityController = new
AvailabilityController();
    int[] weekDays = {2,3,4,5,6,7,1}; //
because Sunday has number 1, we swap
Monday(2) on that place - project
requirements
    public final boolean confirmed= true;

    Label weekLabel;
    Button incrementByOneButton;
    Button decrementByOneButton;

    Button logOutButton;

    GridPane gridPane;
    GridPane customTableGridPane= new
GridPane();
```

```
    BorderPane borderPane= new
BorderPane();
    Scene scene;
    Stage stage;

    int weekOfYear;


    public VBox createCustomTable() {

        VBox customTableView = new
VBox();  // VBox containing Hbox with
custom table column names and GridPane with
        // all the entries

        HBox customTableColumnBox= new
HBox();   // HBox containing all the column
names for the custom table

        Label customTableLabel= new
Label("Employees          ");   //
        Label customTableLabel1 = new
Label("Phone   ");        //   Column
labels
        Label customTableLable2 = new
Label("Post");            //

        customTableColumnBox.setPadding(new
Insets(186, 42, 30, 0)); // Setting padding
so the title column Hbox is
        // exactly matching our background
```

```
picture (same with grid pane of entries)

customTableColumnBox.setSpacing(10); //
setting spacing between column labels

customTableColumnBox.getChildren().addAll(c
ustomTableLabel, customTableLabel1,
customTableLable2);



customTableView.getChildren().addAll(custom
TableColumnBox, customTableGridPane); //
putting the column titles
        //and entries GridPane in a
matching

        return customTableView;
    }



    public void updateCustomTable(String
date, String shiftType){

customTableGridPane.getChildren().clear();
// clearing all the entries of the custom
table
        ArrayList<Employee>
confirmedEmployees=
ultimateScheduleController.getEmployeesForU
```

```
ltimateSchedule(date, shiftType,
confirmed);
       // putting all the confirmed
employees of the given date and shift in an
array list

        for (int i=0;
i<confirmedEmployees.size(); i++){  // for
the all the employees

            Label employeesNameLabel= new
Label(confirmedEmployees.get(i).getFullName
());  // creating and setting name label
            // to their name
            employeesNameLabel.setStyle("-
fx-font-size: 10px");
            Label employeesPhoneLabel = new
Label(confirmedEmployees.get(i).getMobileNu
mber()); //creating and setting phone label
            // to their phone number
            employeesPhoneLabel.setStyle("-
fx-font-size: 10px");
            Label employeesPostLabel = new
Label(confirmedEmployees.get(i).getPostcode
()); //creating and setting postcode label
            // to their phone postcode
            employeesPostLabel.setStyle("-
fx-font-size: 10px"); // setting font size
to 10px
```

```
customTableGridPane.setConstraints(employee
sNameLabel, 0,i);

customTableGridPane.setConstraints(employee
sPhoneLabel, 1, i); // placing the labels
in right spots in grid

customTableGridPane.setConstraints(employee
sPostLabel, 2, i);


customTableGridPane.getChildren().addAll(em
ployeesNameLabel, employeesPhoneLabel,
employeesPostLabel);
            // adding labels to the grid
        }
    }


    public void
createUltimateScheduleView() {
        cal = Calendar.getInstance();  //
getting the current date

cal.setFirstDayOfWeek(Calendar.MONDAY); //
setting the Monday to be the first day of
the week instead of Sunday
        weekOfYear =
cal.get(Calendar.WEEK_OF_YEAR); // getting
the current week number
```

```
        gridPane = new GridPane(); //
creating a grid pane for the calendar and
all the functional buttons
        gridPane.setVgap(8); // setting
spacing between grid rows
        gridPane.setHgap(8);// setting
spacing between grid columns

        decrementByOneButton = new
Button("<"); //

decrementByOneButton.setOnAction(event1 ->
{
            decrementByOneButtonAction();
        });
        incrementByOneButton = new
Button(">"); //  //   creating control
buttons for the calendar

incrementByOneButton.setOnAction(event1 ->
{
            incrementByOneButtonAction();
        });

decrementByOneButton.setMaxWidth(Double.MAX
_VALUE);//

incrementByOneButton.setMaxWidth(Double.MAX
_VALUE);//setting buttons size to match the
grid field dimensions
```

```
        weekLabel = new Label("WEEK: " +
weekOfYear);//  creating label displaying
week number for the calendar


gridPane.setConstraints(decrementByOneButto
n, 3, 1);//
        gridPane.setConstraints(weekLabel,
4, 1);           //  setting all the
calendar nodes in grid

gridPane.setConstraints(incrementByOneButto
n, 5, 1);//


decrementByOneButton.setAlignment(Pos.CENTE
R_RIGHT);// setting the alignment so all
the elements look consistent


gridPane.getChildren().addAll(decrementByOn
eButton, weekLabel,
incrementByOneButton);// adding all the
calendar
        // nodes in grid


        Button goBack = new Button("Go
back");      // create a back button
        goBack.setOnAction(event1 -> {
```

```
            //LoginController
loginController= new LoginController();
            hideWindow();
            if ( false ==
loginController.passLoggedEmployee().getIfM
anager()){ // checking if a logged user is
                //an employee or a manager
and redirecting him/her to a proper view
                 //AvailabilityController
availabilityController = new
AvailabilityController();

availabilityController.goToAvailabilityView
();}
            else {
                //
ManagerConfirmationController
managerConfirmationController= new
ManagerConfirmationController();

managerConfirmationController.goToManagerCo
nfirmationView();

            }
        });


        logOutButton = new Button("Log
out");   // button for logging out
        logOutButton.setOnAction(event -> {
            hideWindow();
```

```
            LoginController loginController
= new LoginController();

loginController.goBackToLoginView();

        });
        loginController= new
LoginController();
        String loginCredit =
loginController.passLoggedEmployee().getFul
lName();
        Label ifLoggedInLabel =new
Label("Welcome: " + loginCredit);

gridPane.setConstraints(ifLoggedInLabel, 0,
0, 3, 2, HPos.LEFT, VPos.TOP);

gridPane.getChildren().add(ifLoggedInLabel)
;


gridPane.setConstraints(logOutButton, 0,
12);       // setting buttons in the right
position in grid
        gridPane.setConstraints(goBack, 6,
12, 2, 1, HPos.RIGHT, VPos.CENTER);//



gridPane.getChildren().addAll(logOutButton,
goBack);// adding buttons to grid
```

```
gridPane.getColumnConstraints().add(new
ColumnConstraints(130));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));// setting columns
and rows constraints for the grid

gridPane.getColumnConstraints().add(new
ColumnConstraints(56));//

gridPane.getRowConstraints().addAll(new
RowConstraints(90));//

gridPane.getRowConstraints().addAll(new
```

```
RowConstraints(30));//

gridPane.getRowConstraints().addAll(new
RowConstraints(30));//

gridPane.getRowConstraints().addAll(new
RowConstraints(33));//

gridPane.getRowConstraints().addAll(new
RowConstraints(63));//

gridPane.getRowConstraints().addAll(new
RowConstraints(63));//

gridPane.getRowConstraints().addAll(new
RowConstraints(63));//



customTableGridPane.getColumnConstraints().
add(new ColumnConstraints(100));//

customTableGridPane.getColumnConstraints().
add(new ColumnConstraints(60));// setting
columns constrains for the

customTableGridPane.getColumnConstraints().
add(new ColumnConstraints(30));// custom
table grid

customTableGridPane.getColumnConstraints().
```

```
add(new ColumnConstraints(5));//



        createLabels();  // CREATING LABELS
AND SETTING THEM TO GRID

        createButtons(); // CREATING
BUTTONS AND SETTING THEM TO GRID

        borderPane.setCenter(gridPane);////
using border pane to set all the components
(calendar grid pane

borderPane.setRight(createCustomTable());
// and custom table VBox to scene


        borderPane.setId("pane"); // for
setting the backGround image

        scene = new Scene(borderPane, 878,
600);

scene.getStylesheets().add("backGround.css"
); // using the CSS stylesheet

        stage = new Stage();
        stage.setScene(scene);
        stage.show();
```

```
    }

    /**
     * Buttons methods
     */

    public void  createButtons() {

        //LoginController loginController=
new LoginController();
        //ManagerConfirmationController
managerConfirmationController= new
ManagerConfirmationController();

        for(int j=0; j<shiftTypes.length; j
++) { // for every shift type
            for(int i=0; i<7; i++) { //
there is seven days of the week

                String id =
labels.get(i).getText() + shiftTypes[j]; //
creating buttons ID which consist of
                //date and shift type
                String date =
id.substring(0,10);//
                String shiftType =
id.substring(10, 11);// spliting the id so
we can use it in the parameters of the
query
                Button button = new
Button(); // creating a button for a
```

```
specific day and shift
                button.setId(id);
                button.setOnAction(event ->
{
                    updateCustomTable(date,
shiftType); // updating the information for
the buttons
                });

button.setMaxWidth(Double.MAX_VALUE);//

button.setMaxHeight(Double.MAX_VALUE);//
setting buttons size to match the grid
field dimensions

                boolean ifBooked =
ultimateScheduleController.ifYouAreBooked(d
ate, shiftType,
loginController.passLoggedEmployee().getEmp
Id(),confirmed );
                // query checking a person
is confirmed on a given date and shift
                if (ifBooked)
                {

                    button.setStyle("-fx-
base: #00b300"); // setting a green color
for the button when logged employee is
working
                    //on a give day at a
given shift
```

```
                }

                buttons.add(button); //
adding a button to array list of buttons

gridPane.getChildren().add(button); //
adding a button to the grid calendar x
shift grid pane

gridPane.setConstraints(button, i + 1, j +
4); // dynamically setting the button in
the right position
                // in grid pane
            }
        }
    }

    public void updateButtons() {
        int indexButton = 0; // index which
is used to get a specific item from an
array list of labels for buttons
        //ManagerConfirmationController
managerConfirmationController= new
ManagerConfirmationController();
        //LoginController loginController=
new LoginController();
        for(int j=0; j<shiftTypes.length; j
++) { // for every shift type
            for (int i = 0; i < 7; i++) {//
there is seven days of the week
```

```
                String id =
labels.get(i).getText() + shiftTypes[j]; //
creating buttons ID which consist of
                //date and shift type
                String date =
id.substring(0, 10);//
                String shiftType =
id.substring(10, 11);// splitting the id so
we can use it in the parameters of the
query
                Button button =
buttons.get(indexButton);// get a button of
a given index
                button.setOnAction(event ->
{
                    updateCustomTable(date,
shiftType); // update custom table
                });
                button.setId(id);//setting
a new id to the button (with updated date
and shift type)
                boolean ifBooked =
ultimateScheduleController.ifYouAreBooked(d
ate, shiftType,
loginController.passLoggedEmployee().getEmp
Id(), confirmed);
                // query checking a person
is confirmed on a given date and shift
                if (ifBooked == true)
                {
```

```
                    button.setStyle("-fx-
base: #00b300");// setting a green color
for the button when logged employee is
working

                }
                else
                {
                    button.setStyle("-fx-
base: #e3e3e3");// setting a gray color for
the button when logged employee is not
working

                }

                indexButton++;// in each
loop setting the pointer to next element
(incrementing the pointer)
            }
        }
    }



    /**
     *
     * Labels methods
     *
     */
    public void createLabels (){
```

```
        //int[] weekDays =
{2,3,4,5,6,7,1}; // because Sunday has
number 1, we swap Monday(2) on that place -
project requirements

        for(int i =0; i<7; i++) {    // 7
times we create a label, from Monday label
to Sunday label

            cal.set(Calendar.WEEK_OF_YEAR,
cal.get(Calendar.WEEK_OF_YEAR)); // setting
a calendar to a desired week
            cal.set(Calendar.DAY_OF_WEEK,
weekDays[i]);

            String date=
simpleDateFormat.format(cal.getTime());//
labels Id format (date)
            Label label = new Label(); //
creating a label for a date
            label.setText(date);// setting
date to label
            label.setStyle("-fx-font-size:
8px");// setting size of the label
            labels.add(label);// adding
labels to array list of labels


gridPane.getChildren().add(label); // put
label to grid
            gridPane.setConstraints(label,
```

```
i + 1, 3); // position label at column
1...7, row 3
        }


    }

    public void updateDateLabels
(ArrayList<Label> labels) {
        weekLabel.setText("WEEK: " +
weekOfYear);  // setting the week label to
a week of interest
        cal = Calendar.getInstance(); //
getting the current date

cal.setFirstDayOfWeek(Calendar.MONDAY);//
setting the Monday to be the first day of
the week instead of Sunday


        for (int i = 0; i < labels.size();
i++) {
            cal.set(Calendar.WEEK_OF_YEAR,
weekOfYear);
            cal.set(Calendar.DAY_OF_WEEK,
weekDays[i]);
            String date =
simpleDateFormat.format(cal.getTime()); //
setting a string for a desired date
            labels.get(i).setText(date); //
setting the name of the label to desired
```

```
date


        }
    }


    public void
incrementByOneButtonAction() { //main motor
to changing the properties of the buttons
of the calendar x shift
        if(weekOfYear < 52) {  // checking
if the week is smaller than 52
            weekOfYear = weekOfYear + 1;//
incrementing the week of the year
            updateDateLabels(labels); //
updating the date labels
            updateButtons();// updating the
button information

customTableGridPane.getChildren().clear();
// clearing all the entries in custom table

        }
    }
    public void
decrementByOneButtonAction() {//main motor
to changing the properties of the buttons
of the calendar x shift
        if(weekOfYear > 1) {// checking if
the week is smaller than 52
```

```
            weekOfYear = weekOfYear - 1; //
decrementing the week of the year
            updateDateLabels(labels);//
updating the date labels
            updateButtons();// updating the
button information

customTableGridPane.getChildren().clear();/
/ clearing all the entries in custom table

        }
    }

    public void hideWindow()
{stage.hide();}// hiding a stage

}
```