Appendix C: Code

PSGEventsManagementSystem (main class)

```
package Class;

import javafx.application.Application;
import javafx.application.Platform;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.stage.Stage;

public class PSGEventsManagementSystem extends Application {

    // Application start
    @Override
    public void start(Stage stage) throws Exception {
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/FXML/Main.fxml"));
        Parent root = loader.load();
        Scene scene = new Scene(root);
        scene.getStylesheets().add("/CSS/Main.css");
        stage.getIcons().add(new Image("images/PW_Symbol.jpg"));
        stage.setTitle("PSG Event Management System");
        stage.setScene(scene);
        stage.setResizable(false);
        stage.sizeToScene();
        stage.show();
        stage.setOnCloseRequest(e -> Platform.exit());
    }
```

```
    public static void main(String[]args) {
        System.out.println("\n\n\n----------------------------------
-------------");
        launch(args);
        System.out.println("----------------------------------------
-------\n\n\n");
    }
}
```

MainController

```
package Class;


import com.google.common.hash.Hashing;
import com.jfoenix.controls.JFXButton;
import com.jfoenix.controls.JFXDialog;
import com.jfoenix.controls.JFXDialogLayout;
import com.jfoenix.controls.JFXDrawer;
import com.jfoenix.controls.JFXHamburger;
import com.jfoenix.controls.JFXListView;
import com.jfoenix.controls.JFXPasswordField;
import com.jfoenix.controls.JFXSnackbar;
import com.jfoenix.controls.JFXTextField;
import com.jfoenix.effects.JFXDepthManager;
import
com.jfoenix.transitions.hamburger.HamburgerBasicCloseTransition;
import com.jfoenix.validation.RequiredFieldValidator;
import java.awt.Desktop;
import java.io.File;
import java.io.IOException;
import java.net.URL;
```

```
import java.nio.charset.StandardCharsets;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.animation.FadeTransition;
import javafx.animation.Interpolator;
import javafx.animation.ScaleTransition;
import javafx.application.Platform;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Pane;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Text;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.util.Duration;

public class MainController implements Initializable {
```

```
    @FXML
    private AnchorPane base; //Base pane
    @FXML
    private StackPane dialogStack; //Transparent pane to display
messages
    @FXML
    private Pane paneLogin; //Pane showing the login screen
    @FXML
    private JFXTextField fieldUser_Login; //Username field on Login
screen
    @FXML
    private JFXPasswordField fieldPass_Login; //Password field on
Login screen
    @FXML
    private JFXButton buttonLogin_Login; //Submit button on Login
screen
    @FXML
    private JFXButton buttonClose_Login; //Close button on Login
screen
    @FXML
    private JFXButton buttonView_Login; //View button on Login screen.
For read only access.
    @FXML
    private JFXHamburger buttonMenu; //Menu button
    @FXML
    private JFXDrawer drawerMenu; //Menu drawer
    @FXML
    private Pane paneEvents; //Pane showing the list of events
    @FXML
    private JFXButton buttonEdit_Events; //Edit selected event record
    @FXML
    private JFXButton buttonAdd_Events; //Add new event record
    @FXML
```

```
    private JFXListView<Pane> listView_Events; //List of all events
    @FXML
    private JFXButton buttonRefresh; //Refresh Events
    @FXML
    private Pane paneHome; //Pane showing house standings
    @FXML
    private Pane podium1_Home; //Coloured background for 1st place
house
    @FXML
    private Label name1_Home; //Name of 1st place house
    @FXML
    private Label points1_Home; //Points of 1st place house
    @FXML
    private Pane podium2_Home; //Coloured background for 2nd place
house
    @FXML
    private Label name2_Home; //Name of 2nd place house
    @FXML
    private Label points2_Home; //Points of 2nd place house
    @FXML
    private Pane podium3_Home; //Coloured background for 3rd place
house
    @FXML
    private Label name3_Home; //Name of 3rd place house
    @FXML
    private Label points3_Home; //Points of 3rd place house
    @FXML
    private Pane podium4_Home; //Coloured background for 4th place
house
    @FXML
    private Label name4_Home; //Name of 4th place house
    @FXML
    private Label points4_Home; //Points of 4th place house
```

```
    private int[] listCellNo = {-1,-1}; //ID number for element in
list
    static Pane visiblePane; //Currently Visible pane
    static int accessLevel; //Access level of user

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        try {
//            Create connection to database "PSGeventmanage"
            SQL sql = new SQL("/PSGeventmanage");

//            Click on pane, will draw focus from any other element
            base.setOnMouseClicked((MouseEvent evt) -> {
                base.requestFocus();
            });

//            LOGIN SCREEN
//            Click on pane, will draw focus from any other element
            paneLogin.setOnMouseClicked((MouseEvent evt) -> {
                paneLogin.requestFocus();
            });

//              If the field is empty, but focus is lost, the login
button will be disabled and error message will display
            RequiredFieldValidator        valU        =        new
RequiredFieldValidator();
            valU.setMessage("Username Required");
            fieldUser_Login.getValidators().add(valU);

fieldUser_Login.focusedProperty().addListener((ObservableValue<?
extends Boolean> observable, Boolean oldValue, Boolean newValue) -> {
                if (oldValue){
```

```
                    if (!fieldUser_Login.validate()) {
                        fieldUser_Login.setUnFocusColor(Color.RED);
                        buttonLogin_Login.setDisable(true);
                    } else {

fieldUser_Login.setUnFocusColor(Color.web("#01579b"));
                        buttonLogin_Login.setDisable(false);
                    }
                }
            });


//              If the field is empty, but focus is lost, the login
button will be disabled and error message will display
            RequiredFieldValidator        valP        =        new
RequiredFieldValidator();
            valP.setMessage("Password Required");
            fieldPass_Login.getValidators().add(valP);

fieldPass_Login.focusedProperty().addListener((ObservableValue<?
extends Boolean> observable, Boolean oldValue, Boolean newValue) -> {
                if (oldValue){
                    if (!fieldPass_Login.validate()) {
                        fieldPass_Login.setUnFocusColor(Color.RED);
                        buttonLogin_Login.setDisable(true);
                    } else {

fieldPass_Login.setUnFocusColor(Color.web("#01579b"));
                        buttonLogin_Login.setDisable(false);
                    }
                }
            });


//            On button click, login.
            buttonLogin_Login.setOnAction(Event -> {
```

```
                final           String           hashed           =
Hashing.sha256().hashString(fieldPass_Login.getText(),
StandardCharsets.UTF_8).toString();  //Convert  password  input  into
cypertext

                String  login  =  SQL.login(fieldUser_Login.getText(),
hashed); //Verify credentials from SQL Database

                fieldPass_Login.setText(""); //Clear password field

//                Scene trasition animations

                ScaleTransition         scale         =         new
ScaleTransition(Duration.millis(300));

                scale.setInterpolator(Interpolator.EASE_BOTH);

                JFXSnackbar msg_Login = new JFXSnackbar(dialogStack);
//Message container

                if (login.substring(0, 5).equals("match")) {

                    fieldUser_Login.setText("");   //Clear   username
field

                    scale.setNode(paneLogin);

                    scale.setByX(-1);

                    scale.setByY(-1);

                    scale.play(); //Run animation

                    scale.setOnFinished(e1 -> { //Run following code
after the animation ends

                        paneLogin.setVisible(false);

                        scale.setNode(paneHome);

                        scale.setDuration(Duration.ONE);

                        scale.play();

                        scale.setOnFinished(e2 -> {

                            scale.setToX(1.0);

                            scale.setToY(1.0);

                            scale.setDuration(Duration.millis(300));

                            paneHome.setVisible(true);

                            visiblePane = paneHome;

                            scale.play();

                            scale.setOnFinished(e3 -> {
```

```
                                msg_Login.show("Login    Successful.
Access   Level:   "+login.substring(5,   6).replaceAll("1",   "Edit
Only").replaceAll("2",   "Add/Edit   Only").replaceAll("3",   "Full
Access"), 2000);
                                buttonMenu.setVisible(true);
                                drawerMenu.setVisible(true);
                                accessLevel(login.substring(5, 6));
                            });
                        });
                    });
                } else {
                    msg_Login.show("Login     Failed.     Incorrect
Credentials.", 2000);
                }
                buttonLogin_Login.setDisable(true);
            });


            buttonView_Login.setOnAction(Event -> { //Identical to
Login button, except for username/password validation.
                fieldUser_Login.reset();
                fieldUser_Login.setText("");
                fieldPass_Login.reset();
                fieldPass_Login.setText("");
                ScaleTransition         scale         =         new
ScaleTransition(Duration.millis(300));
                scale.setInterpolator(Interpolator.EASE_BOTH);
                scale.setNode(paneLogin);
                scale.setByX(-1);
                scale.setByY(-1);
                scale.play();
                scale.setOnFinished(e1 -> {
                    paneLogin.setVisible(false);
                    scale.setNode(paneHome);
                    scale.setDuration(Duration.ONE);
```

```
                    scale.play();
                    scale.setOnFinished(e2 -> {
                        scale.setToX(1.0);
                        scale.setToY(1.0);
                        scale.setDuration(Duration.millis(300));
                        paneHome.setVisible(true);
                        visiblePane = paneHome;
                        scale.play();
                        scale.setOnFinished(e3 -> {
                            JFXSnackbar     msg_Login     =     new
JFXSnackbar(dialogStack);
                            msg_Login.show("Login Successful. Access
Level: View Only", 2000);
                            buttonMenu.setVisible(true);
                            drawerMenu.setVisible(true);
                            accessLevel("0");
                        });
                    });
                });
            });

            buttonClose_Login.setOnAction(Event -> {
                //Scene trasition animations
                FadeTransition          fade          =          new
FadeTransition(Duration.millis(300),paneLogin); //Fade Animation
                fade.setToValue(0); //Fade to transparency 0%
                fade.play();
                fade.setOnFinished(e -> {
                    Platform.exit();
                    SQL.close();
                });
            });
```

```
            //MENU DRAWER
            FXMLLoader            loader            =            new
FXMLLoader(getClass().getResource("/FXML/HamburgerDrawer.fxml"));
//Create loader object
            VBox box = null;
            try {
                box  =  loader.load();  //Load  the  elements  for  the
drawer
            } catch (IOException ex) {

Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
            }
            HamburgerDrawerController          menuDrawer          =
loader.getController(); //load drawer controller
            drawerMenu.setSidePane(box); //Set the elements for the
drawer
            JFXDepthManager.setDepth(drawerMenu, 2); //Shadow beneath
drawer

            HamburgerBasicCloseTransition   HamTransition   =   new
HamburgerBasicCloseTransition(buttonMenu); //Animation for the Menu
button
            HamTransition.setRate(-1); //Default state of menu button

            buttonMenu.setOnMouseClicked(Event -> { //Run following
code when button is clicked
                HamTransition.setRate(HamTransition.getRate()*-1);
//Switch between default and activated state
                HamTransition.play();
                if (drawerMenu.isShown()){ //Checks whether menu is
shown
                    drawerMenu.setMouseTransparent(true);  //Ignores
mouse clicks
                    drawerMenu.close(); //Closes menu
                    listView_Events.setMouseTransparent(false);
```

```
                } else if (visiblePane == paneEvents) {
                    listView_Events.setMouseTransparent(true);
                    drawerMenu.open(); //Opens menu
                    drawerMenu.setMouseTransparent(false); //Ignores
mouse clicks
                } else {
                    drawerMenu.open(); //Opens menu
                    drawerMenu.setMouseTransparent(false); //Ignores
mouse clicks
                }
            });

            menuDrawer.buttonClose.setOnAction(Event -> {
                FadeTransition          fade          =          new
FadeTransition(Duration.millis(300),visiblePane); //Fade Animation
                fade.setToValue(0); //Fade to transparency 0%
                drawerMenu.close();
                drawerMenu.setOnDrawerClosed(h -> {
                    fade.play();
                    fade.setOnFinished(e -> {
                        Platform.exit(); //Exit application
                    });
                });
            });

            menuDrawer.buttonLogout.setOnAction(Event -> {
                dialogStack.setMouseTransparent(false);
                JFXDialogLayout content = new JFXDialogLayout();
                JFXDialog   dialog   =   new   JFXDialog(dialogStack,
content, JFXDialog.DialogTransition.CENTER);
                JFXButton button1 = new JFXButton("Logout");
                JFXButton button2 = new JFXButton("Cancel");
```

```
                content.setHeading(new Label("ARE YOU SURE YOU WANT
TO                    LOGOUT?",new                    ImageView(new
Image("/images/ic_account_circle_black_48dp_2x.png",32.0,32.0,true,t
rue))));
                content.setBody(new  Text("Please  save  any  unsaved
changes before proceeding."));
                content.setActions(button1,button2);
                button1.setOnAction((e) -> {
                    dialog.close();
                    dialogStack.setMouseTransparent(true);
                    FadeTransition        fade        =        new
FadeTransition(Duration.millis(300),visiblePane); //Fade Animation
                    ScaleTransition        scale        =        new
ScaleTransition(Duration.millis(300)); //Grow/Shrink Animation
                    scale.setInterpolator(Interpolator.EASE_BOTH);
//Animation accelerates and decelerates
                    fade.setToValue(0);
                    fade.setDuration(Duration.millis(300));
                    drawerMenu.close();
                    HamTransition.setRate(HamTransition.getRate()*-
1);
                    HamTransition.play();
                    buttonMenu.setVisible(false);
                    fade.play();
                    fade.setOnFinished(e1 -> {
                        visiblePane.setVisible(false);
                        paneHome.setOpacity(1.0);
                        paneEvents.setOpacity(0.0);
                        scale.setNode(paneLogin);
                        scale.setToX(1.0);
                        scale.setToY(1.0);
                        scale.setDuration(Duration.millis(300));
                        paneLogin.setVisible(true);
                        scale.play();
```

```
                        scale.setOnFinished(e2 -> {
                            fade.setNode(paneLogin);
                            fade.setFromValue(0.0);
                            fade.setToValue(1.0);
                            visiblePane = null;
                        });
                    });
                });
                button2.setOnAction(e -> {
                    dialog.close();
                    dialogStack.setMouseTransparent(true);
                });
                dialog.setOverlayClose(false);
                dialog.show();
            });

            menuDrawer.buttonSettings.setOnAction(e -> {
                try {
                    Desktop.getDesktop().open(new
File("./Settings.txt"));
                } catch (IOException ex) {

Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
                }
            });



            //HOME SCREEN
            //House standings
            populateHouseRank();

            //EVENTS SCREEN
```

```
            //Events List
            populateEventList(); //Fill List with data from DB


listView_Events.getSelectionModel().selectedItemProperty().addListen
er((ObservableValue<? extends Pane> observable, Pane oldValue, Pane
newValue) -> {
                if (newValue != null){
                    listCellNo[0]                                   =
Integer.parseInt(newValue.getId().replaceFirst("\\.(.*)",""));  //Set
local id of currently selected cell
                    listCellNo[1]                                   =
Integer.parseInt(newValue.getId().replaceFirst("(.*)\\.",""));  //Set
database id of currently selected cell
                }
            });


            //Interaction with Events List
            buttonAdd_Events.setOnAction(Event -> {
                launchEventModify();
            });


            buttonEdit_Events.setOnAction(Event -> {
                if (listCellNo[0] >= 0){
                    launchEventModify(listCellNo[1]);
                } else {
                    JFXSnackbar msg = new JFXSnackbar(paneEvents);
                    msg.show("Please select entry to edit",4000);
                }
            });


            buttonRefresh.setOnAction(Event -> {
                populateEventList();
            });
```

```
//-----------------------------------------------------------------
----------- MAIN END ------------------------------------------------
--------------------------------


        } catch (SQLException ex) {

Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
            if    (ex.getMessage().startsWith("Communications    link
failure")){
                sqlDisconnect();
            }else  if(ex.getMessage().startsWith("Access  denied  for
user")){
                sqlDisconnect();
            }else             if(ex.getMessage().startsWith("Unknown
database")){
                this.dialogStack.setMouseTransparent(false);
                JFXDialogLayout content = new JFXDialogLayout();
                JFXDialog  dialog  =  new  JFXDialog(this.dialogStack,
content, JFXDialog.DialogTransition.CENTER);
                JFXButton button = new JFXButton("EXIT");
                content.setHeading(new                        Label("
COMMUNICATION     WITH     DATABASE     FAILED",new     ImageView(new
Image("/images/ic_sync_problem_black_48dp_2x.png",32.0,32.0,true,tru
e))));
                content.setBody(new Text("Please ensure that Server
is running.\nAnd run PSGEventsManagementSoftware.sql."));
                content.setActions(button);
                button.setOnAction((e) -> {
                    dialog.close();
                    Platform.exit();
                });
                dialog.setOverlayClose(false);
                dialog.show();
            }
```

```
        }
    }


    private void accessLevel(String level){
        MainController.accessLevel = Integer.parseInt(level);
        switch (MainController.accessLevel){
            case 3: //Full Access
                buttonAdd_Events.setDisable(false);
                buttonEdit_Events.setText("Edit");
                break;
            case 2: //Add/Edit Access
                buttonAdd_Events.setDisable(false);
                buttonEdit_Events.setText("Edit");
                break;
            case 1: //Edit Only Access
                buttonAdd_Events.setDisable(true);
                buttonEdit_Events.setText("Edit");
                break;
            case 0: //View Only Access
                buttonAdd_Events.setDisable(true);
                buttonEdit_Events.setText("View");
                break;
        }
    }


    public void sqlDisconnect(){
        this.dialogStack.setMouseTransparent(false);
        JFXDialogLayout content = new JFXDialogLayout();
        JFXDialog dialog = new JFXDialog(this.dialogStack, content,
JFXDialog.DialogTransition.CENTER);
        JFXButton exit = new JFXButton("EXIT");
```

```
        JFXButton setup = new JFXButton("SETTINGS");

        content.setHeading(new Label("            COMMUNICATION WITH
SERVER                   FAILED",new                   ImageView(new
Image("/images/ic_sync_problem_black_48dp_2x.png",32.0,32.0,true,tru
e))));

        content.setBody(new Text("You may not be connected to the
server or\nthe server settings are incorrect. Please contact\nthe
Administrator for more information."));

        content.setActions(setup, exit);

        exit.setOnAction((e) -> {

            dialog.close();

            Platform.exit();

        });

        setup.setOnAction(e ->{

            doSettings();

        });

        dialog.setOverlayClose(false);

        dialog.show();

    }


    private void doSettings(){

        try {

            Desktop.getDesktop().open(new File("./Settings.txt"));

        } catch (IOException ex) {


Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);

        }

    }


    String color1;

    String color2;

    String color3;

    String color4;
```

```
    @FXML
    public void populateHouseRank(){
        String[][] rank = SQL.houseRank();
        name1_Home.setText(rank[0][0]);
        points1_Home.setText(rank[0][1]);
        podium1_Home.setStyle("-fx-background-
color:"+rank[0][2]+";");
        name2_Home.setText(rank[1][0]);
        points2_Home.setText(rank[1][1]);
        podium2_Home.setStyle("-fx-background-
color:"+rank[1][2]+";");
        name3_Home.setText(rank[2][0]);
        points3_Home.setText(rank[2][1]);
        podium3_Home.setStyle("-fx-background-
color:"+rank[2][2]+";");
        name4_Home.setText(rank[3][0]);
        points4_Home.setText(rank[3][1]);
        podium4_Home.setStyle("-fx-background-
color:"+rank[3][2]+";");

        color1 = rank[0][2];
        color2 = rank[1][2];
        color3 = rank[2][2];
        color4 = rank[3][2];
    }

    private void populateEventList(){
        listView_Events.getItems().clear();
        try {
            ResultSet events = SQL.eventList(); //Get event data from
DB
            int c = 0;
```

```
            while (events.next()){
                FXMLLoader loader = new FXMLLoader(getClass().getResource("/FXML/CellEvent.fxml")); //Create Loader
                Pane listCell = loader.load(); //Load FXML
                CellEventController cellEvt = loader.getController(); //Load Controller
                cellEvt.setEventName(events.getString(2));     //Set event name
                cellEvt.setEventGrades(events.getInt(3));      //Set event grades

cellEvt.setEventTime(events.getString(4).substring(0,   16));   //Set event time
                listCell.setId(c+"."+events.getString(1)); //Set the corresponding ID for the list cell from DB
                c++;
                listView_Events.getItems().add(listCell); //Add Cell to list
            }
        } catch (Exception ex) {

Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, null, ex);
        }
        if                    (listView_Events.getItems().isEmpty()) listView_Events.setVisible(false); //Hide the List if empty
    }


    private void launchEventModify(){ //Open a new window with editable form for Events
        try {
            Stage stage = new Stage(); //New Window
            stage.initModality(Modality.APPLICATION_MODAL); //Disable all other windows of application
```

```
            Parent                          root                          =
FXMLLoader.load(getClass().getResource("/FXML/EventRecordEditor.fxml
")); //Load visual file

            stage.getIcons().add(new Image("images/PW_Symbol.jpg"));
//Icon for Window

            stage.setTitle("PSG Event Management System - Editing
Event"); //Text for Window

            Scene scene = new Scene(root);

            scene.getStylesheets().add("/CSS/Main.css");

            stage.setScene(scene);

            stage.setResizable(false);

            stage.sizeToScene();

            stage.show();

        } catch (IOException ex) {


Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);

        }

    }


    private String houseFromInt(int house){ //Convert house number
into house name

        String houseName = null;

        switch (house){

            case 1:

                houseName = "Air";

                break;

            case 2:

                houseName = "Earth";

                break;

            case 3:

                houseName = "Fire";

                break;

            case 4:
```

```
                houseName = "Water";
                break;
        }
        return houseName;
    }


    private void launchEventModify(int id){ //Open a new window with editable form for Events
        try {
            Stage stage = new Stage(); //New Window
            stage.initModality(Modality.APPLICATION_MODAL); //Disable all other windows of application
            FXMLLoader loader = new FXMLLoader(getClass().getResource("/FXML/EventRecordEditor.fxml"));
            Parent root = loader.load(); //Load visual file
            EventRecordEditorController evtRec = loader.getController(); //Load visual file controller


            ResultSet event = SQL.loadEvent(id); //Load event data
            event.first(); //First record of resultset
            String winner1 = houseFromInt(event.getInt(9));
            String winner2 = houseFromInt(event.getInt(10));
            String winner3 = houseFromInt(event.getInt(11));
            ResultSet participants = SQL.loadParticipants(id); //Load participants in event
            participants.last(); //Last record of resultset


            evtRec.loadEvent(id, event.getString(1), event.getString(2), event.getString(3), event.getInt(4), participants.getRow(), event.getInt(5), event.getInt(6), event.getInt(7), event.getInt(8), winner1, winner2, winner3, event.getString(12)); //Load data into fxml


            stage.getIcons().add(new Image("images/PW_Symbol.jpg")); //Icon for Window
```

```
            stage.setTitle("PSG Event Management System - Editing
Event"); //Text for Window
            Scene scene = new Scene(root);
            scene.getStylesheets().add("/CSS/Main.css");
            stage.setScene(scene);
            stage.setResizable(false);
            stage.sizeToScene();
            stage.show();
        } catch (Exception ex) {

Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }


    @FXML
    private void viewSwitch(ActionEvent event) {
        FadeTransition             fade             =             new
FadeTransition(Duration.millis(300),visiblePane);
        fade.setToValue(0.0);
        fade.play();
        fade.setOnFinished(e2 -> {
            visiblePane.setVisible(false);
            if (visiblePane != paneHome){

listView_Events.getSelectionModel().clearSelection();
                visiblePane = paneHome;
            } else if (visiblePane != paneEvents)
                visiblePane = paneEvents;
            fade.setNode(visiblePane);
            fade.setToValue(1.0);
            fade.play();
            visiblePane.setVisible(true);
```

```
            fade.setOnFinished(e3 -> {
                fade.pause();
            });
        });
    }

    @FXML
    private void houseMembers(MouseEvent event) {
        try {
            Stage stage = new Stage();
            stage.initModality(Modality.APPLICATION_MODAL);
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/FXML/HouseMembers.fxml"));
            Parent root = loader.load();
            HouseMembersController houseMembersCon =
loader.getController();
            if(event.getSceneX() < 180){
                houseMembersCon.setHouseName(name3_Home.getText());
                houseMembersCon.setHouseColor(color3);
            } else if(event.getSceneX() < 350) {
                houseMembersCon.setHouseName(name1_Home.getText());
                houseMembersCon.setHouseColor(color1);
            } else if(event.getSceneX() < 500) {
                houseMembersCon.setHouseName(name2_Home.getText());
                houseMembersCon.setHouseColor(color2);
            } else {
                houseMembersCon.setHouseName(name4_Home.getText());
                houseMembersCon.setHouseColor(color4);
            }

            Scene scene = new Scene(root);
            scene.getStylesheets().add("/CSS/Main.css");
```

```
            stage.getIcons().add(new Image("images/PW_Symbol.jpg"));
            stage.setTitle("PSG Event Management System - House
Members");
            stage.setScene(scene);
            stage.setResizable(false);
            stage.sizeToScene();
            stage.show();
        } catch (IOException ex) {

Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }


    @FXML
    private void popOutStandings(ActionEvent event) { //Open House
standings in new Window
        try {
            Stage stage = new Stage(); //New Window
            Parent                        root                        =
FXMLLoader.load(getClass().getResource("/FXML/HouseStandings.fxml"))
; //Load visual file
            stage.getIcons().add(new Image("images/PW_Symbol.jpg"));
//Icon for Window
            stage.setTitle("PSG Event Management System - House
Standings"); //Text for Window
            Scene scene = new Scene(root);
            scene.getStylesheets().add("/CSS/Main.css");
            stage.setScene(scene); //Set visual file to window
            stage.setResizable(false);
            stage.sizeToScene();
            stage.show();
        } catch (IOException ex) {
```

```
Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
}
```

HouseStandingsController

```
package Class;


import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.layout.AnchorPane;


public class HouseStandingsController implements Initializable {


    @FXML
    private AnchorPane base;
    @FXML
    private AnchorPane podium3;
    @FXML
    private Label name3;
    @FXML
    private Label points3;
    @FXML
    private AnchorPane podium1;
    @FXML
    private Label name1;
    @FXML
```

```
private Label points1;
@FXML
private AnchorPane podium2;
@FXML
private Label name2;
@FXML
private Label points2;
@FXML
private AnchorPane podium4;
@FXML
private Label name4;
@FXML
private Label points4;

@Override
public void initialize(URL url, ResourceBundle rb) {
    populateHouseRank();
}

@FXML
public void populateHouseRank() {
    String[][] rank = SQL.houseRank();
    name1.setText(rank[0][0]);
    points1.setText(rank[0][1]);
    podium1.setStyle("-fx-background-color:"+rank[0][2]+";");
    name2.setText(rank[1][0]);
    points2.setText(rank[1][1]);
    podium2.setStyle("-fx-background-color:"+rank[1][2]+";");
    name3.setText(rank[2][0]);
    points3.setText(rank[2][1]);
    podium3.setStyle("-fx-background-color:"+rank[2][2]+";");
    name4.setText(rank[3][0]);
```

```
        points4.setText(rank[3][1]);
        podium4.setStyle("-fx-background-color:"+rank[3][2]+";");
    }

}
```

HouseMembersController

```
package Class;

import com.jfoenix.controls.JFXButton;
import com.jfoenix.controls.JFXListView;
import com.jfoenix.controls.JFXSnackbar;
import java.io.IOException;
import java.net.URL;
import java.sql.ResultSet;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.layout.Pane;
import javafx.stage.Modality;
import javafx.stage.Stage;
```

```
public class HouseMembersController implements Initializable {

    @FXML
    private Pane base;
    @FXML
    private Label house;
    @FXML
    private JFXListView<Pane> listPerson;
    @FXML
    private JFXButton buttonEditPerson;
    @FXML
    private JFXButton buttonAddPerson;
    @FXML
    private JFXButton buttonRefresh;

    private int[] listCellNo = {-1,-1};

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        switch (MainController.accessLevel){
            case 3: //Full Access
                buttonAddPerson.setDisable(false);
                buttonEditPerson.setText("Edit");
                break;
            case 2: //Add/Edit Access
                buttonAddPerson.setDisable(false);
                buttonEditPerson.setText("Edit");
                break;
            case 1: //Edit Only Access
                buttonAddPerson.setDisable(true);
                buttonEditPerson.setText("Edit");
                break;
```

```
            case 0: //View Only Access
                buttonAddPerson.setDisable(true);
                buttonEditPerson.setText("View");
                break;
        }


listPerson.getSelectionModel().selectedItemProperty().addListener((O
bservableValue<? extends Pane> observable, Pane oldValue, Pane
newValue) -> {
            if (newValue != null){
                System.out.println("newValue    ID    =    "    +
newValue.getId()); //Get ID of selection of a cell in the list
                listCellNo[0]                                       =
Integer.parseInt(newValue.getId().replaceFirst("\\.(.*)","")); //Set
local id of currently selected cell
                listCellNo[1]                                       =
Integer.parseInt(newValue.getId().replaceFirst("(.*)\\.","")); //Set
database id of currently selected cell
            }
        });


        //Interaction with Person List
        buttonAddPerson.setOnAction(Event -> {
            launchPersonEditor();
        });


        buttonEditPerson.setOnAction(Event -> {
            if (listCellNo[0] >= 0){
                launchPersonEditor(listCellNo[1]);
            } else {
                JFXSnackbar msg = new JFXSnackbar(base);
                msg.show("Please select entry to edit",4000);
            }
```

```
        });

        buttonRefresh.setOnAction(Event -> {
            populateList(house.getText());
        });
    }

    public void setHouseName(String houseName){
        this.populateList(houseName); //Fill List with data from DB
//                            if    (listPerson.getItems().isEmpty())
listPerson.setVisible(false); //Hide the List if empty
        this.house.setText(houseName); //Set Label text
    }

    public void setHouseColor(String houseColor){

house.setStyle(house.getStyle().replaceAll("#999999",houseColor));
//Set label color
    }

    private void launchPersonEditor() {
        try {
            Stage stage = new Stage(); //New Window
            stage.initModality(Modality.APPLICATION_MODAL);
            FXMLLoader            loader            =            new
FXMLLoader(getClass().getResource("/FXML/PersonRecordEditor.fxml"));
            Parent root = loader.load();
            Scene scene = new Scene(root);
            scene.getStylesheets().add("/CSS/Main.css");
            stage.getIcons().add(new Image("images/PW_Symbol.jpg"));
            stage.setTitle("PSG  Event  Management  System  -  Editing
Person");
            stage.setScene(scene);
```

```
            stage.setResizable(false);
            stage.sizeToScene();
            stage.show();
        } catch (IOException ex) {
Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    private String programFromInt(int program){
        String prog = null;
        switch (program){
            case 3:
                prog = "DYP";
                break;
            case 2:
                prog = "MYP";
                break;
            case 1:
                prog = "PYP";
                break;
        }
        return prog;
    }

    private String accessFromInt(int accessLvl){
        String AccLvl = null;
        switch (accessLvl){
            case 3:
                AccLvl = "Full Access";
                break;
```

```
            case 2:
                AccLvl = "Add + Edit";
                break;
            case 1:
                AccLvl = "Edit Only";
                break;
            case 0:
                AccLvl = "View Only";
                break;
        }
        return AccLvl;
    }

    private void launchPersonEditor(int id) {
        try {
            Stage stage = new Stage(); //New Window
            stage.initModality(Modality.APPLICATION_MODAL);
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/FXML/PersonRecordEditor.fxml"));
            Parent root = loader.load();
            Scene scene = new Scene(root);
            scene.getStylesheets().add("/CSS/Main.css");
            PersonRecordEditorController personRec =
loader.getController();

            ResultSet person = SQL.loadPerson(id);
            person.first();
            personRec.loadPerson(id, person.getString(1),
person.getString(2), person.getString(3), (person.getInt(4)==0),
person.getInt(5), person.getString(6).charAt(0),
programFromInt(person.getInt(7)), accessFromInt(person.getInt(8)),
person.getString(9));

            stage.getIcons().add(new Image("images/PW_Symbol.jpg"));
```

```
            stage.setTitle("PSG Event Management System - Editing
Person");
            stage.setScene(scene);
            stage.setResizable(false);
            stage.sizeToScene();
            stage.show();
        } catch (Exception ex) {

Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }


    private void populateList(String house){
        listPerson.getItems().clear();
        System.out.println("House : "+house);
        try {
            ResultSet person = SQL.personList(house); //Get event
data from DB
            int c = 0;
            while (person.next()){
                FXMLLoader loader = new
FXMLLoader(getClass().getResource("/FXML/CellPerson.fxml"));
                Pane listCell = loader.load();
                CellPersonController cellCon =
loader.getController();
                cellCon.setName(person.getString(2));
                cellCon.setGrade("Grade: "+(person.getInt(3)==0 ?
"KG" : person.getString(3)) +" "+person.getString(4));
                cellCon.setPoints(person.getInt(5));
                listCell.setId(c+"."+person.getString(1)); //Set ID
as "localID.databaseID"
                c++;
                listPerson.getItems().add(listCell); //Add Cell to
list
```

```
            }
        } catch (Exception ex) {
Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    @FXML
    private void buttonBack(ActionEvent event) {
        Stage stage = (Stage) base.getScene().getWindow(); //Get
target window
        stage.close(); //Close target window
    }

}
```

HamburgerDrawerController

```
package Class;

import com.jfoenix.controls.JFXButton;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;

public class HamburgerDrawerController implements Initializable {

    @FXML
    JFXButton buttonClose;
    @FXML
    JFXButton buttonLogout;
```

```
    @FXML
    JFXButton buttonSettings;

    @Override
    public void initialize(URL url, ResourceBundle rb) {

    }
}
```

SQL

```
package Class;

import com.google.common.io.Files;
import java.awt.Desktop;
import java.io.File;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.sql.*;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Platform;

public class SQL {
    static Connection conn;
    static boolean firstRun;
    static MainController mc = new MainController();

    public SQL(){}
```

```
    public SQL(String dbName) throws SQLException{
        try {
            String url = null;
            String user = null;
            String pass = null;
            List <String> list = Files.readLines(new File("./Settings.txt"), StandardCharsets.UTF_8);
            Iterator<String> iterator = list.iterator();
            int i = 0;
            while (iterator.hasNext()){
                i++;
                String ele = iterator.next();
                switch (i) {
                    case 8:
                        firstRun = ele.replaceFirst("(.*)=", "").equals("Y");
                        break;
                    case 10:
                        url = ele.replaceFirst("(.*)=", "");
                        break;
                    case 11:
                        user = ele.replaceFirst("(.*)=", "");
                        break;
                    case 12:
                        pass = ele.replaceFirst("(.*)=", "");
                        break;
                }
            }
            if (!firstRun){
                conn = DriverManager.getConnection("jdbc:mysql://" + url + dbName, user, pass);
            } else {
```

```
                Desktop.getDesktop().open(new
File("./Settings.txt"));
                System.exit(0);
            }
        } catch (IOException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }


    public static String login(String user, String pass){
        String success = "false";
        try {
            ResultSet                       rs                      =
conn.createStatement().executeQuery("SELECT     name,     pass_hash,
permissions FROM individual WHERE permissions != 0;");
            while (rs.next()) {
                if (rs.getString(1).equals(user)){
                    if (rs.getString(2).equals(pass)){
                        success = "match"+rs.getString(3);
                    }
                }
            }
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if      (ex.toString().contains("Communications      link
failure")){
                mc.sqlDisconnect();
                System.out.println("\n\n SQL DISCONNECTED \n\n");
            }
        }
        return success;
    }
```

```
    public static void close(){
        try {
            conn.close();
            Platform.exit(); //Exit application
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            Platform.exit();
        }
    }


    public static String[][] houseRank(){
        String[][] rank = new String[4][3];
        try {
            ResultSet                       rs                       =
conn.createStatement().executeQuery("SELECT name, points, color FROM
house ORDER BY points DESC;");
            int c = 0;
            while (rs.next()) {
                rank[c][0] = rs.getString(1);
                rank[c][1] = rs.getString(2);
                rank[c][2] = rs.getString(3);
                c++;
            }
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if      (ex.toString().contains("Communications      link
failure")){
                mc.sqlDisconnect();
            }
        }
```

```
        return rank;
    }


    public static ResultSet eventList() {
        ResultSet rs = null;
        try {
            rs  =  conn.createStatement().executeQuery("SELECT  id,
name, grades, datetime FROM events ORDER BY datetime DESC;");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if     (ex.toString().contains("Communications     link
failure")){
                mc.sqlDisconnect();
            }
        }
        return rs;
    }


    public static ResultSet personList(String house) {
        ResultSet rs = null;
        try {
            rs  =  conn.createStatement().executeQuery("SELECT  id,
name, grade, section, points FROM individual WHERE house = '"+ house
+"' ORDER BY is_student ASC, points DESC;");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if     (ex.toString().contains("Communications     link
failure")){
                mc.sqlDisconnect();
            }
        }
        return rs;
```

```
    }


    public static ResultSet participantList(){
        ResultSet rs = null;
        try {
            rs  =  conn.createStatement().executeQuery("SELECT  id,
name, house FROM individual WHERE is_student = 1 ORDER BY name;");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if     (ex.toString().contains("Communications     link
failure")){
                mc.sqlDisconnect();
            }
        }
        return rs;
    }


    public static ResultSet participantSelection(int id){
        ResultSet rs = null;
        try {
            rs     =     conn.createStatement().executeQuery("SELECT
participant FROM participant WHERE event = "+ id +";");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if     (ex.toString().contains("Communications     link
failure")){
                mc.sqlDisconnect();
            }
        }
        return rs;
    }
```

```
    public static void addParticipants(int event, int participant[]){
        try {
            conn.createStatement().executeUpdate("DELETE        FROM
participant WHERE event = "+event+";");
            PreparedStatement  stmt  =  conn.prepareStatement("INSERT
INTO participant (event, participant) VALUES ("+event+", ?);");
            for (int element : participant){
                stmt.setInt(1, element);
                stmt.executeUpdate();
            }
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if     (ex.toString().contains("Communications     link
failure")){
                mc.sqlDisconnect();
            }
        }
    }

//    EVENTS
    private static int houseInt(String house){
        int hInt = 0;
        switch (house){
            case "Air":
                hInt = 1;
                break;
            case "Earth":
                hInt = 2;
                break;
            case "Fire":
                hInt = 3;
                break;
```

```
            case "Water":
                hInt = 4;
                break;
        }
        return hInt;
    }


    public static void addEvent(String name, String datetime, String venue, int grades, int points1, int points2, int points3, int points4, String win1, String win2, String win3, String win4, String description){
        int winner1 = 0, winner2 = 0, winner3 = 0, winner4 = 0;
        if (win1!=null)
            winner1 = houseInt(win1);
        if (win2!=null)
            winner2 = houseInt(win2);
        if (win3!=null)
            winner3 = houseInt(win3);
        if (win4!=null)
            winner4 = houseInt(win4);
        try {
            conn.createStatement().executeUpdate("INSERT INTO events (name, datetime, venue, grades, points1, points2, points3, points4, win1, win2, win3, win4, description) VALUES ('"+ name +"','"+ datetime +"','"+ venue +"',"+ grades +","+ points1 +","+ points2 +","+ points3 +","+ points4 +","+ winner1 +","+ winner2 +","+ winner3 +","+ winner4 +",'"+ description +"');");
            if (!(winner1==0||winner2==0||winner3==0||winner4==0)){
                ResultSet rs = conn.createStatement().executeQuery("SELECT id FROM events WHERE name = '"+ name +"' AND datetime = '"+ datetime +"';");
                rs.first();
                updatePoints(rs.getInt(1), winner1, points1, winner2, points2, winner3, points3, winner4, points4);
            }
        } catch (SQLException ex) {
```

```
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if      (ex.toString().contains("Communications      link
failure")){
                mc.sqlDisconnect();
            }
        }
    }

    public static void updateEvent(int id, String name, String
datetime, String venue, int grades, int points1, int points2, int
points3, int points4, String win1, String win2, String win3, String
win4, String description){
        int winner1 = 0, winner2 = 0, winner3 = 0, winner4 = 0;
        if (win1!=null)
            winner1 = houseInt(win1);
        if (win2!=null)
            winner2 = houseInt(win2);
        if (win3!=null)
            winner3 = houseInt(win3);
        if (win4!=null)
            winner4 = houseInt(win4);
        try {
            if (!(winner1==0||winner2==0||winner3==0||winner4==0))
                updatePoints(id, winner1, points1, winner2, points2,
winner3, points3, winner4, points4);
            conn.createStatement().executeUpdate("UPDATE events SET
name = '"+ name +"',datetime = '"+ datetime +"',venue = '"+ venue
+"',grades = "+ grades +",points1 = "+ points1 +",points2 = "+ points2
+",points3 = "+ points3 +",points4 = "+ points4 +",win1 = '"+ winner1
+"',win2 = '"+ winner2 +"',win3 = '"+ winner3 +"',win4 = '"+ winner4
+"',description = '"+ description +"' WHERE id = "+ id +";");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
```

```
            if      (ex.toString().contains("Communications      link
failure")){
                mc.sqlDisconnect();
            }
        }
    }


    public static void deleteEvent(int id){
        try {
            conn.createStatement().executeUpdate("DELETE FROM events
WHERE id = "+ id +";");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if      (ex.toString().contains("Communications      link
failure")){
                mc.sqlDisconnect();
            }
        }
    }


    public static ResultSet loadEvent(int id){
        ResultSet rs = null;
        try {
            rs  =  conn.createStatement().executeQuery("SELECT  name,
datetime, venue, grades, points1, points2, points3, points4, win1,
win2, win3, description FROM events WHERE id = "+ id +";");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if      (ex.toString().contains("Communications      link
failure")){
                mc.sqlDisconnect();
            }
```

```
        }
        return rs;
    }


    private static void updatePoints(int EvID, int win1, int point1, int win2, int point2, int win3, int point3, int win4, int point4){
        try {
            //House
            ResultSet house = conn.createStatement().executeQuery("SELECT id, points FROM house;");
            while (house.next()){
                if (house.getInt(1) == win1){
                    int val = house.getInt(1)+point1;
                    conn.createStatement().executeUpdate("UPDATE house SET points = "+ val +" WHERE id = "+ win1 +";");
                }
                if (house.getInt(1) == win2){
                    int val = house.getInt(1)+point2;
                    conn.createStatement().executeUpdate("UPDATE house SET points = "+ val +" WHERE id = "+ win2 +";");
                }
                if (house.getInt(1) == win3){
                    int val = house.getInt(1)+point3;
                    conn.createStatement().executeUpdate("UPDATE house SET points = "+ val +" WHERE id = "+ win3 +";");
                }
                if (house.getInt(1) == win4){
                    int val = house.getInt(1)+point4;
                    conn.createStatement().executeUpdate("UPDATE house SET points = "+ val +" WHERE id = "+ win4 +";");
                }
            }

            //Participants
```

```
            ResultSet participants = participantSelection(EvID);
            while (participants.next()){
                ResultSet person = conn.createStatement().executeQuery("SELECT points, house FROM individual WHERE id = "+ participants.getInt(1) +";");
                person.next();
                int houseNum = houseInt(person.getString(2));
                int val = 0;
                if (houseNum == win1){
                    val = person.getInt(1)+point1;
                }
                if (houseNum == win2){
                    val = person.getInt(1)+point2;
                }
                if (houseNum == win3){
                    val = person.getInt(1)+point3;
                }
                if (houseNum == win4){
                    val = person.getInt(1)+point4;
                }
                conn.createStatement().executeUpdate("UPDATE individual SET points = "+ val +" WHERE id = "+ participants.getInt(1) +";");
            }
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE, null, ex);
            if (ex.toString().contains("Communications link failure")){
                mc.sqlDisconnect();
            }
        }
    }
```

```
    public static ResultSet loadParticipants(int id){
        ResultSet rs = null;
        try {
            rs = conn.createStatement().executeQuery("SELECT * FROM
participant WHERE event = "+ id +";");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if     (ex.toString().contains("Communications     link
failure")){
                mc.sqlDisconnect();
            }
        }
        return rs;
    }


//    PERSON


    public static void addPerson(String name, String designation,
String house, boolean isFaculty, int grade, char section, int program,
int permissions, String pass){
        try {
            conn.createStatement().executeUpdate("INSERT        INTO
individual (name, designation, house, is_student, grade, section,
program, permissions, pass_hash) VALUES ('"+ name +"','"+ designation
+"','"+ house +"',"+ (isFaculty ? 0 : 1) +","+ grade +",'"+ section
+"',"+ program +","+ permissions +",'"+ pass +"');");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if     (ex.toString().contains("Communications     link
failure")){
                mc.sqlDisconnect();
            }
        }
    }
```

```
    public static void updatePerson(int id, String name, String designation, String house, boolean isFaculty, int grade, char section, int program, int permissions, String pass){
        try {
            conn.createStatement().executeUpdate("UPDATE individual SET name = '"+ name +"',designation = '"+ designation +"',house = '"+ house +"',is_student = "+ (isFaculty ? 0 : 1) +",grade = "+ grade +",section = '"+ section +"',program = "+ program +",permissions = "+ permissions +",pass_hash = '"+ pass +"' WHERE id = "+ id +";");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE, null, ex);
            if (ex.toString().contains("Communications link failure")){
                mc.sqlDisconnect();
            }
        }
    }

    public static void deletePerson(int id){
        try {
            conn.createStatement().executeUpdate("DELETE FROM individual WHERE id = "+ id +";");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE, null, ex);
            if (ex.toString().contains("Communications link failure")){
                mc.sqlDisconnect();
            }
        }
    }

    public static ResultSet loadPerson(int id){
        ResultSet rs = null;
```

```
        try {
            rs  =  conn.createStatement().executeQuery("SELECT  name,
designation, house, is_student, grade, section, program, permissions,
points FROM individual WHERE id = "+ id +";");
        } catch (SQLException ex) {
            Logger.getLogger(SQL.class.getName()).log(Level.SEVERE,
null, ex);
            if      (ex.toString().contains("Communications      link
failure")){
                mc.sqlDisconnect();
            }
        }
        return rs;
    }


}
```

PersonRecordEditorController

```
package Class;


import com.google.common.hash.Hashing;
import com.jfoenix.controls.JFXButton;
import com.jfoenix.controls.JFXComboBox;
import com.jfoenix.controls.JFXPasswordField;
import com.jfoenix.controls.JFXSnackbar;
import com.jfoenix.controls.JFXTextField;
import com.jfoenix.controls.JFXToggleButton;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.util.ResourceBundle;
import java.util.regex.Pattern;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
```

```
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class PersonRecordEditorController implements Initializable {

    @FXML
    private Pane base;
    @FXML
    private JFXTextField name;
    @FXML
    private JFXTextField designation;
    @FXML
    private JFXToggleButton isFaculty;
    @FXML
    private JFXComboBox<String> grade;
    @FXML
    private JFXTextField section;
    @FXML
    private JFXComboBox<String> program;
    @FXML
    private JFXComboBox<String> house;
    @FXML
    private JFXComboBox<String> accessLvl;
    @FXML
    private JFXPasswordField password;
    @FXML
    private Label points;
```

```
    @FXML
    private JFXButton buttonDelete;
    @FXML
    private JFXButton buttonDone;

    boolean loaded = false;
    int loadID;

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        base.setOnMouseClicked((MouseEvent evt) -> { //If you click
on the login pane
            base.requestFocus(); //it will draw focus from any other
element
        });

        if (MainController.accessLevel == 0){
            buttonDone.setVisible(false);
            buttonDelete.setDisable(true);
            name.setMouseTransparent(true);
            designation.setMouseTransparent(true);
            isFaculty.setMouseTransparent(true);
            grade.setMouseTransparent(true);
            section.setMouseTransparent(true);
            program.setMouseTransparent(true);
            house.setMouseTransparent(true);
            accessLvl.setMouseTransparent(true);
            password.setMouseTransparent(true);
        }

        grade.getItems().add("KG");
        grade.getItems().add("1");
```

```
        grade.getItems().add("2");
        grade.getItems().add("3");
        grade.getItems().add("4");
        grade.getItems().add("5");
        grade.getItems().add("6");
        grade.getItems().add("7");
        grade.getItems().add("8");
        grade.getItems().add("9");
        grade.getItems().add("10");
        grade.getItems().add("11");
        grade.getItems().add("12");

        program.getItems().add("PYP");
        program.getItems().add("MYP");
        program.getItems().add("DYP");

        house.getItems().add("Air");
        house.getItems().add("Earth");
        house.getItems().add("Fire");
        house.getItems().add("Water");

        accessLvl.getItems().add("View Only");
        accessLvl.getItems().add("Edit Only");
        accessLvl.getItems().add("Add + Edit");
        accessLvl.getItems().add("Full Access");

        isFaculty.selectedProperty().addListener((ObservableValue<?
extends Boolean> observable, Boolean oldValue, Boolean newValue) -> {
            if (newValue){
                grade.setVisible(false);
                section.setVisible(false);
                program.setVisible(true);
```

```
            } else {
                grade.setVisible(true);
                section.setVisible(true);
                program.setVisible(false);

            }
        });

        accessLvl.focusedProperty().addListener((ObservableValue<? extends Boolean> observable, Boolean oldValue, Boolean newValue) -> {
            if (oldValue && accessLvl.getValue().equals("View Only")){
                password.setDisable(true);
            } else if (oldValue && !(accessLvl.getValue().equals("View Only") || accessLvl.getValue().equals(""))){
                password.setDisable(false);
            }
        });

        buttonDelete.setOnAction(click ->{
            base.requestFocus();
            JFXSnackbar msg = new JFXSnackbar(base);
            msg.show("Are you sure that you want to delete?", "CONFIRM", 6000, evt -> {
                SQL.deletePerson(loadID);
                //Close Window
                Stage stage = (Stage) base.getScene().getWindow();
                stage.close();
            });
        });
    }
```

```
    void loadPerson(int id, String name, String designation, String house, boolean isFaculty, int grade, char section, String program, String accessLvl, String points){
        this.name.setText(name);
        this.designation.setText(designation);
        this.house.setValue(house);
        this.isFaculty.selectedProperty().setValue(isFaculty);
        this.grade.setValue(String.valueOf(grade));
        this.section.setText(String.valueOf(section));
        this.program.setValue(program);
        this.accessLvl.setValue(accessLvl);
        this.points.setText(points);

        loaded = true;
        loadID = id;
        buttonDelete.setVisible(true);
    }

    @FXML
    private void buttonDone(ActionEvent event) {
        int err = 0; //Number of errors in form
        if (name.getText().isEmpty()){
            err++;
            name.setUnFocusColor(Color.RED);
        } else {name.setUnFocusColor(Color.web("#01579b"));}
        if (isFaculty.isSelected()){
            if (program.getValue() == null){
                err++;
                program.setStyle("-fx-border-color:red");
            } else {program.setStyle("");}
            grade.setStyle("");
            section.setUnFocusColor(Color.web("#01579b"));
```

```
        } else {
            if (section.getLength() != 1 || !(Pattern.matches("^[a-
zA-Z]+$", section.getText()))){
            err++;
            section.setUnFocusColor(Color.RED);
            } else {section.setUnFocusColor(Color.web("#01579b"));}
            if (grade.getValue() == null){
                err++;
                grade.setStyle("-fx-border-color:red");
            } else {grade.setStyle("");}
            program.setStyle("");
        }
        if (house.getValue() == null){
            err++;
            house.setStyle("-fx-border-color:red");
        } else {house.setStyle("");}
        if (accessLvl.getValue() == null){
            err++;
            accessLvl.setStyle("-fx-border-color:red");
        } else {accessLvl.setStyle("");}
        if    ((!accessLvl.getValue().equals("View    Only"))    &&
password.getText().length() <10){
            err++;
            password.setUnFocusColor(Color.RED);
            JFXSnackbar msg = new JFXSnackbar(base);
            msg.show("Password needs to be at least 10 characters
long", 4000);
        } else {password.setUnFocusColor(Color.web("#01579b"));}

        if (err == 0){
            if (isFaculty.isSelected() == true){
                grade.setValue(null);
                section.setText(null);
```

```
            } else {
                program.setValue(null);
            }
            int prog = 0;
            switch (program.getValue()){
                case "DYP":
                    prog = 3;
                    break;
                case "MYP":
                    prog = 2;
                    break;
                case "PYP":
                    prog = 1;
                    break;
            }
            int perm = 0;
            switch (accessLvl.getValue()){
                case "Full Access":
                    perm = 3;
                    break;
                case "Add + Edit":
                    perm = 2;
                    break;
                case "Edit Only":
                    perm = 1;
                    break;
            }
            String hashed = null;
            if (!password.getText().isEmpty())
                hashed                                                    =
Hashing.sha256().hashString(password.getText(),
StandardCharsets.UTF_8).toString();  //Convert  password  input  into
cypertext
```

```
            int gradeVal = -1;
            if (grade.getValue() != null){

Integer.parseInt(grade.getValue().equals("KG")?"0":grade.getValue())
;
            }
            char sectionVal = 0;
            if (section.getText() != null){
                sectionVal = section.getText().charAt(0);
            }


            if (loaded == false) {
                SQL.addPerson(name.getText(), designation.getText(),
house.getValue(), isFaculty.selectedProperty().getValue(), gradeVal,
sectionVal, prog, perm, hashed);
            }else{
                SQL.updatePerson(loadID,             name.getText(),
designation.getText(),                              house.getValue(),
isFaculty.selectedProperty().getValue(), gradeVal, sectionVal, prog,
perm, hashed);
            }
            Stage stage = (Stage) base.getScene().getWindow(); //Get
target window
            stage.close(); //Close target window
        }
    }


    @FXML
    private void buttonBack(ActionEvent event) {
        Stage  stage  =  (Stage)  base.getScene().getWindow();  //Get
target window
        stage.close(); //Close target window
    }
```

```
}
```

EventRecordEditorController

```
package Class;

import com.jfoenix.controls.JFXButton;
import com.jfoenix.controls.JFXComboBox;
import com.jfoenix.controls.JFXDatePicker;
import com.jfoenix.controls.JFXSnackbar;
import com.jfoenix.controls.JFXTextArea;
import com.jfoenix.controls.JFXTextField;
import com.jfoenix.controls.JFXToggleNode;
import com.jfoenix.validation.NumberValidator;
import java.io.IOException;
import java.net.URL;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
```

```
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class EventRecordEditorController implements Initializable {

    @FXML
    private Pane base;
    @FXML
    private JFXTextField name;
    @FXML
    private JFXDatePicker time;
    @FXML
    private JFXDatePicker date;
    @FXML
    private JFXTextField venue;
    @FXML
    private JFXTextArea description;
    @FXML
    private Label grLbl;
    @FXML
    private JFXTextField participants;
    @FXML
    private JFXTextField points1;
    @FXML
    private JFXTextField points2;
    @FXML
    private JFXTextField points3;
    @FXML
    private JFXTextField points4;
```

```
@FXML
private JFXComboBox<String> win1;
@FXML
private JFXComboBox<String> win2;
@FXML
private JFXComboBox<String> win3;
@FXML
private JFXButton buttonDelete;
@FXML
private JFXButton buttonDone;

private final JFXToggleNode gr0 = new JFXToggleNode();
private final JFXToggleNode gr1 = new JFXToggleNode();
private final JFXToggleNode gr2 = new JFXToggleNode();
private final JFXToggleNode gr3 = new JFXToggleNode();
private final JFXToggleNode gr4 = new JFXToggleNode();
private final JFXToggleNode gr5 = new JFXToggleNode();
private final JFXToggleNode gr6 = new JFXToggleNode();
private final JFXToggleNode gr7 = new JFXToggleNode();
private final JFXToggleNode gr8 = new JFXToggleNode();
private final JFXToggleNode gr9 = new JFXToggleNode();
private final JFXToggleNode gr10 = new JFXToggleNode();
private final JFXToggleNode gr11 = new JFXToggleNode();
private final JFXToggleNode gr12 = new JFXToggleNode();

int grades = 1;
boolean loaded = false;
int loadID;

@Override
public void initialize(URL url, ResourceBundle rb) {
```

```
        base.setOnMouseClicked((MouseEvent evt) -> { //If you click
on the login pane
            base.requestFocus(); //it will draw focus from any other
element
        });


        if (MainController.accessLevel == 0){
            buttonDone.setVisible(false);
            buttonDelete.setDisable(true);
            name.setMouseTransparent(true);
            time.setMouseTransparent(true);
            date.setMouseTransparent(true);
            venue.setMouseTransparent(true);
            description.setMouseTransparent(true);
            participants.setMouseTransparent(true);
            points1.setMouseTransparent(true);
            points2.setMouseTransparent(true);
            points3.setMouseTransparent(true);
            points4.setMouseTransparent(true);
            win1.setMouseTransparent(true);
            win2.setMouseTransparent(true);
            win3.setMouseTransparent(true);
        }


        setToggle(gr0, 33, "KG");
        setToggle(gr1, 59, "1");
        setToggle(gr2, 85, "2");
        setToggle(gr3, 111, "3");
        setToggle(gr4, 137, "4");
        setToggle(gr5, 163, "5");
        setToggle(gr6, 189, "6");
        setToggle(gr7, 215, "7");
```

```
        setToggle(gr8, 241, "8");
        setToggle(gr9, 267, "9");
        setToggle(gr10, 293, "10");
        setToggle(gr11, 319, "11");
        setToggle(gr12, 345, "12");

base.getChildren().addAll(gr0,gr1,gr2,gr3,gr4,gr5,gr6,gr7,gr8,gr9,gr
10,gr11,gr12);

        NumberValidator valnum1 = new NumberValidator();
        participants.setValidators(valnum1);
        NumberValidator valnum2 = new NumberValidator();
        points1.setValidators(valnum2);
        NumberValidator valnum3 = new NumberValidator();
        points2.setValidators(valnum3);
        NumberValidator valnum4 = new NumberValidator();
        points3.setValidators(valnum4);
        NumberValidator valnum5 = new NumberValidator();
        points4.setValidators(valnum5);

        win1.getItems().add("Air");
        win1.getItems().add("Earth");
        win1.getItems().add("Fire");
        win1.getItems().add("Water");
        win2.getItems().add("Air");
        win2.getItems().add("Earth");
        win2.getItems().add("Fire");
        win2.getItems().add("Water");
        win3.getItems().add("Air");
        win3.getItems().add("Earth");
        win3.getItems().add("Fire");
        win3.getItems().add("Water");
```

```
        buttonDelete.setOnAction(click ->{
            base.requestFocus();
            JFXSnackbar msg = new JFXSnackbar(base);
            msg.show("Are you sure that you want to delete?", "CONFIRM", 6000, evt -> {
                SQL.deleteEvent(loadID);
                //Close Window
                Stage stage = (Stage) base.getScene().getWindow();
                stage.close();
            });
        });
    }

    private void setToggle(JFXToggleNode node, double x, String lbl){ //Set toggle postion and text
        node.setLayoutX(x); //Set location X
        node.setLayoutY(326); //Set location Y
        node.setPrefSize(22, 22); //Set Size
        node.setMinSize(22, 22);
        node.setGraphic(new Label(lbl)); //Add label for text
        node.setSelectedColor(Color.web("#29B6F6"));  //Set  default color
        node.setUnSelectedColor(Color.WHITE); //Set activated color
    }

    void loadEvent(int id, String name, String datetime, String venue, int grade, int participants, int point1, int point2, int point3, int point4, String winner1, String winner2, String winner3, String description){
        // Load an event to be edited
        this.name.setText(name);
        this.venue.setText(venue);
        this.description.setText(description);
```

```
        // Converts datetime string to date and time
        this.date.setValue(LocalDate.parse(datetime.substring(0,
10),DateTimeFormatter.ISO_DATE));
        this.time.setTime(LocalTime.parse(datetime.substring(11,
16),DateTimeFormatter.ISO_TIME));
        // If grade is divisible by corresoinding prime number, then
select node
        gr0.setSelected(grade%2==0);
        gr1.setSelected(grade%3==0);
        gr2.setSelected(grade%5==0);
        gr3.setSelected(grade%7==0);
        gr4.setSelected(grade%11==0);
        gr5.setSelected(grade%13==0);
        gr6.setSelected(grade%17==0);
        gr7.setSelected(grade%19==0);
        gr8.setSelected(grade%23==0);
        gr9.setSelected(grade%29==0);
        gr10.setSelected(grade%31==0);
        gr11.setSelected(grade%37==0);
        gr12.setSelected(grade%41==0);
        this.participants.setText(String.valueOf(participants/4));
        this.points1.setText(String.valueOf(point1));
        this.points2.setText(String.valueOf(point2));
        this.points3.setText(String.valueOf(point3));
        this.points4.setText(String.valueOf(point4));
        // Select option with text winner
        this.win1.getSelectionModel().select(winner1);
        this.win2.getSelectionModel().select(winner2);
        this.win3.getSelectionModel().select(winner3);

        loaded = true;
        loadID = id;
        buttonDelete.setVisible(true);
```

```
    }


    @FXML
    private void chooseParticipants(ActionEvent event) {
        try {
            Stage stage = new Stage();
            stage.initModality(Modality.APPLICATION_MODAL);
//Disable all other windows of application
            ParticipantChooseController.setEventID(loadID);

ParticipantChooseController.setPartNum(Integer.parseInt(participants
.getText()));
            FXMLLoader            loader            =            new
FXMLLoader(getClass().getResource("/FXML/ParticipantChoose.fxml"));
            Parent root = loader.load();
            Scene scene = new Scene(root);
            scene.getStylesheets().add("/CSS/Main.css");
            stage.getIcons().add(new Image("images/PW_Symbol.jpg"));
            stage.setTitle("PSG  Event  Management  System  -  Choose
Participants");
            stage.setScene(scene);
            stage.setResizable(false);
            stage.sizeToScene();
            stage.show();
        } catch (IOException ex) {

Logger.getLogger(MainController.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }


    @FXML
    private void buttonDone(ActionEvent event) {
        int err = 0; //Number of errors in form
```

```
        if (name.getText().isEmpty()){
            err++;
            name.setUnFocusColor(Color.RED);
        } else {name.setUnFocusColor(Color.web("#01579b"));}
        if (venue.getText().isEmpty()){
            err++;
            venue.setUnFocusColor(Color.RED);
        } else {venue.setUnFocusColor(Color.web("#01579b"));}
        if (time.getTime() == null){
            err++;
            time.setStyle("-fx-border-color:red");
        } else {time.setStyle("");}
        if (date.getValue() == null){
            err++;
            date.setStyle("-fx-border-color:red");
        } else {date.setStyle("");}
        if (!participants.validate()){
            err++;
            participants.setUnFocusColor(Color.RED);
        } else {participants.setUnFocusColor(Color.web("#01579b"));}
        if (gr0.isSelected() || gr1.isSelected() || gr2.isSelected()
||  gr3.isSelected()  ||  gr4.isSelected()  ||  gr5.isSelected()  ||
gr6.isSelected()   ||   gr7.isSelected()   ||   gr8.isSelected()   ||
gr9.isSelected()  ||  gr10.isSelected()  ||  gr11.isSelected()  ||
gr12.isSelected()){
            grLbl.setTextFill(Color.BLACK);
        } else {
            grLbl.setTextFill(Color.RED);
            err++;
        }


        if (err == 0){
            //Datetime
```

```
            String        datetime        =        date.getValue()+"
"+time.getTime()+":00";


            // Grades involved in event. Each grade has a unique prime
number assigned. While checking

            grades = 1;

            grades = gr0.isSelected() ? grades*2 : grades; //If gr0
is selected then grades = grades*2, else grades = 1

            grades = gr1.isSelected() ? grades*3 : grades;

            grades = gr2.isSelected() ? grades*5 : grades;

            grades = gr3.isSelected() ? grades*7 : grades;

            grades = gr4.isSelected() ? grades*11 : grades;

            grades = gr5.isSelected() ? grades*13 : grades;

            grades = gr6.isSelected() ? grades*17 : grades;

            grades = gr7.isSelected() ? grades*19 : grades;

            grades = gr8.isSelected() ? grades*23 : grades;

            grades = gr9.isSelected() ? grades*29 : grades;

            grades = gr10.isSelected() ? grades*31 : grades;

            grades = gr11.isSelected() ? grades*37 : grades;

            grades = gr12.isSelected() ? grades*41 : grades;


            String win4 = null;

            if (!(win1.getValue() == null || win2.getValue() == null
|| win3.getValue() == null)) {

                win4 = "AirEarthFireWater";

                win4       =       win4.replaceFirst(win1.getValue(),
"").replaceFirst(win2.getValue(),  "").replaceFirst(win3.getValue(),
"");

            }


            if (loaded == false)

                SQL.addEvent(name.getText(),                datetime,
venue.getText(),     grades,     Integer.parseInt(points1.getText()),
Integer.parseInt(points2.getText()),
Integer.parseInt(points3.getText()),
```

```
Integer.parseInt(points4.getText()),                win1.getValue(),
win2.getValue(), win3.getValue(), win4, description.getText());

            else

                SQL.updateEvent(loadID   ,name.getText(),   datetime,
venue.getText(),     grades,     Integer.parseInt(points1.getText()),
Integer.parseInt(points2.getText()),
Integer.parseInt(points3.getText()),
Integer.parseInt(points4.getText()),                win1.getValue(),
win2.getValue(), win3.getValue(), win4, description.getText());

            Stage stage = (Stage) base.getScene().getWindow(); //Get
target window

            stage.close(); //Close target window

        }

    }


    @FXML

    private void buttonBack(ActionEvent event) {

        Stage  stage  =  (Stage)  base.getScene().getWindow();  //Get
target window

        stage.close(); //Close target window

    }

}
```

ParticipantChooserController

```
package Class;


import com.jfoenix.controls.JFXCheckBox;

import com.jfoenix.controls.JFXListView;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;
```

```
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class ParticipantChooseController implements Initializable {

    @FXML
    private JFXListView<JFXCheckBox> list;
    @FXML
    private Label label;

    private static int EventID;
    private static int partNum;

    public static void setEventID(int EventID) {
        ParticipantChooseController.EventID = EventID;
    }

    public static void setPartNum(int partNum) {
        ParticipantChooseController.partNum = partNum;
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        try {
            ResultSet data = SQL.participantList();
            int c = 0;
            while (data.next()){
```

```
                JFXCheckBox                  cell                  =
makeCell(data.getString(2),data.getString(3));
                cell.setId(c+"."+data.getString(1));           //ID=
'local'.'personid'
                c++;
                list.getItems().add(cell);
            }
            ResultSet selection = SQL.participantSelection(EventID);
            while (selection.next()){
                for (JFXCheckBox cell : list.getItems()){
                    if
(Integer.parseInt(cell.getId().replaceFirst("(.*)\\.",""))        ==
selection.getInt(1)){
                        cell.setSelected(true);
                    }
                }
            }

        } catch (SQLException ex) {

Logger.getLogger(ParticipantChooseController.class.getName()).log(Le
vel.SEVERE, null, ex);
        }
    }

    private JFXCheckBox makeCell(String name, String house) {
        JFXCheckBox cell = new JFXCheckBox(name);
        cell.setPrefSize(300, 20);
        switch (house){
            case "Air":
                cell.setCheckedColor(Color.web("#29B6F6"));
                cell.setTextFill(Color.web("#0996D6"));
                break;
            case "Earth":
```

```
                cell.setCheckedColor(Color.web("#2E7D32"));
                cell.setTextFill(Color.web("#0E5D12"));
                break;
            case "Fire":
                cell.setCheckedColor(Color.web("#C62828"));
                cell.setTextFill(Color.web("#A60808"));
                break;
            case "Water":
                cell.setCheckedColor(Color.web("#283593"));
                cell.setTextFill(Color.web("#081573"));
                break;
        }
        return cell;
    }

    @FXML
    void buttonDone(ActionEvent event) {
        int selected = 0;
        for (JFXCheckBox cell : list.getItems()) {
            if (cell.isSelected()){
                selected++;
            }
        }
        if (selected/4 == partNum){
            int[] participants = new int[selected];
            int c = 0;
            for (JFXCheckBox cell : list.getItems()) {
                if (cell.isSelected()){
                    participants[c]                                    =
Integer.parseInt(cell.getId().replaceFirst("(.*)\\.","").replaceFirs
t("\\.(.*)", ""));
                    c++;
```

```
                }
            }
            SQL.addParticipants(EventID, participants);
            Stage stage = (Stage) list.getScene().getWindow(); //Get
target window
            stage.close(); //Close target window
        }else{
            label.setTextFill(Color.RED);
        }
    }

    @FXML
    private void buttonBack(ActionEvent event) {
        Stage stage = (Stage) list.getScene().getWindow(); //Get
target window
        stage.close(); //Close target window
    }

}
```

CellPersonController

```
package Class;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;

public class CellPersonController implements Initializable {

    @FXML
```

```
    private Label name;
    @FXML
    private Label grade;
    @FXML
    private Label points;

    @Override
    public void initialize(URL url, ResourceBundle rb) {
    }

    public void setGrade(String grade){
        this.grade.setText(grade);
    }

    public void setName(String name){
        this.name.setText(name);
    }

    public void setPoints(int points){
        this.points.setText(String.valueOf(points));
    }
}
```

CellEventController

```
package Class;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
```

```
public class CellEventController implements Initializable {

    @FXML
    private Label name;
    @FXML
    private Label grades;
    @FXML
    private Label time;

    @Override
    public void initialize(URL url, ResourceBundle rb) {
    }

    public void setEventGrades(int grades){
        String gradeTxt = "Grades:";
        gradeTxt = (grades%2==0) ? gradeTxt+" KG," : gradeTxt;
        gradeTxt = (grades%3==0) ? gradeTxt+" 1," : gradeTxt;
        gradeTxt = (grades%5==0) ? gradeTxt+" 2," : gradeTxt;
        gradeTxt = (grades%7==0) ? gradeTxt+" 3," : gradeTxt;
        gradeTxt = (grades%11==0) ? gradeTxt+" 4," : gradeTxt;
        gradeTxt = (grades%13==0) ? gradeTxt+" 5," : gradeTxt;
        gradeTxt = (grades%17==0) ? gradeTxt+" 6," : gradeTxt;
        gradeTxt = (grades%19==0) ? gradeTxt+" 7," : gradeTxt;
        gradeTxt = (grades%23==0) ? gradeTxt+" 8," : gradeTxt;
        gradeTxt = (grades%29==0) ? gradeTxt+" 9," : gradeTxt;
        gradeTxt = (grades%31==0) ? gradeTxt+" 10," : gradeTxt;
        gradeTxt = (grades%37==0) ? gradeTxt+" 11," : gradeTxt;
        gradeTxt = (grades%41==0) ? gradeTxt+" 12," : gradeTxt;

        setEventGrades(gradeTxt.substring(0,      gradeTxt.length()-
1)+".");
```

```
    }

    private void setEventGrades(String txt){ //Polymorphism
Overloading
        this.grades.setText(txt);
    }

    public void setEventName(String name){
        this.name.setText(name);
    }

    public void setEventTime(String name){
        this.time.setText(name);
    }
}
```