Appendix – VII

Table of Contents for Classes

Code Listing:

CLASS: Application.java

```
package com.example.test;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

/**
 * The entry point of the Spring Boot application.
 */
@SpringBootApplication
public class Application extends SpringBootServletInitializer {

    public static void main(String[] args) {
                SpringApplication.run(Application.class, args);
    }

}
```

CLASS: Attendance.java

```
package com.example.test;

import java.util.ArrayList;

public class Attendance {

 //instance variables - more need to be added
 private ArrayList<Boolean> presentOrAbsent = new ArrayList<Boolean>();
 private ArrayList<String> reasonAbsent = new ArrayList<String>();
 private ArrayList<Boolean> covidScreening= new ArrayList<Boolean>();
 private ArrayList<String> reasonCovidScreening =  new ArrayList<String>();

 //constructor
 public Attendance () {

 }

 //getter methods
 public ArrayList <Boolean> getAttendance() {
  return presentOrAbsent;
 }

 public ArrayList <String> getReasonAbsent() {
  return reasonAbsent;
 }

 public ArrayList <Boolean> getCovidScreening() {
  return covidScreening;
 }

 public ArrayList <String> getReasonCovidScreening () {
  return reasonCovidScreening;
 }

 //setter adding methods
 public void addAttendance(boolean pOrA) {
  presentOrAbsent.add(pOrA);
 }

 public void addReasonAbsent(String reason) {
  reasonAbsent.add(reason);
 }

 public void addCovidScreening(boolean screening) {
  covidScreening.add(screening);
 }

 public void addReasonCovidScreening(String reason2) {
  reasonCovidScreening.add(reason2);
 }
}
```

CLASS: MainView.java

```
package com.example.test;

import com.vaadin.flow.component.Key;
import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.dependency.CssImport;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.Route;
import com.vaadin.flow.server.PWA;
import org.springframework.beans.factory.annotation.Autowired;

/**
 * A sample Vaadin view class.
 * <p>
 * To implement a Vaadin view just extend any Vaadin component and
```

```
 * use @Route annotation to announce it in a URL as a Spring managed
 * bean.
 * Use the @PWA annotation make the application installable on phones,
 * tablets and some desktop browsers.
 * <p>
 * A new instance of this class is created for every new user and every
 * browser tab/window.
 */
@Route
@PWA(name = "Vaadin Application",
                shortName = "Vaadin App",
                description = "This is an example Vaadin application.",
                enableInstallPrompt = false)
@CssImport("./styles/shared-styles.css")
@CssImport(value = "./styles/vaadin-text-field-styles.css", themeFor = "vaadin-text-field")
public class MainView extends VerticalLayout {

    /**
                * Construct a new Vaadin view.
                * <p>
                * Build the initial UI state for the user accessing the application.
                *
                * @param service The message service. Automatically injected Spring managed bean.
                */
    public MainView(@Autowired GreetService service) {

                //welcome teacher
                        H2 intro = new H2 ("Welcome to Miftahul Qur'an Acadamy!");
                intro.setMinWidth("390px");

                //make logo of institute
                Image logo = new Image("images/image0.png", "Logo");

                //login button
                Button login = new Button("Login",
                e -> {
                        UI.getCurrent().navigate("login");
                });
                login.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                login.setMinWidth("250px");

                // You can specify keyboard shortcuts for buttons.
                // Example: Pressing enter in this view clicks the Button.
                login.addClickShortcut(Key.ENTER);

                // Use custom CSS classes to apply styling. This is defined in shared-styles.css.
        addClassName("centered-content");

                add(intro, logo, login);

                setSizeFull();
        setJustifyContentMode(JustifyContentMode.CENTER);
        setDefaultHorizontalComponentAlignment(Alignment.CENTER);
                getStyle().set("text-align", "center");
    }

}
```

CLASS: Login.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

import com.vaadin.flow.component.Text;
import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.icon.Icon;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.notification.NotificationVariant;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.PasswordField;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.router.Route;

@Route("login")
public class Login extends VerticalLayout {
                static ArrayList <Teacher> teacherInformation = new ArrayList <Teacher>();
                static Scanner fileScanner;

                public Login() {

                teacherInformation.removeAll(teacherInformation);
                fileTwoOpen();

                H2 intro = new H2 ("Log in");

                TextField fname = new TextField();
                                fname.setLabel("First Name");
                                fname.setRequiredIndicatorVisible(true);
                                fname.setErrorMessage("This field is required");

                                TextField lname = new TextField();
                                lname.setLabel("Last Name");
```

```
                lname.setRequiredIndicatorVisible(true);
                lname.setErrorMessage("This field is required");

                PasswordField password = new PasswordField();
                password.setLabel("Password");
                password.setErrorMessage("This field is required");

                Button login = new Button("Log in", e->{
                        boolean flag = false;
                        for (int i = 0; i < teacherInformation.size(); i++) {
                if (fname.getValue().equals(teacherInformation.get(i).getFirstName()) && lname.getValue().equals(teacherInformation.get(i).getLastName()) && pass-
word.getValue().equals(teacherInformation.get(i).getPassword())) {
                UI.getCurrent().navigate("emergencyOrNot");
                flag = true;
                break;
                }
                }
                        if (flag == false) {
                    Notification notification = new Notification();
                    notification.addThemeVariants(NotificationVariant.LUMO_ERROR);

                    Div text = new Div(new Text("Incorrect name or password entered"));

                    Button closeButton = new Button(new Icon("lumo", "cross"));
                    closeButton.addThemeVariants(ButtonVariant.LUMO_TERTIARY_INLINE);
                    closeButton.getElement().setAttribute("aria-label", "Close");
                    closeButton.addClickListener(event -> {
                      notification.close();
                    });

                    HorizontalLayout layout = new HorizontalLayout(text, closeButton);
                    layout.setAlignItems(Alignment.CENTER);

                    notification.add(layout);
                    notification.open();
                        }
                });
                login.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                Button cancel = new Button("Cancel", e-> {
                        UI.getCurrent().navigate("");
                });

                HorizontalLayout buttonLayout = new HorizontalLayout(login, cancel);

            // Use custom CSS classes to apply styling. This is defined in shared-styles.css.
            addClassName("centered-content");

                add (intro, fname, lname, password, buttonLayout);

                setSizeFull();
    setJustifyContentMode(JustifyContentMode.CENTER);
    setDefaultHorizontalComponentAlignment(Alignment.CENTER);
                getStyle().set("text-align", "center");

  }

                //open teacher file
                 public static ArrayList <Teacher> fileTwoOpen() {
                try {
                fileScanner = new Scanner(new File("../marchbreakia/teacher.txt"));
                } catch (FileNotFoundException e) {
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
                }

                String fname, lname, password;

                while (fileScanner.hasNextLine()) {

                fname = (fileScanner.nextLine()).toLowerCase();
                lname = (fileScanner.nextLine()).toLowerCase();
                password = fileScanner.nextLine();

                Teacher tempT = new Teacher(fname, lname, password);
                teacherInformation.add(tempT);
                }
                fileScanner.close();

                return teacherInformation;
                 }
}
```

ClASS: Menu.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.io.PrintWriter;
import java.util.Scanner;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.contextmenu.MenuItem;
import com.vaadin.flow.component.contextmenu.SubMenu;
```

```
import com.vaadin.flow.component.datepicker.DatePicker;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.menubar.MenuBar;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.html.Hr;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.radiobutton.RadioButtonGroup;
import com.vaadin.flow.component.radiobutton.RadioGroupVariant;
import com.vaadin.flow.component.textfield.PasswordField;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.data.binder.Binder;
import com.vaadin.flow.router.Route;

@Route(value = "menu", layout = Welcome.class)
public class Menu extends VerticalLayout {
                    static Scanner fileScanner;
                    static ArrayList <Student> listOfStudents = new ArrayList <Student>();
                    static ArrayList <Teacher> teacherInformation = new ArrayList <Teacher>();
                    public Menu() {
                    //read from file
                    listOfStudents.removeAll(listOfStudents);
                    teacherInformation.removeAll(teacherInformation);
                    listOfStudents = fileOneOpen();
                    teacherInformation = fileTwoOpen();

                    H1 intro = new H1 ("Let's Get Started For Today's Class!");
                    intro.setMinWidth("700px");
                                    MenuBar menuBar = new MenuBar();
                                    menuBar.setOpenOnHover(true);
                                    menuBar.setHeight("200px");
                                    addItems(menuBar);
                                    add(intro, menuBar);
                                    addClassName("centered-content");
                                    setSizeFull();
            setJustifyContentMode(JustifyContentMode.CENTER);
            setDefaultHorizontalComponentAlignment(Alignment.CENTER);
                                    getStyle().set("text-align", "center");
                                    }

                                    private void addItems(MenuBar menuBar) {

                                    MenuItem attendance = menuBar.addItem("Attendance");
                                    SubMenu attendanceSubMenu = attendance.getSubMenu();
                                    attendanceSubMenu.addItem("Today's Attendance", e -> {
                                      UI.getCurrent().navigate("attendance");
                                    });
                                    attendanceSubMenu.add(new Hr());
                                    MenuItem aAttendance =  attendanceSubMenu.addItem("View a student attendance");
                                    SubMenu aAttendanceSubMenu = aAttendance.getSubMenu();
                                    aAttendanceSubMenu.addItem("Today", e -> {
                                      Dialog dialog = new Dialog();
                                                    dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                                                    VerticalLayout dialogLayout = createDialogLayout2(dialog);
                                                    dialog.add(dialogLayout);
                                                    add(dialog);
                                                    dialog.open();

                                    });
                                    aAttendanceSubMenu.addItem("Previous date", e -> {
                                      Dialog dialog = new Dialog();
                                                    dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                                                    VerticalLayout dialogLayout = createDialogLayout(dialog);
                                                    dialog.add(dialogLayout);
                                                    add(dialog);
                                                    dialog.open();
                                    });

                                    MenuItem allAttendance = attendanceSubMenu.addItem("View all student attendances");
                                    SubMenu allAttendanceSubMenu = allAttendance.getSubMenu();
                                    allAttendanceSubMenu.addItem("Today", e -> {
                                                      UI.getCurrent().navigate("menuAllStudentT");

                                    });
                                    allAttendanceSubMenu.addItem("Previous date", e -> {
                                      Dialog dialog = new Dialog();
                                                    dialog.getElement().setAttribute("aria-label", "Enter Date");

                                                    VerticalLayout dialogLayout = createDialogLayout3(dialog);
                                                    dialog.add(dialogLayout);
                                                    add(dialog);
                                                    dialog.open();
                                    });
                                    attendanceSubMenu.addItem("View days absent", e -> {

                                                      Dialog dialog = new Dialog();
                                                    dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                                                    VerticalLayout dialogLayout = createDialogLayout4(dialog);
                                                    dialog.add(dialogLayout);
                                                    dialog.open();
                                                    add(dialog);
                                    });

                                    MenuItem sRecords = menuBar.addItem("Student Records");
                                    SubMenu sRecordsSubMenu = sRecords.getSubMenu();
                                    sRecordsSubMenu.addItem("View a student record", e -> {
                                      Dialog dialog = new Dialog();
                                                    dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                                                    VerticalLayout dialogLayout = createDialogLayout5(dialog, 1);
                                                    dialog.add(dialogLayout);
                                                    dialog.open();
                                                    add(dialog);
                                    });
                                    sRecordsSubMenu.addItem("Edit a student record", e -> {
                                    Dialog dialog = new Dialog();
                                                    dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                                                    VerticalLayout dialogLayout = createDialogLayout5(dialog, 2);
                                                    dialog.add(dialogLayout);
                                                    dialog.open();
                                                    add(dialog);
                                    });
```

```
MenuItem sProgress = menuBar.addItem("Student Progress");
SubMenu sProgressSubMenu = sProgress.getSubMenu();
sProgressSubMenu.addItem("Record Today's Progress", e-> {
  Dialog dialog = new Dialog();
                dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                VerticalLayout dialogLayout = createDialogLayout5(dialog, 3);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
  });
sProgressSubMenu.add(new Hr());
MenuItem viewProgress = sProgressSubMenu.addItem("View progress");
SubMenu viewProgressSubMenu = viewProgress.getSubMenu();
viewProgressSubMenu.addItem("Daily progress", e-> {

  Dialog dialog = new Dialog();
                dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                VerticalLayout dialogLayout = createDialogLayout5(dialog, 4);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
});
viewProgressSubMenu.addItem("Monthly progress", e -> {
  Dialog dialog = new Dialog();
                dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                VerticalLayout dialogLayout = createDialogLayout5(dialog, 5);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
});
sProgressSubMenu.addItem("Change program chosen", e -> {
  Dialog dialog = new Dialog();
                dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                VerticalLayout dialogLayout = createDialogLayout12(dialog);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
});

MenuItem other = menuBar.addItem("Other");
SubMenu otherSubMenu = other.getSubMenu();
MenuItem otherSubMenu15 = otherSubMenu.addItem("Manage passwords and personal information");
 SubMenu otherSubMenu1 = otherSubMenu15.getSubMenu();
otherSubMenu1.addItem("View teacher information", e -> {
  Dialog dialog = new Dialog();
                VerticalLayout dialogLayout = createDialogLayout6(dialog);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
                 getStyle().set("position", "fixed").set("top","0").set("right", "0")
.set("bottom", "0").set("left", "0").set("display", "flex")
.set("align-items", "center").set("justify-content", "center");
});
otherSubMenu1.add(new Hr());
otherSubMenu1.addItem("Change your first name", e -> {
  Dialog dialog = new Dialog();
                VerticalLayout dialogLayout = createDialogLayout7(dialog);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
                 getStyle().set("position", "fixed").set("top","0").set("right", "0")
.set("bottom", "0").set("left", "0").set("display", "flex")
.set("align-items", "center").set("justify-content", "center");

});
otherSubMenu1.addItem("Change your last name", e -> {

  Dialog dialog = new Dialog();
                VerticalLayout dialogLayout = createDialogLayout8(dialog);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
                 getStyle().set("position", "fixed").set("top","0").set("right", "0")
.set("bottom", "0").set("left", "0").set("display", "flex")
.set("align-items", "center").set("justify-content", "center");
});
otherSubMenu1.addItem("Change your password", e -> {
  Dialog dialog = new Dialog();
  VerticalLayout dialogLayout = createDialogLayout9(dialog);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
                 getStyle().set("position", "fixed").set("top","0").set("right", "0")
.set("bottom", "0").set("left", "0").set("display", "flex")
.set("align-items", "center").set("justify-content", "center");
});
otherSubMenu.addItem("Display class list", e -> {
  Dialog dialog = new Dialog();
dialog.getElement().setAttribute("aria-label", "Class list");

VerticalLayout dialogLayout = createDialogLayout10();
dialog.add(dialogLayout);
dialog.setDraggable(true);
dialog.setResizable(true);
dialog.open();
add(dialog);
});
otherSubMenu.addItem("Add a new student", e -> UI.getCurrent().navigate("menuDOtherN"));
otherSubMenu.addItem("Delete student", e -> {

  Dialog dialog = new Dialog();
                VerticalLayout dialogLayout = createDialogLayout11(dialog);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
                 getStyle().set("position", "fixed").set("top","0").set("right", "0")
.set("bottom", "0").set("left", "0").set("display", "flex")
.set("align-items", "center").set("justify-content", "center");

});
}

 //dialog for entering another student's information
private static VerticalLayout createDialogLayout(Dialog dialog) {
                H2 headline = new H2("Enter Student Information");
```

```
                headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                .set("font-size", "1.5em").set("font-weight", "bold");

                TextField firstNameField = new TextField("First Name");
                TextField lastNameField = new TextField("Last Name");
                DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                DatePicker singleFormatI18n = new DatePicker("Pick a Date");
                VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                lastNameField, singleFormatI18n);
                fieldLayout.setSpacing(false);
                fieldLayout.setPadding(false);
                fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                Button cancelButton = new Button("Cancel", e -> dialog.close());
                Button saveButton = new Button("Done", e -> {
                  int index = -2;
                boolean found = false;
                boolean found2 = false;
                  for (int i = 0; i < listOfStudents.size(); i++) {
                if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
                                found = true;
                                for (int k = 0; k < listOfStudents.get(i).getDate().size(); k++) {
                                  if (listOfStudents.get(i).getDate().get(k).equals(firstFormatter1.format(singleFormatI18n.getValue()))) {
                                  found2 = true;
                                  index = i;
                                          store(index, firstFormatter1.format(singleFormatI18n.getValue()));
                                          closeFileOne(listOfStudents);
                                                  dialog.close();
                                          UI.getCurrent().navigate("menuAStudentP");
                                          break;
                                  }
                }
                                if (found2 == true) {
                                break;
                                }
                }
                }
                  if (!((found == true) && (found2 == true))) {
                                  if (found == true) {
                                  Notification.show("Invalid date entered.",
                3000, Notification.Position.MIDDLE);
                                  } else {
                                   Notification.show("Invalid name entered.",
                3000, Notification.Position.MIDDLE);
                                  }
                                  }


                });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
                buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                buttonLayout);
                dialogLayout.setPadding(false);
                dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                return dialogLayout;
                }

        //dialog for entering another student's information
        private static VerticalLayout createDialogLayout12(Dialog dialog) {
                H2 headline = new H2("Enter Student Information");
                headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                .set("font-size", "1.5em").set("font-weight", "bold");
                TextField firstNameField = new TextField("First Name");
                TextField lastNameField = new TextField("Last Name");
                RadioButtonGroup<String> radioGroup = new RadioButtonGroup<>();
                radioGroup.addThemeVariants(RadioGroupVariant.LUMO_VERTICAL);
                radioGroup.setLabel("Choose new program");
                radioGroup.setItems("Beginner (less than five saparas memorized)", "Intermediate (between five to twelve saparas memorized)", "Advanced (more than twelve saparas memorized)", "Hafiz
(the Qur'an memorized)");
                VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                lastNameField, radioGroup);
                fieldLayout.setSpacing(false);
                fieldLayout.setPadding(false);
                fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                Button cancelButton = new Button("Cancel", e -> dialog.close());
                Button saveButton = new Button("Done", e -> {
                  int index = -2;
                boolean found = false;
                  for (int i = 0; i < listOfStudents.size(); i++) {
                if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
                                found = true;
                                index = i;
                                if (radioGroup.getValue().equals("Beginner (less than five saparas memorized)")) {
                                listOfStudents.get(index).setProgramChosen("beginner");
                                } else if (radioGroup.getValue().equals("Intermediate (between five to twelve saparas memorized)")) {
                                listOfStudents.get(index).setProgramChosen("intermediate");
                                } else if (radioGroup.getValue().equals("Advanced (more than twelve saparas memorized)")) {
                                listOfStudents.get(index).setProgramChosen("advanced");
                                } else {
                                listOfStudents.get(index).setProgramChosen("hafiz");
                                }
                                closeFileOne(listOfStudents);
                                dialog.close();
                                if (found == true) {
                                break;
                                }
                }
                }
                  if (!((found == true))) {
                                  Notification.show("Invalid name entered.",
                3000, Notification.Position.MIDDLE);
                                  }


                });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
                buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                buttonLayout);
```

```
        dialogLayout.setPadding(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

        return dialogLayout;
        }

private static VerticalLayout createDialogLayout10() {
        H2 headline = new H2("Class List");
        headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
        .set("font-size", "1.5em").set("font-weight", "bold");

        Grid<Student> grid = new Grid<>(Student.class, false);
        grid.addColumn(Student::getFirstName).setHeader("First name").setAutoWidth(true).setFlexGrow(0);
        grid.addColumn(Student::getLastName).setHeader("Last name").setAutoWidth(true).setFlexGrow(0);
        grid.addColumn(Student::getProgramChosen).setHeader("Program chosen").setAutoWidth(true).setFlexGrow(0);
        grid.addColumn(Student::getAge).setHeader("Age").setAutoWidth(true).setFlexGrow(0);
        grid.addColumn(Student::getApplicable).setHeader("Applicable for Monthly Report?").setAutoWidth(true).setFlexGrow(0);

        grid.setItems(listOfStudents);

        VerticalLayout dialogLayout = new VerticalLayout(headline, grid);
        dialogLayout.setPadding(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("min-width", "350px")
        .set("max-width", "100%").set("height", "100%");

        return dialogLayout;
        }

private static VerticalLayout createDialogLayout7(Dialog dialog) {

        H2 headline = new H2("Change first name");
        headline.getStyle().set("margin-top", "0");

        Binder<Teacher> personBinder = new Binder<>(Teacher.class);
        personBinder.setBean(teacherInformation.get(0));

        TextField firstName = new TextField("First name");
        personBinder.forField(firstName).bind(
        Teacher::getFirstName,
    Teacher::setFirstName);

        Button cancelButton = new Button("Cancel", e -> dialog.close());
        Button saveButton = new Button("Change", e -> {
          closeFileTwo();
          closeFileOne(listOfStudents);
          dialog.close();
          });
        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
        saveButton);
        buttonLayout
          .setJustifyContentMode(FlexComponent.JustifyContentMode.END);
        buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

        VerticalLayout dialogLayout = new VerticalLayout(headline, firstName,
        buttonLayout);
        dialogLayout.setPadding(false);
        dialogLayout.setSpacing(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

        return dialogLayout;
        }

private static VerticalLayout createDialogLayout8(Dialog dialog) {

        H2 headline = new H2("Change last name");
        headline.getStyle().set("margin-top", "0");

        Binder<Teacher> personBinder = new Binder<>(Teacher.class);
        personBinder.setBean(teacherInformation.get(0));

        TextField lastName = new TextField("Last name");
        personBinder.forField(lastName).bind(
        Teacher::getLastName,
        Teacher::setLastName);

        Button cancelButton = new Button("Cancel", e -> dialog.close());
        Button saveButton = new Button("Change", e -> {
          closeFileTwo();
          closeFileOne(listOfStudents);
          dialog.close();
          });
        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
        saveButton);
        buttonLayout
      .setJustifyContentMode(FlexComponent.JustifyContentMode.END);
        buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

        VerticalLayout dialogLayout = new VerticalLayout(headline, lastName,
        buttonLayout);
        dialogLayout.setPadding(false);
        dialogLayout.setSpacing(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

        return dialogLayout;
        }

private static VerticalLayout createDialogLayout11(Dialog dialog) {

        H2 headline = new H2("Delete student");
        headline.getStyle().set("margin-top", "0");

        Paragraph warn = new Paragraph ("Notice: Clicking 'Done' will remove the student forever. This cannot be undone.");
        TextField firstName = new TextField("First name");
        TextField lastName = new TextField("Last name");

        Button cancelButton = new Button("Cancel", e -> dialog.close());
        Button saveButton = new Button("Done", e -> {
          boolean found = false;
          for (int i = 0; i < listOfStudents.size(); i++) {
                  if (firstName.getValue().equals(listOfStudents.get(i).getFirstName()) && (lastName.getValue().equals(listOfStudents.get(i).getLastName()))) {
                  found = true;
                  listOfStudents.remove(i);
                  closeFileOne(listOfStudents);
                  dialog.close();
                  }
```

```
                }
                if (found == false) {
                        Notification.show("Invalid name entered.",
            1500, Notification.Position.MIDDLE);
                }
                closeFileOne(listOfStudents);
                });
            saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
            HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
            saveButton);
            buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);
            buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

            VerticalLayout dialogLayout = new VerticalLayout(headline, warn, firstName, lastName,
            buttonLayout);
            dialogLayout.setPadding(false);
            dialogLayout.setSpacing(false);
            dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
            dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

            return dialogLayout;
            }

    private static VerticalLayout createDialogLayout9(Dialog dialog) {

            H2 headline = new H2("Change password");
            headline.getStyle().set("margin-top", "0");

            Binder<Teacher> personBinder = new Binder<>(Teacher.class);
            personBinder.setBean(teacherInformation.get(0));

            PasswordField password = new PasswordField("Password");
            PasswordField confirmPassword = new PasswordField("Confirm Password");

            Button cancelButton = new Button("Cancel", e -> dialog.close());
            Button saveButton = new Button("Change", e -> {
                if (password.getValue().equals(confirmPassword.getValue())) {
                                        teacherInformation.get(0).setPassword(confirmPassword.getValue());
                                        closeFileTwo();
                                        closeFileOne(listOfStudents);
                                        dialog.close();
                                        Notification.show("Success!",
            1500, Notification.Position.MIDDLE);
                        } else {
                                        Notification.show("Passwords do not match.",
                    3000, Notification.Position.MIDDLE);
                        }
                });
            saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
            HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
            saveButton);
            buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);
            buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

            VerticalLayout dialogLayout = new VerticalLayout(headline, password, confirmPassword,
            buttonLayout);
            dialogLayout.setPadding(false);
            dialogLayout.setSpacing(false);
            dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
            dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

            return dialogLayout;
            }


//dialog for entering another student's information
            private static VerticalLayout createDialogLayout2(Dialog dialog) {
            H2 headline = new H2("Enter Student Information");
            headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
            .set("font-size", "1.5em").set("font-weight", "bold");

            //text fields where the client can input the student they would like to view's first and last names
            TextField firstNameField = new TextField("First Name");
            TextField lastNameField = new TextField("Last Name");
            //add to fieldLayout, which gets added onto the user's screen
            VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
            lastNameField);
            //add styling
            fieldLayout.setSpacing(false);
            fieldLayout.setPadding(false);
            fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

            //cancel button to close dialog
            Button cancelButton = new Button("Cancel", e -> dialog.close());

            //done button to navigate user to the attendance view
            Button saveButton = new Button("Done", e -> {
              int index = -2;
            boolean studentFound = false;
            boolean dateFound = false;
            //search through listOfStudents ArrayList for a student with the same first and last name as the student information the client has entered
              for (int i = 0; i < listOfStudents.size(); i++) {
                                //if the match is found
            if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
                                studentFound = true;
                                //get todays date
                                DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                    LocalDateTime firstNow2 = LocalDateTime.now(ZoneId.systemDefault());
                    String alreadyDoneAttendance = firstFormatter1.format(firstNow2);
                    //search through the student's dates ArrayList and see if todays date matches any of the dates
                                for (int k = 0; k < listOfStudents.get(i).getDate().size(); k++) {
                                //if the match is found
                                 if (listOfStudents.get(i).getDate().get(k).equals(alreadyDoneAttendance)) {
                                 dateFound = true;
                                 index = i;
                                 //store index of student and date into temp.txt file and close dialog
                                        store(index,alreadyDoneAttendance);
                                        closeFileOne(listOfStudents);
                                        dialog.close();
                                    //navigate to desired attendance menu
                                        UI.getCurrent().navigate("menuAStudentT");
                                        //break for loop
                                        break;
                                 }
            }
                                //if date is found break the second for loop as well
                                if (dateFound == true) {
                                break;
                                }
```

```
                }
                }
                  //display warning messages for data entry errors
                  if (!((studentFound == true) && (dateFound == true))) {
                                if (studentFound == true) {
                                Notification.show("Attendance for today does not exist.",
                               3000, Notification.Position.MIDDLE);
                                } else {
                                  Notification.show("Invalid name entered.",
                3000, Notification.Position.MIDDLE);
                                }
                                }


                });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
                buttonLayout
.setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                buttonLayout);
                dialogLayout.setPadding(false);
                dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                return dialogLayout;
                }

                 //dialog for entering another student's information
                private static VerticalLayout createDialogLayout3(Dialog dialog) {
                H2 headline = new H2("Enter Date");
                headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                .set("font-size", "1.5em").set("font-weight", "bold");

                   DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                DatePicker singleFormatI18n = new DatePicker("Pick a Date");
                VerticalLayout fieldLayout = new VerticalLayout(singleFormatI18n);
                fieldLayout.setSpacing(false);
                fieldLayout.setPadding(false);
                fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                Button cancelButton = new Button("Cancel", e -> dialog.close());
                Button saveButton = new Button("Done", e -> {
                boolean found = false;
                  for (int i = 0; i < listOfStudents.size(); i++) {
                                for (int k = 0; k < listOfStudents.get(i).getDate().size(); k++) {
                                  if (listOfStudents.get(i).getDate().get(k).equals(firstFormatter1.format(singleFormatI18n.getValue()))) {
                                  found = true;
                                                store3(firstFormatter1.format(singleFormatI18n.getValue()));
                                                closeFileOne(listOfStudents);
                                                dialog.close();
                                                UI.getCurrent().navigate("menuAllStudentP");
                                                break;
                                  }
                }
                }
                  if (!(found == true)) {
                                  Notification.show("Invalid date entered.",
                                3000, Notification.Position.MIDDLE);
                                  }


                });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
                buttonLayout
.setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                buttonLayout);
                dialogLayout.setPadding(false);
                   dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                return dialogLayout;
                }
                private static VerticalLayout createDialogLayout4(Dialog dialog) {
                H2 headline = new H2("Enter Student Information");
                headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                .set("font-size", "1.5em").set("font-weight", "bold");

                TextField firstNameField = new TextField("First Name");
                TextField lastNameField = new TextField("Last Name");
                VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                lastNameField);
                fieldLayout.setSpacing(false);
                fieldLayout.setPadding(false);
                fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                Button cancelButton = new Button("Cancel", e -> dialog.close());
                Button saveButton = new Button("Done", e -> {
                  int index = -2;
                boolean found = false;
                  for (int i = 0; i < listOfStudents.size(); i++) {
                if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
                index = i;
                found = true;
                //store4(index);
                int daysAbsent = 0;
                for(int k = 0; k < listOfStudents.get(index).getAttendance().size(); k++) {
                if (listOfStudents.get(index).getAttendance().get(k) == false) {
                daysAbsent++;
                }

                }
                Notification.show("Days Absent: " + daysAbsent,
                5000, Notification.Position.MIDDLE);
                closeFileOne(listOfStudents);
                                dialog.close();
                break;
                }
                }

                if (found == false) {
                                Notification.show("Invalid name entered.",
                3000, Notification.Position.MIDDLE);
                }
```

```
        });
        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
        saveButton);
        buttonLayout
.setJustifyContentMode(FlexComponent.JustifyContentMode.END);

        VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
        buttonLayout);
        dialogLayout.setPadding(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

        return dialogLayout;
        }

        private static VerticalLayout createDialogLayout5(Dialog dialog, int send) {
        H2 headline = new H2("Enter Student Information");
        headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
        .set("font-size", "1.5em").set("font-weight", "bold");

        TextField firstNameField = new TextField("First Name");
        TextField lastNameField = new TextField("Last Name");
        VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
        lastNameField);
        fieldLayout.setSpacing(false);
        fieldLayout.setPadding(false);
        fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

        Button cancelButton = new Button("Cancel", e -> dialog.close());
        Button saveButton = new Button("Done", e -> {
          int index = -2;
        boolean found = false;
          for (int i = 0; i < listOfStudents.size(); i++) {
        if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
        index = i;
        found = true;
        store4(index);
        closeFileOne(listOfStudents);
        dialog.close();
                        if (send == 1) {
        UI.getCurrent().navigate("menuBRecordsV");
                        } else if (send == 2) {
                                UI.getCurrent().navigate("menuBRecordsE");
                        } else if (send == 3) {
                                UI.getCurrent().navigate("menuCProgressR");
                        } else if (send == 4) {
                                UI.getCurrent().navigate("menuCProgressD");
                        } else if (send == 5) {
                                UI.getCurrent().navigate("menuCProgressM");
                        }
        break;
        }
        }

        if (found == false) {
                        Notification.show("Invalid name entered.",
        3000, Notification.Position.MIDDLE);
        }


        });
        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
        saveButton);
        buttonLayout
.setJustifyContentMode(FlexComponent.JustifyContentMode.END);

        VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
        buttonLayout);
        dialogLayout.setPadding(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

        return dialogLayout;
        }


        private static VerticalLayout createDialogLayout6(Dialog dialog) {

        H2 headline = new H2("Teacher Information");
        headline.getStyle().set("margin-top", "0");

        Paragraph fName = new Paragraph ("First Name: " + teacherInformation.get(0).getFirstName());
        Paragraph lName = new Paragraph ("Last Name: " + teacherInformation.get(0).getLastName());
        Paragraph password = new Paragraph ("Password: " + teacherInformation.get(0).getPassword());

        Button saveButton = new Button("Done", e -> {
          closeFileOne(listOfStudents);
          dialog.close();
        });
        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        HorizontalLayout buttonLayout = new HorizontalLayout(
        saveButton);
        buttonLayout
.setJustifyContentMode(FlexComponent.JustifyContentMode.END);
        buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

        VerticalLayout dialogLayout = new VerticalLayout(headline, fName, lName, password,
        buttonLayout);
        dialogLayout.setPadding(false);
        dialogLayout.setSpacing(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

        return dialogLayout;
        }




        public static void closeFileOne(ArrayList <Student> listOfStudents) {
                        PrintWriter pw = null;
                        try {
                        pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                        } catch (FileNotFoundException e) {
                        System.err.print("couldn't open file for writing!");
                        System.exit(0);
                        }
```

```
                        for (int y = 0; y < listOfStudents.size(); y++) {

                        if (y == 0) {
                        pw.println(listOfStudents.get(y).getFirstName());
                        } else {
                        pw.println(listOfStudents.get(y).getFirstName());
                        }
                        pw.println(listOfStudents.get(y).getMiddleName());
                        pw.println(listOfStudents.get(y).getLastName());
                        pw.println(listOfStudents.get(y).getAddress());
                        pw.println(listOfStudents.get(y).getDateOfBirth());
                        pw.println(listOfStudents.get(y).getAge());
                        pw.println(listOfStudents.get(y).getPostalCode());
                        pw.println(listOfStudents.get(y).getLanguage());
                        pw.println(listOfStudents.get(y).getCountryOfBirth());
pw.println(listOfStudents.get(y).getProgramChosen());
                        pw.println(listOfStudents.get(y).getLastRecord());


                        String holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
                        if (k == 0) {
 holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
                        } else {
                        holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
                        }
                        }
                        pw.println(holder);

                        holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
                        if (k == 0) {
                        holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
                        } else {
                        holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
                        }
                        }
                        pw.println(holder);

  pw.println(listOfStudents.get(y).getCurrentQuarter());

  holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getNumOfDourSaparasDoneMonth().length; k++) {
                        if (k == 0) {
                        holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
                        } else {
                        holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
                        }
                        }
                        pw.println(holder);
                        pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStudents.get(y).getDourNextFill());

                        DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                        LocalDateTime firstNow = LocalDateTime.now();
                        String alreadyDone = firstFormatter.format(firstNow);

                        if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
                        pw.println(false);
                        pw.println(-1);
                        pw.println(false);
                        pw.println(-1);
                        pw.println(false);
                        pw.println(-1);
                        pw.println(-1);
                        pw.println(false);
                        pw.println(-1);
                        } else {
                        pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
                        pw.println(listOfStudents.get(y).getTodayQuartersDone());
                        pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
                        pw.println(listOfStudents.get(y).getTodayDourSaparaDone());

                        if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
                                        pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
                                        pw.println(listOfStudents.get(y).getTodayLinesMemorized());
                                        pw.println(listOfStudents.get(y).getTodayMistakesMade());
                                        pw.println(listOfStudents.get(y).isTodaySaparaFinished());
                          pw.println(listOfStudents.get(y).getTodaySaparaDone());
                        } else {
                        pw.println(false);
                        pw.println(-1);
                        pw.println(-1);
                        pw.println(false);
                        pw.println(-1);
                        }
                        }



                        if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
                                        holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
                        if (k == 0) {
                        holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
                        } else {
                        holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
                        }
                        }
                        pw.println(holder);

                        holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
                        if (k == 0) {
                        holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
                        } else {
                        holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
                        }
                        }
                        pw.println(holder);

                        holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
                        if (k == 0) {
                        holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
                        } else {
                        holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
                        }
                        }
```

```
                    pw.println(holder);

                    holder = "";
                    for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
                    if (k == 0) {
                    holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
                    }
                    }
                    pw.println(holder);

                    holder = "";
                    for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++) {
                    if (k == 0) {
                    holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
                    }
                    }
                    pw.println(holder);
                    pw.println(listOfStudents.get(y).getTotalSaparasDone());
                    pw.println(listOfStudents.get(y).getSaparasDone());
pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStudents.get(y).getSaparaNextFill());
                    } else {
                    pw.println(false);
                    pw.println(0);
                    pw.println(0);
                    pw.println(false);
                    pw.println(0);
                    pw.println(0);
                    pw.println(0);
                    pw.println(0);
                    pw.println(0);
                    }


                    //attendance
                    //printing to file for attendance
                    holder = "";
                    for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

                    if (k == 0) {
                    holder = "" + (listOfStudents.get(y).getAttendance().get(k));

                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));

                    }
                    }
                    pw.println(holder);

                    holder = "";
                    //printing to file for reason absent
                    for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

                    if (d == 0) {
                    holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

                    }
                    }
                    pw.println(holder);
                    //printing to file for covid screening
                    holder = "";
                    for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

                    if (r == 0) {
                    holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

                    }
                    }
                    pw.println(holder);

                    //printing to file for reason covid screening was not done
                    holder = "";
                    for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++) {

                    if (p == 0) {
                    holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreening().get(p));

                    }
                    }
                    pw.println(holder);

                    //printing to file for dates
                    holder = "";
                    for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

                    if (z == 0) {
                    holder = ""+(listOfStudents.get(y).getDate().get(z));

                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getDate().get(z));

                    }
                    }
                    pw.println(holder);
                    pw.println(listOfStudents.get(y).getGuardianOneFirstName());
                    pw.println(listOfStudents.get(y).getGuardianOneLastName());
                    pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
                    pw.println(listOfStudents.get(y).getGuardianOneEmail());
                    pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
                    pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
                    pw.println(listOfStudents.get(y).getGuardianTwoLastName());
                    pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
                    pw.println(listOfStudents.get(y).getGuardianTwoEmail());
                    pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

                    pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
                    pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
                     pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
                    pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
```

```
                        pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

                        pw.println(listOfStudents.get(y).getHealthFactorOne());
                        pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorTwo());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorThree());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                        }
                  pw.close();


                        }

                public static ArrayList <Student> fileOneOpen() {
                try {
                fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
                } catch (FileNotFoundException e) {
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
                }

                //programChosen  - CHECK CONSTRUCTORS
                //add health factors to printing out in emergency situation stuff

                String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tempReasonAttendance, tempCovid, tempReasonCovid;

Boolean[] dourDoneOrNot;
                Boolean todayDourDoneOrNot;
                int[] quarterNumDoneMonth;
                int todayQuartersDone, currentQuarter;
                Boolean[] numOfDourSaparasDoneMonth;
                Boolean todayDourSaparaDoneOrNot;
                int todayDourSaparaDone;
                int dourCurrentSapara, dourNextFill;

                String programChosen;
                String lastRecord;

                Boolean[] sabaqDoneOrNot;
                Boolean todaySabaqDoneOrNot;
                int[] linesMemorized;
                int todayLinesMemorized;
                int[] mistakesMade;
                int todayMistakesMade;
                Boolean[] numOfSaparasDoneMonth;
                Boolean todaySaparaFinished;
                int[] nameOfSaparasDoneMonth;
                int totalSaparasDone;
                int todaySaparaDone;
                String saparasDone;
                int currentSaparaMemorizing;
                int saparaNextFill = 0;

                int age;

                String tempDate;
                ArrayList<String> dates;


                String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
                String guardianOneEmail;
                Boolean guardianOneCallAtWork;
                String guardianTwoFirstName, guardianTwoLastName;
                String guardianTwoPhoneNumber;
                String guardianTwoEmail;
                Boolean guardianTwoCallAtWork;

                String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
                String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
                String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
                String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

                String healthFactorOne;
                Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
                String healthFactorTwo;
                  Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
                String healthFactorThree;
                Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

                Attendance attendanceOfStudent;
                StudentProgress progressOfStudent;

                while (fileScanner.hasNextLine()) {
                                  dourDoneOrNot = new Boolean[30];
                                quarterNumDoneMonth = new int[30];
                                numOfDourSaparasDoneMonth = new Boolean[30];
                                sabaqDoneOrNot = new Boolean[30];
                                linesMemorized = new int[30];
                                mistakesMade = new int[30];
                                numOfSaparasDoneMonth = new Boolean[30];
                                nameOfSaparasDoneMonth = new int[30];

                firstName = (fileScanner.nextLine()).toLowerCase();
                middleName = (fileScanner.nextLine()).toLowerCase();
                lastName = (fileScanner.nextLine()).toLowerCase();
                address = (fileScanner.nextLine()).toLowerCase();
                dateOfBirth = fileScanner.nextLine();
                age = Integer.parseInt(fileScanner.nextLine());
                postalCode = (fileScanner.nextLine()).toLowerCase();
                language = (fileScanner.nextLine()).toLowerCase();
                countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                //progress of student
                programChosen = (fileScanner.nextLine()).toLowerCase();
                progressOfStudent = new StudentProgress();
                progressOfStudent.setProgramChosen(programChosen);
```

```
lastRecord = (fileScanner.nextLine());
progressOfStudent.setLastRecord(lastRecord);


String tempDourDoneOrNot = fileScanner.nextLine();
String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
for (int i = 0; i < strDourDoneOrNot.length; i++) {
dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
}
progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

String tempQuarterNumDoneMonth = fileScanner.nextLine();
String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
}
progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

currentQuarter = Integer.parseInt(fileScanner.nextLine());              progressOfStudent.setOpenCurrentQuarter(currentQuarter);

String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                        String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
}
progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCurrentSapara);

dourNextFill = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenDourNextFill(dourNextFill);

DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
LocalDateTime firstNow = LocalDateTime.now();
String alreadyDone = firstFormatter.format(firstNow);

if (!(alreadyDone.equals(lastRecord))) {
if (programChosen.equals("hafiz")) {
Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
int holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());

todayDourDoneOrNot = false;
todayDourSaparaDoneOrNot = false;
todayQuartersDone = 0;
todayDourSaparaDone = 0;
progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

} else {
Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
int holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
todayDourDoneOrNot = false;
todayQuartersDone = 0;
todayDourSaparaDoneOrNot = false;
todayDourSaparaDone = 0;
todaySabaqDoneOrNot = false;
todayLinesMemorized = 0;
todayMistakesMade = 0;
todaySaparaFinished = false;
todaySaparaDone = 0;
        progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
}


} else {
todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

if (!(programChosen.equals("hafiz"))) {

todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);

todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
} else {
Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
int holder = Integer.parseInt(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
```

```
holder = Integer.parseInt(fileScanner.nextLine());
}
}



if (!(programChosen.equals("hafiz"))) {
String tempSabaqDoneOrNot = fileScanner.nextLine();
String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
}
progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

String tempLinesMemorized = fileScanner.nextLine();
String strLinesMemorized[] = tempLinesMemorized.split(",");
for (int i = 0; i < strLinesMemorized.length; i++) {
linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
}
progressOfStudent.setOpenLinesMemorized(linesMemorized);

String tempMistakesMade = fileScanner.nextLine();
String strMistakesMade[] = tempMistakesMade.split(",");
for (int i = 0; i < strMistakesMade.length; i++) {
mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
}
progressOfStudent.setOpenMistakesMade(mistakesMade);

String tempNumOfSaparasFinished = fileScanner.nextLine();
String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
}
progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

String tempNameOfSaparasFinished = fileScanner.nextLine();
String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
}
progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

saparasDone = fileScanner.nextLine();
progressOfStudent.setOpenSaparasDone(saparasDone);

currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

saparaNextFill = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
} else {
String hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
}


//attendance

tempAttendance = fileScanner.nextLine();
String attendance[] = tempAttendance.split(",");
attendanceOfStudent = new Attendance();
for (int i = 0; i < attendance.length; i++ ) {
attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
}
tempReasonAttendance = fileScanner.nextLine();
String tempReason[] = tempReasonAttendance.split(",");
for (int i = 0; i < tempReason.length; i++ ) {
attendanceOfStudent.addReasonAbsent(tempReason[i]);
}
tempCovid = fileScanner.nextLine();
String covid[] = tempCovid.split(",");
for (int i = 0; i < covid.length; i++ ) {
attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
}
tempReasonCovid = fileScanner.nextLine();
String reasonCov[] = tempReasonCovid.split(",");
for (int i = 0; i < reasonCov.length; i++ ) {
attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
}

dates = new ArrayList<String>();

tempDate = fileScanner.nextLine();
String date[] = tempDate.split(",");
for (int i = 0; i < date.length; i++ ) {
dates.add(date[i]);
}

guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
guardianOnePhoneNumber = fileScanner.nextLine();
guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
    guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
guardianTwoPhoneNumber = fileScanner.nextLine();
guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
emergencyContactOneHomeNumber = (fileScanner.nextLine());
emergencyContactOneCellNumber = (fileScanner.nextLine());
emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
emergencyContactTwoHomeNumber = (fileScanner.nextLine());
emergencyContactTwoCellNumber = (fileScanner.nextLine());

healthFactorOne = (fileScanner.nextLine()).toLowerCase();
```

```
                healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

                Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,countryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardia-
nOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail, guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardi-
anTwoCallAtWork, emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCellNumber,emergencyContactT-
woFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFac-
torOnePlanOfCareRequired, healthFactorOneMedicationsRequired, healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree,
healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                listOfStudents.add(tempS);
                }
                fileScanner.close();

                return listOfStudents;
                }

                //write to teacher file method
                public static void closeFileTwo() {
                //make new print writer called pw
                PrintWriter pw = null;
                //try and catch statement to assign print writer to file
                try {
                pw = new PrintWriter(new File("teacher.txt"));
                //if file not found then display error and exit program
                } catch (FileNotFoundException e) {
                System.err.print("couldn't open file for writing!");
                System.exit(0);
                }
                //iterate through teacher information array
                for (int y = 0; y < teacherInformation.size(); y++) {
                                //print the teachers information to file
                pw.println(teacherInformation.get(y).getFirstName());
        pw.println(teacherInformation.get(y).getLastName());
                pw.println(teacherInformation.get(y).getPassword());
                }
                //close print writer when done
                pw.close();
                }

        public static void store(int index, String date) {
        PrintWriter pw = null;
                try {
                pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                pw.println(index);
                pw.println(date);
                pw.close();
                } catch (FileNotFoundException e) {
                System.err.print("couldn't open file for writing!");
                System.exit(0);
                }
        }
        public static void store3(String date) {
        PrintWriter pw = null;
                try {
                pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                pw.println(date);
                pw.close();
                } catch (FileNotFoundException e) {
                   System.err.print("couldn't open file for writing!");
                System.exit(0);
                }
        }
        public static void store4(int index) {
        PrintWriter pw = null;
                try {
                pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                pw.println(index);
                pw.close();
                } catch (FileNotFoundException e) {
                System.err.print("couldn't open file for writing!");
                System.exit(0);
                }
        }

        //reading from file for teacher.txt method
        public static ArrayList <Teacher> fileTwoOpen() {
        //try and catch statement to assign a scanner to the text file
                try {
                fileScanner = new Scanner(new File("../marchbreakia/teacher.txt"));
                //print out error and close program if not found
                } catch (FileNotFoundException e) {
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
                }

                //variables to hold information from file
                String fname, lname, password;
                //while scanner has a next line, keep reading from file
                while (fileScanner.hasNextLine()) {


                //read from file line by line and assign each line to its own variable
                //these lines are lower cased strings too maintain ambiguity except for password
                fname = (fileScanner.nextLine()).toLowerCase();
                lname = (fileScanner.nextLine()).toLowerCase();
                password = fileScanner.nextLine();
                //make new teacher object and store with information read from file
                Teacher tempT = new Teacher(fname, lname, password);
                teacherInformation.add(tempT);
                }
                //close scanner
                fileScanner.close();
                //return the teacher information
                return teacherInformation;
        }


}
```

CLASS: EmergencyOrNot.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.io.PrintWriter;
import java.util.Scanner;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.contextmenu.MenuItem;
import com.vaadin.flow.component.contextmenu.SubMenu;
import com.vaadin.flow.component.datepicker.DatePicker;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.menubar.MenuBar;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.html.Hr;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.radiobutton.RadioButtonGroup;
import com.vaadin.flow.component.radiobutton.RadioGroupVariant;
import com.vaadin.flow.component.textfield.PasswordField;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.data.binder.Binder;
import com.vaadin.flow.router.Route;

@Route(value = "menu", layout = Welcome.class)
public class Menu extends VerticalLayout {
		static Scanner fileScanner;
		static ArrayList <Student> listOfStudents = new ArrayList <Student>();
		static ArrayList <Teacher> teacherInformation = new ArrayList <Teacher>();
		public Menu() {
		//read from file
		listOfStudents.removeAll(listOfStudents);
		teacherInformation.removeAll(teacherInformation);
		listOfStudents = fileOneOpen();
		teacherInformation = fileTwoOpen();

		H1 intro = new H1 ("Let's Get Started For Today's Class!");
		intro.setMinWidth("700px");
				MenuBar menuBar = new MenuBar();
				menuBar.setOpenOnHover(true);
				menuBar.setHeight("200px");
				addItems(menuBar);
				add(intro, menuBar);
				addClassName("centered-content");
				setSizeFull();
	setJustifyContentMode(JustifyContentMode.CENTER);
	setDefaultHorizontalComponentAlignment(Alignment.CENTER);
				getStyle().set("text-align", "center");
				}

				private void addItems(MenuBar menuBar) {

				MenuItem attendance = menuBar.addItem("Attendance");
				SubMenu attendanceSubMenu = attendance.getSubMenu();
				attendanceSubMenu.addItem("Today's Attendance", e -> {
				  UI.getCurrent().navigate("attendance");
				});
				attendanceSubMenu.add(new Hr());
				MenuItem aAttendance =  attendanceSubMenu.addItem("View a student attendance");
				SubMenu aAttendanceSubMenu = aAttendance.getSubMenu();
				aAttendanceSubMenu.addItem("Today", e -> {
				  Dialog dialog = new Dialog();
						dialog.getElement().setAttribute("aria-label", "Enter Student Information");

						VerticalLayout dialogLayout = createDialogLayout2(dialog);
						dialog.add(dialogLayout);
						add(dialog);
						dialog.open();

				});
				aAttendanceSubMenu.addItem("Previous date", e -> {
				  Dialog dialog = new Dialog();
				dialog.getElement().setAttribute("aria-label", "Enter Student Information");

				VerticalLayout dialogLayout = createDialogLayout(dialog);
						dialog.add(dialogLayout);
						add(dialog);
						dialog.open();
				});

				MenuItem allAttendance = attendanceSubMenu.addItem("View all student attendances");
				SubMenu allAttendanceSubMenu = allAttendance.getSubMenu();
				allAttendanceSubMenu.addItem("Today", e -> {
						  UI.getCurrent().navigate("menuAllStudentT");

				});
				allAttendanceSubMenu.addItem("Previous date", e -> {
				  Dialog dialog = new Dialog();
				dialog.getElement().setAttribute("aria-label", "Enter Date");

				VerticalLayout dialogLayout = createDialogLayout3(dialog);
						dialog.add(dialogLayout);
						add(dialog);
						dialog.open();
				});
				attendanceSubMenu.addItem("View days absent", e -> {

						  Dialog dialog = new Dialog();
				dialog.getElement().setAttribute("aria-label", "Enter Student Information");

				VerticalLayout dialogLayout = createDialogLayout4(dialog);
						dialog.add(dialogLayout);
						dialog.open();
						add(dialog);
				});

				MenuItem sRecords = menuBar.addItem("Student Records");
```

```
SubMenu sRecordsSubMenu = sRecords.getSubMenu();
sRecordsSubMenu.addItem("View a student record", e -> {
  Dialog dialog = new Dialog();
dialog.getElement().setAttribute("aria-label", "Enter Student Information");

VerticalLayout dialogLayout = createDialogLayout5(dialog, 1);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
});
sRecordsSubMenu.addItem("Edit a student record", e -> {
Dialog dialog = new Dialog();
dialog.getElement().setAttribute("aria-label", "Enter Student Information");

VerticalLayout dialogLayout = createDialogLayout5(dialog, 2);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
});


MenuItem sProgress = menuBar.addItem("Student Progress");
SubMenu sProgressSubMenu = sProgress.getSubMenu();
sProgressSubMenu.addItem("Record Today's Progress", e-> {
  Dialog dialog = new Dialog();
dialog.getElement().setAttribute("aria-label", "Enter Student Information");

VerticalLayout dialogLayout = createDialogLayout5(dialog, 3);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
  });
sProgressSubMenu.add(new Hr());
MenuItem viewProgress = sProgressSubMenu.addItem("View progress");
SubMenu viewProgressSubMenu = viewProgress.getSubMenu();
viewProgressSubMenu.addItem("Daily progress", e-> {

  Dialog dialog = new Dialog();
dialog.getElement().setAttribute("aria-label", "Enter Student Information");

VerticalLayout dialogLayout = createDialogLayout5(dialog, 4);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
});
viewProgressSubMenu.addItem("Monthly progress", e -> {
  Dialog dialog = new Dialog();
                        dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                        VerticalLayout dialogLayout = createDialogLayout5(dialog, 5);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
});
sProgressSubMenu.addItem("Change program chosen", e -> {
  Dialog dialog = new Dialog();
                        dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                        VerticalLayout dialogLayout = createDialogLayout12(dialog);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
});

MenuItem other = menuBar.addItem("Other");
SubMenu otherSubMenu = other.getSubMenu();
MenuItem otherSubMenu15 = otherSubMenu.addItem("Manage passwords and personal information");
  SubMenu otherSubMenu1 = otherSubMenu15.getSubMenu();
otherSubMenu1.addItem("View teacher information", e -> {
  Dialog dialog = new Dialog();
                        VerticalLayout dialogLayout = createDialogLayout6(dialog);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
                         getStyle().set("position", "fixed").set("top","0").set("right", "0")
.set("bottom", "0").set("left", "0").set("display", "flex")
.set("align-items", "center").set("justify-content", "center");
});
otherSubMenu1.add(new Hr());
otherSubMenu1.addItem("Change your first name", e -> {
  Dialog dialog = new Dialog();
                        VerticalLayout dialogLayout = createDialogLayout7(dialog);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
                         getStyle().set("position", "fixed").set("top","0").set("right", "0")
.set("bottom", "0").set("left", "0").set("display", "flex")
.set("align-items", "center").set("justify-content", "center");

});
otherSubMenu1.addItem("Change your last name", e -> {

  Dialog dialog = new Dialog();
                        VerticalLayout dialogLayout = createDialogLayout8(dialog);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
                         getStyle().set("position", "fixed").set("top","0").set("right", "0")
.set("bottom", "0").set("left", "0").set("display", "flex")
.set("align-items", "center").set("justify-content", "center");
});
otherSubMenu1.addItem("Change your password", e -> {
  Dialog dialog = new Dialog();
  VerticalLayout dialogLayout = createDialogLayout9(dialog);
                        dialog.add(dialogLayout);
                        dialog.open();
                        add(dialog);
                         getStyle().set("position", "fixed").set("top","0").set("right", "0")
.set("bottom", "0").set("left", "0").set("display", "flex")
.set("align-items", "center").set("justify-content", "center");
});
otherSubMenu.addItem("Display class list", e -> {
  Dialog dialog = new Dialog();
dialog.getElement().setAttribute("aria-label", "Class list");

VerticalLayout dialogLayout = createDialogLayout10();
dialog.add(dialogLayout);
dialog.setDraggable(true);
dialog.setResizable(true);
dialog.open();
add(dialog);
```

```
                        });
                        otherSubMenu.addItem("Add a new student", e -> UI.getCurrent().navigate("menuDOtherN"));
                        otherSubMenu.addItem("Delete student", e -> {

                          Dialog dialog = new Dialog();
                                        VerticalLayout dialogLayout = createDialogLayout11(dialog);
                                        dialog.add(dialogLayout);
                                        dialog.open();
                                        add(dialog);
                                         getStyle().set("position", "fixed").set("top","0").set("right", "0")
                        .set("bottom", "0").set("left", "0").set("display", "flex")
                        .set("align-items", "center").set("justify-content", "center");

                        });
                        }

    //dialog for entering another student's information
    private static VerticalLayout createDialogLayout(Dialog dialog) {
                        H2 headline = new H2("Enter Student Information");
                        headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                        .set("font-size", "1.5em").set("font-weight", "bold");

                        TextField firstNameField = new TextField("First Name");
                        TextField lastNameField = new TextField("Last Name");
                        DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                        DatePicker singleFormatI18n = new DatePicker("Pick a Date");
                        VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                        lastNameField, singleFormatI18n);
                        fieldLayout.setSpacing(false);
                        fieldLayout.setPadding(false);
                        fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                        Button cancelButton = new Button("Cancel", e -> dialog.close());
                        Button saveButton = new Button("Done", e -> {
                          int index = -2;
                        boolean found = false;
                        boolean found2 = false;
                          for (int i = 0; i < listOfStudents.size(); i++) {
                        if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
                                        found = true;
                                        for (int k = 0; k < listOfStudents.get(i).getDate().size(); k++) {
                                          if (listOfStudents.get(i).getDate().get(k).equals(firstFormatter1.format(singleFormatI18n.getValue()))) {
                                          found2 = true;
                                          index = i;
                                                  store(index, firstFormatter1.format(singleFormatI18n.getValue()));
                                                  closeFileOne(listOfStudents);
                                                          dialog.close();
                                                  UI.getCurrent().navigate("menuAStudentP");
                                                  break;
                                          }
                        }
                                        if (found2 == true) {
                                        break;
                                        }
                        }
                        }
                          if (!((found == true) && (found2 == true))) {
                                          if (found == true) {
                                          Notification.show("Invalid date entered.",
                        3000, Notification.Position.MIDDLE);
                                          } else {
                                           Notification.show("Invalid name entered.",
                        3000, Notification.Position.MIDDLE);
                                          }
                                          }


                        });
                        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                        saveButton);
                        buttonLayout
                .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                        VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                        buttonLayout);
                        dialogLayout.setPadding(false);
                        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                        return dialogLayout;
                        }

    //dialog for entering another student's information
    private static VerticalLayout createDialogLayout12(Dialog dialog) {
                        H2 headline = new H2("Enter Student Information");
                        headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                        .set("font-size", "1.5em").set("font-weight", "bold");
                        TextField firstNameField = new TextField("First Name");
                        TextField lastNameField = new TextField("Last Name");
                        RadioButtonGroup<String> radioGroup = new RadioButtonGroup<>();
                        radioGroup.addThemeVariants(RadioGroupVariant.LUMO_VERTICAL);
                        radioGroup.setLabel("Choose new program");
                        radioGroup.setItems("Beginner (less than five saparas memorized)", "Intermediate (between five to twelve saparas memorized)", "Advanced (more than twelve saparas memorized)", "Hafiz
(the Qur'an memorized)");
                        VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                        lastNameField, radioGroup);
                        fieldLayout.setSpacing(false);
                        fieldLayout.setPadding(false);
                        fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                        Button cancelButton = new Button("Cancel", e -> dialog.close());
                        Button saveButton = new Button("Done", e -> {
                          int index = -2;
                        boolean found = false;
                          for (int i = 0; i < listOfStudents.size(); i++) {
                        if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
                                        found = true;
                                         index = i;
                                         if (radioGroup.getValue().equals("Beginner (less than five saparas memorized)")) {
                                         listOfStudents.get(index).setProgramChosen("beginner");
                                         } else if (radioGroup.getValue().equals("Intermediate (between five to twelve saparas memorized)")) {
                                         listOfStudents.get(index).setProgramChosen("intermediate");
                                         } else if (radioGroup.getValue().equals("Advanced (more than twelve saparas memorized)")) {
                                         listOfStudents.get(index).setProgramChosen("advanced");
                                         } else {
                                         listOfStudents.get(index).setProgramChosen("hafiz");
                                         }
                                         closeFileOne(listOfStudents);
                                        dialog.close();
                                        if (found == true) {
```

```
                                    break;
                                    }
                        }
                        }
                          if (!(((found == true))) {
                                      Notification.show("Invalid name entered.",
                        3000, Notification.Position.MIDDLE);
                                      }


                        });
                        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                        saveButton);
                        buttonLayout
                .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                        VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                        buttonLayout);
                        dialogLayout.setPadding(false);
                        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                        return dialogLayout;
                        }

        private static VerticalLayout createDialogLayout10() {
                        H2 headline = new H2("Class List");
                        headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                        .set("font-size", "1.5em").set("font-weight", "bold");

                        Grid<Student> grid = new Grid<>(Student.class, false);
                        grid.addColumn(Student::getFirstName).setHeader("First name").setAutoWidth(true).setFlexGrow(0);
                        grid.addColumn(Student::getLastName).setHeader("Last name").setAutoWidth(true).setFlexGrow(0);
                        grid.addColumn(Student::getProgramChosen).setHeader("Program chosen").setAutoWidth(true).setFlexGrow(0);
                        grid.addColumn(Student::getAge).setHeader("Age").setAutoWidth(true).setFlexGrow(0);
                        grid.addColumn(Student::getApplicable).setHeader("Applicable for Monthly Report?").setAutoWidth(true).setFlexGrow(0);

                        grid.setItems(listOfStudents);

                        VerticalLayout dialogLayout = new VerticalLayout(headline, grid);
                        dialogLayout.setPadding(false);
                        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                        dialogLayout.getStyle().set("min-width", "350px")
                        .set("max-width", "100%").set("height", "100%");

                        return dialogLayout;
                        }

        private static VerticalLayout createDialogLayout7(Dialog dialog) {

                        H2 headline = new H2("Change first name");
                        headline.getStyle().set("margin-top", "0");

                        Binder<Teacher> personBinder = new Binder<>(Teacher.class);
                        personBinder.setBean(teacherInformation.get(0));

                        TextField firstName = new TextField("First name");
                        personBinder.forField(firstName).bind(
                        Teacher::getFirstName,
                Teacher::setFirstName);

                        Button cancelButton = new Button("Cancel", e -> dialog.close());
                        Button saveButton = new Button("Change", e -> {
                          closeFileTwo();
                          closeFileOne(listOfStudents);
                          dialog.close();
                          });
                        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                        saveButton);
                        buttonLayout
                          .setJustifyContentMode(FlexComponent.JustifyContentMode.END);
                        buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

                        VerticalLayout dialogLayout = new VerticalLayout(headline, firstName,
                        buttonLayout);
                        dialogLayout.setPadding(false);
                        dialogLayout.setSpacing(false);
                        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                        dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

                        return dialogLayout;
                        }

        private static VerticalLayout createDialogLayout8(Dialog dialog) {

                        H2 headline = new H2("Change last name");
                        headline.getStyle().set("margin-top", "0");

                        Binder<Teacher> personBinder = new Binder<>(Teacher.class);
                        personBinder.setBean(teacherInformation.get(0));

                        TextField lastName = new TextField("Last name");
                        personBinder.forField(lastName).bind(
                        Teacher::getLastName,
                        Teacher::setLastName);

                        Button cancelButton = new Button("Cancel", e -> dialog.close());
                        Button saveButton = new Button("Change", e -> {
                          closeFileTwo();
                          closeFileOne(listOfStudents);
                          dialog.close();
                          });
                        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                        saveButton);
                        buttonLayout
                .setJustifyContentMode(FlexComponent.JustifyContentMode.END);
                        buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

                        VerticalLayout dialogLayout = new VerticalLayout(headline, lastName,
                        buttonLayout);
                        dialogLayout.setPadding(false);
                        dialogLayout.setSpacing(false);
                        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                        dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

                        return dialogLayout;
                        }
```

```
private static VerticalLayout createDialogLayout11(Dialog dialog) {

                H2 headline = new H2("Delete student");
                headline.getStyle().set("margin-top", "0");

                Paragraph warn = new Paragraph ("Notice: Clicking 'Done' will remove the student forever. This cannot be undone.");
                TextField firstName = new TextField("First name");
                TextField lastName = new TextField("Last name");

                Button cancelButton = new Button("Cancel", e -> dialog.close());
                Button saveButton = new Button("Done", e -> {
                  boolean found = false;
                  for (int i = 0; i < listOfStudents.size(); i++) {
                                if (firstName.getValue().equals(listOfStudents.get(i).getFirstName()) && (lastName.getValue().equals(listOfStudents.get(i).getLastName()))) {
                                found = true;
                                listOfStudents.remove(i);
                                closeFileOne(listOfStudents);
                                dialog.close();
                                }
                  }
                  if (found == false) {
                                Notification.show("Invalid name entered.",
                1500, Notification.Position.MIDDLE);
                  }
                  closeFileOne(listOfStudents);
                  });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
                buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);
                buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

                VerticalLayout dialogLayout = new VerticalLayout(headline, warn, firstName, lastName,
                buttonLayout);
                dialogLayout.setPadding(false);
                dialogLayout.setSpacing(false);
                dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

                return dialogLayout;
                }

private static VerticalLayout createDialogLayout9(Dialog dialog) {

                H2 headline = new H2("Change password");
                headline.getStyle().set("margin-top", "0");

                Binder<Teacher> personBinder = new Binder<>(Teacher.class);
                personBinder.setBean(teacherInformation.get(0));

                PasswordField password = new PasswordField("Password");
                PasswordField confirmPassword = new PasswordField("Confirm Password");

                Button cancelButton = new Button("Cancel", e -> dialog.close());
                Button saveButton = new Button("Change", e -> {
                  if (password.getValue().equals(confirmPassword.getValue())) {
                                                teacherInformation.get(0).setPassword(confirmPassword.getValue());
                                                closeFileTwo();
                                                closeFileOne(listOfStudents);
                                                dialog.close();
                                                Notification.show("Success!",
                1500, Notification.Position.MIDDLE);
                                } else {
                                                Notification.show("Passwords do not match.",
                        3000, Notification.Position.MIDDLE);
                                }
                  });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
                buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);
                buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

                VerticalLayout dialogLayout = new VerticalLayout(headline, password, confirmPassword,
                buttonLayout);
                dialogLayout.setPadding(false);
                dialogLayout.setSpacing(false);
                dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

                return dialogLayout;
                }


//dialog for entering another student's information
                private static VerticalLayout createDialogLayout2(Dialog dialog) {
                H2 headline = new H2("Enter Student Information");
                headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                .set("font-size", "1.5em").set("font-weight", "bold");

                //text fields where the client can input the student they would like to view's first and last names
                TextField firstNameField = new TextField("First Name");
                TextField lastNameField = new TextField("Last Name");
                //add to fieldLayout, which gets added onto the user's screen
                VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                lastNameField);
                //add styling
                fieldLayout.setSpacing(false);
                fieldLayout.setPadding(false);
                fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                //cancel button to close dialog
                Button cancelButton = new Button("Cancel", e -> dialog.close());

                //done button to navigate user to the attendance view
                Button saveButton = new Button("Done", e -> {
                  int index = -2;
                boolean studentFound = false;
                boolean dateFound = false;
                //search through listOfStudents ArrayList for a student with the same first and last name as the student information the client has entered
                  for (int i = 0; i < listOfStudents.size(); i++) {
                                //if the match is found
                if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
                                  studentFound = true;
                                  //get todays date
                                  DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                        LocalDateTime firstNow2 = LocalDateTime.now(ZoneId.systemDefault());
                        String alreadyDoneAttendance = firstFormatter1.format(firstNow2);
                        //search through the student's dates ArrayList and see if todays date matches any of the dates
```

```
                                for (int k = 0; k < listOfStudents.get(i).getDate().size(); k++) {
                                //if the match is found
                                  if (listOfStudents.get(i).getDate().get(k).equals(alreadyDoneAttendance)) {
                                  dateFound = true;
                                  index = i;
                                  //store index of student and date into temp.txt file and close dialog
                                            store(index,alreadyDoneAttendance);
                                            closeFileOne(listOfStudents);
                                            dialog.close();
                                       //navigate to desired attendance menu
                                            UI.getCurrent().navigate("menuAStudentT");
                                            //break for loop
                                            break;
                                  }
                        }
                                //if date is found break the second for loop as well
                                if (dateFound == true) {
                                break;
                                }
                        }
                        }
                          //display warning messages for data entry errors
                          if (!((studentFound == true) && (dateFound == true))) {
                                  if (studentFound == true) {
                                  Notification.show("Attendance for today does not exist.",
                                3000, Notification.Position.MIDDLE);
                                  } else {
                                  Notification.show("Invalid name entered.",
                        3000, Notification.Position.MIDDLE);
                                  }
                                  }


                        });
                        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                        saveButton);
                        buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                        VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                        buttonLayout);
                        dialogLayout.setPadding(false);
                        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                        return dialogLayout;
                        }

                         //dialog for entering another student's information
                        private static VerticalLayout createDialogLayout3(Dialog dialog) {
                        H2 headline = new H2("Enter Date");
                        headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                        .set("font-size", "1.5em").set("font-weight", "bold");

                           DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                        DatePicker singleFormatI18n = new DatePicker("Pick a Date");
                        VerticalLayout fieldLayout = new VerticalLayout(singleFormatI18n);
                        fieldLayout.setSpacing(false);
                        fieldLayout.setPadding(false);
                        fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                        Button cancelButton = new Button("Cancel", e -> dialog.close());
                        Button saveButton = new Button("Done", e -> {
                        boolean found = false;
                          for (int i = 0; i < listOfStudents.size(); i++) {
                                  for (int k = 0; k < listOfStudents.get(i).getDate().size(); k++) {
                                    if (listOfStudents.get(i).getDate().get(k).equals(firstFormatter1.format(singleFormatI18n.getValue()))) {
                                    found = true;
                                            store3(firstFormatter1.format(singleFormatI18n.getValue()));
                                            closeFileOne(listOfStudents);
                                            dialog.close();
                                            UI.getCurrent().navigate("menuAllStudentP");
                                            break;
                                  }
                        }
                        }
                          if (!(found == true)) {
                                  Notification.show("Invalid date entered.",
                                3000, Notification.Position.MIDDLE);
                                  }


                        });
                        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                        saveButton);
                        buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                        VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                        buttonLayout);
                        dialogLayout.setPadding(false);
                           dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                        return dialogLayout;
                        }
                        private static VerticalLayout createDialogLayout4(Dialog dialog) {
                        H2 headline = new H2("Enter Student Information");
                        headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                        .set("font-size", "1.5em").set("font-weight", "bold");

                        TextField firstNameField = new TextField("First Name");
                        TextField lastNameField = new TextField("Last Name");
                        VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                        lastNameField);
                        fieldLayout.setSpacing(false);
                        fieldLayout.setPadding(false);
                        fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                        Button cancelButton = new Button("Cancel", e -> dialog.close());
                        Button saveButton = new Button("Done", e -> {
                          int index = -2;
                        boolean found = false;
                          for (int i = 0; i < listOfStudents.size(); i++) {
                        if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
                        index = i;
                        found = true;
                        //store4(index);
                        int daysAbsent = 0;
```

```
                for(int k = 0; k < listOfStudents.get(index).getAttendance().size(); k++) {
                if (listOfStudents.get(index).getAttendance().get(k) == false) {
                daysAbsent++;
                }

                }
                Notification.show("Days Absent: " + daysAbsent,
                5000, Notification.Position.MIDDLE);
                closeFileOne(listOfStudents);
                                dialog.close();
                break;
                }
                }

                if (found == false) {
                                Notification.show("Invalid name entered.",
                3000, Notification.Position.MIDDLE);
                }


                });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
                buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                buttonLayout);
                dialogLayout.setPadding(false);
                dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                return dialogLayout;
                }

                private static VerticalLayout createDialogLayout5(Dialog dialog, int send) {
                H2 headline = new H2("Enter Student Information");
                headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                .set("font-size", "1.5em").set("font-weight", "bold");

                TextField firstNameField = new TextField("First Name");
                TextField lastNameField = new TextField("Last Name");
                VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                lastNameField);
                fieldLayout.setSpacing(false);
                fieldLayout.setPadding(false);
                fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                Button cancelButton = new Button("Cancel", e -> dialog.close());
                Button saveButton = new Button("Done", e -> {
                  int index = -2;
                boolean found = false;
                  for (int i = 0; i < listOfStudents.size(); i++) {
                if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStudents.get(i).getLastName())) {
                index = i;
                found = true;
                store4(index);
                closeFileOne(listOfStudents);
                dialog.close();
                                if (send == 1) {
                UI.getCurrent().navigate("menuBRecordsV");
                                } else if (send == 2) {
                                                UI.getCurrent().navigate("menuBRecordsE");
                                } else if (send == 3) {
                                                UI.getCurrent().navigate("menuCProgressR");
                                } else if (send == 4) {
                                                UI.getCurrent().navigate("menuCProgressD");
                                } else if (send == 5) {
                                                UI.getCurrent().navigate("menuCProgressM");
                                }
                break;
                }
                }

                if (found == false) {
                                Notification.show("Invalid name entered.",
                3000, Notification.Position.MIDDLE);
                }


                });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
                buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                buttonLayout);
                dialogLayout.setPadding(false);
                dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                return dialogLayout;
                }


                private static VerticalLayout createDialogLayout6(Dialog dialog) {

                H2 headline = new H2("Teacher Information");
                headline.getStyle().set("margin-top", "0");

                Paragraph fName = new Paragraph ("First Name: " + teacherInformation.get(0).getFirstName());
                Paragraph lName = new Paragraph ("Last Name: " + teacherInformation.get(0).getLastName());
                Paragraph password = new Paragraph ("Password: " + teacherInformation.get(0).getPassword());

                Button saveButton = new Button("Done", e -> {
                  closeFileOne(listOfStudents);
                  dialog.close();
                });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(
                saveButton);
                buttonLayout
        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);
                buttonLayout.getStyle().set("margin-top", "var(--lumo-space-m)");

                VerticalLayout dialogLayout = new VerticalLayout(headline, fName, lName, password,
                buttonLayout);
                dialogLayout.setPadding(false);
```

```
dialogLayout.setSpacing(false);
dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
dialogLayout.getStyle().set("width", "18rem").set("max-width", "100%");

return dialogLayout;
}




public static void closeFileOne(ArrayList <Student> listOfStudents) {
                        PrintWriter pw = null;
                        try {
                        pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                        } catch (FileNotFoundException e) {
                        System.err.print("couldn't open file for writing!");
                        System.exit(0);
                        }



                        for (int y = 0; y < listOfStudents.size(); y++) {

                        if (y == 0) {
                        pw.println(listOfStudents.get(y).getFirstName());
                        } else {
                        pw.println(listOfStudents.get(y).getFirstName());
                        }
                        pw.println(listOfStudents.get(y).getMiddleName());
                        pw.println(listOfStudents.get(y).getLastName());
                        pw.println(listOfStudents.get(y).getAddress());
                        pw.println(listOfStudents.get(y).getDateOfBirth());
                        pw.println(listOfStudents.get(y).getAge());
                        pw.println(listOfStudents.get(y).getPostalCode());
                        pw.println(listOfStudents.get(y).getLanguage());
                        pw.println(listOfStudents.get(y).getCountryOfBirth());
pw.println(listOfStudents.get(y).getProgramChosen());
                        pw.println(listOfStudents.get(y).getLastRecord());


                        String holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
                        if (k == 0) {
 holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
                        } else {
                        holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
                        }
                        }
                        pw.println(holder);

                        holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
                        if (k == 0) {
                        holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
                        } else {
                        holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
                        }
                        }
                        pw.println(holder);

  pw.println(listOfStudents.get(y).getCurrentQuarter());

  holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getNumOfDourSaparasDoneMonth().length; k++) {
                        if (k == 0) {
                        holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
                        } else {
                        holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
                        }
                        }
                        pw.println(holder);
                        pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStudents.get(y).getDourNextFill());

                        DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                        LocalDateTime firstNow = LocalDateTime.now();
                        String alreadyDone = firstFormatter.format(firstNow);

                        if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
                        pw.println(false);
                        pw.println(-1);
                        pw.println(false);
                        pw.println(-1);
                        pw.println(false);
                        pw.println(-1);
                        pw.println(-1);
                        pw.println(false);
                        pw.println(-1);
                        } else {
                        pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
                        pw.println(listOfStudents.get(y).getTodayQuartersDone());
                        pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
                        pw.println(listOfStudents.get(y).getTodayDourSaparaDone());

                        if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
                                        pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
                                        pw.println(listOfStudents.get(y).getTodayLinesMemorized());
                                        pw.println(listOfStudents.get(y).getTodayMistakesMade());
                                        pw.println(listOfStudents.get(y).isTodaySaparaFinished());
                          pw.println(listOfStudents.get(y).getTodaySaparaDone());
                        } else {
                        pw.println(false);
                        pw.println(-1);
                        pw.println(-1);
                        pw.println(false);
                        pw.println(-1);
                        }
                        }



                        if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
                                        holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
                        if (k == 0) {
                        holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
                        } else {
                        holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
                        }
                        }
                        pw.println(holder);
```

```
holder = "";
for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
if (k == 0) {
holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
} else {
holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
}
}
pw.println(holder);

holder = "";
for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
if (k == 0) {
holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
} else {
holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
}
}
pw.println(holder);

holder = "";
for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
if (k == 0) {
holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
} else {
holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
}
}
pw.println(holder);

holder = "";
for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++) {
if (k == 0) {
holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
} else {
holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
}
}
pw.println(holder);
pw.println(listOfStudents.get(y).getTotalSaparasDone());
pw.println(listOfStudents.get(y).getSaparasDone());
pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStudents.get(y).getSaparaNextFill());
} else {
pw.println(false);
pw.println(0);
pw.println(0);
pw.println(false);
pw.println(0);
pw.println(0);
pw.println(0);
pw.println(0);
pw.println(0);
}


//attendance
//printing to file for attendance
holder = "";
for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

if (k == 0) {
holder = "" + (listOfStudents.get(y).getAttendance().get(k));

} else {
holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));

}
}
pw.println(holder);

holder = "";
//printing to file for reason absent
for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

if (d == 0) {
holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

} else {
holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

}
}
pw.println(holder);
//printing to file for covid screening
holder = "";
for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

if (r == 0) {
holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

} else {
holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

}
}
pw.println(holder);

//printing to file for reason covid screening was not done
holder = "";
for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++) {

if (p == 0) {
holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

} else {
holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreening().get(p));

}
}
pw.println(holder);

//printing to file for dates
holder = "";
for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

if (z == 0) {
holder = ""+(listOfStudents.get(y).getDate().get(z));

} else {
holder = holder + ("," + listOfStudents.get(y).getDate().get(z));
```

```
                        }
                        }
                        pw.println(holder);
                        pw.println(listOfStudents.get(y).getGuardianOneFirstName());
                        pw.println(listOfStudents.get(y).getGuardianOneLastName());
                        pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
                        pw.println(listOfStudents.get(y).getGuardianOneEmail());
                        pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
                        pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
                        pw.println(listOfStudents.get(y).getGuardianTwoLastName());
                        pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
                        pw.println(listOfStudents.get(y).getGuardianTwoEmail());
                        pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

                        pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
                         pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

                        pw.println(listOfStudents.get(y).getHealthFactorOne());
                        pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorTwo());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorThree());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                        }
                  pw.close();


                        }

                public static ArrayList <Student> fileOneOpen() {
                try {
                fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
                } catch (FileNotFoundException e) {
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
                }

                //programChosen  - CHECK CONSTRUCTORS
                //add health factors to printing out in emergency situation stuff

                String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tempReasonAttendance, tempCovid, tempReasonCovid;

Boolean[] dourDoneOrNot;
                Boolean todayDourDoneOrNot;
                int[] quarterNumDoneMonth;
                int todayQuartersDone, currentQuarter;
                Boolean[] numOfDourSaparasDoneMonth;
                Boolean todayDourSaparaDoneOrNot;
                int todayDourSaparaDone;
                int dourCurrentSapara, dourNextFill;

                String programChosen;
                String lastRecord;

                Boolean[] sabaqDoneOrNot;
                Boolean todaySabaqDoneOrNot;
                int[] linesMemorized;
                int todayLinesMemorized;
                int[] mistakesMade;
                int todayMistakesMade;
                Boolean[] numOfSaparasDoneMonth;
                Boolean todaySaparaFinished;
                int[] nameOfSaparasDoneMonth;
                int totalSaparasDone;
                int todaySaparaDone;
                String saparasDone;
                int currentSaparaMemorizing;
                int saparaNextFill = 0;

                int age;

                String tempDate;
                ArrayList<String> dates;


                String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
                String guardianOneEmail;
                Boolean guardianOneCallAtWork;
                String guardianTwoFirstName, guardianTwoLastName;
                String guardianTwoPhoneNumber;
                String guardianTwoEmail;
                Boolean guardianTwoCallAtWork;

                String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
                String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
                String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
                String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

                String healthFactorOne;
                Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
                String healthFactorTwo;
                  Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
                String healthFactorThree;
                Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

                Attendance attendanceOfStudent;
                StudentProgress progressOfStudent;

                while (fileScanner.hasNextLine()) {
                                  dourDoneOrNot = new Boolean[30];
                                quarterNumDoneMonth = new int[30];
                                numOfDourSaparasDoneMonth = new Boolean[30];
                                sabaqDoneOrNot = new Boolean[30];
                                linesMemorized = new int[30];
```

```
                        mistakesMade = new int[30];
                        numOfSaparasDoneMonth = new Boolean[30];
                        nameOfSaparasDoneMonth = new int[30];

firstName = (fileScanner.nextLine()).toLowerCase();
middleName = (fileScanner.nextLine()).toLowerCase();
lastName = (fileScanner.nextLine()).toLowerCase();
address = (fileScanner.nextLine()).toLowerCase();
dateOfBirth = fileScanner.nextLine();
age = Integer.parseInt(fileScanner.nextLine());
postalCode = (fileScanner.nextLine()).toLowerCase();
language = (fileScanner.nextLine()).toLowerCase();
countryOfBirth = (fileScanner.nextLine()).toLowerCase();

//progress of student
programChosen = (fileScanner.nextLine()).toLowerCase();
progressOfStudent = new StudentProgress();
progressOfStudent.setProgramChosen(programChosen);

lastRecord = (fileScanner.nextLine());
progressOfStudent.setLastRecord(lastRecord);


String tempDourDoneOrNot = fileScanner.nextLine();
String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
for (int i = 0; i < strDourDoneOrNot.length; i++) {
dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
}
progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

String tempQuarterNumDoneMonth = fileScanner.nextLine();
String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
}
progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

currentQuarter = Integer.parseInt(fileScanner.nextLine());              progressOfStudent.setOpenCurrentQuarter(currentQuarter);

String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                        String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
}
progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCurrentSapara);

dourNextFill = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenDourNextFill(dourNextFill);

DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
LocalDateTime firstNow = LocalDateTime.now();
String alreadyDone = firstFormatter.format(firstNow);

if (!(alreadyDone.equals(lastRecord))) {
if (programChosen.equals("hafiz")) {
Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
int holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());

todayDourDoneOrNot = false;
todayDourSaparaDoneOrNot = false;
todayQuartersDone = 0;
todayDourSaparaDone = 0;
progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

} else {
Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
int holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
todayDourDoneOrNot = false;
todayQuartersDone = 0;
todayDourSaparaDoneOrNot = false;
todayDourSaparaDone = 0;
todaySabaqDoneOrNot = false;
todayLinesMemorized = 0;
todayMistakesMade = 0;
todaySaparaFinished = false;
todaySaparaDone = 0;
        progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
}


} else {
todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

if (!(programChosen.equals("hafiz"))) {
```

```
todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);

todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
} else {
Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
int holder = Integer.parseInt(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
temporary = Boolean.parseBoolean(fileScanner.nextLine());
holder = Integer.parseInt(fileScanner.nextLine());
}
}



if (!(programChosen.equals("hafiz"))) {
String tempSabaqDoneOrNot = fileScanner.nextLine();
String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
}
progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

String tempLinesMemorized = fileScanner.nextLine();
String strLinesMemorized[] = tempLinesMemorized.split(",");
for (int i = 0; i < strLinesMemorized.length; i++) {
linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
}
progressOfStudent.setOpenLinesMemorized(linesMemorized);

String tempMistakesMade = fileScanner.nextLine();
String strMistakesMade[] = tempMistakesMade.split(",");
for (int i = 0; i < strMistakesMade.length; i++) {
mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
}
progressOfStudent.setOpenMistakesMade(mistakesMade);

String tempNumOfSaparasFinished = fileScanner.nextLine();
String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
}
progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

String tempNameOfSaparasFinished = fileScanner.nextLine();
String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
}
progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

saparasDone = fileScanner.nextLine();
progressOfStudent.setOpenSaparasDone(saparasDone);

currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

saparaNextFill = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
} else {
String hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
hold = fileScanner.nextLine();
}


//attendance

tempAttendance = fileScanner.nextLine();
String attendance[] = tempAttendance.split(",");
attendanceOfStudent = new Attendance();
for (int i = 0; i < attendance.length; i++ ) {
attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
}
tempReasonAttendance = fileScanner.nextLine();
String tempReason[] = tempReasonAttendance.split(",");
for (int i = 0; i < tempReason.length; i++ ) {
attendanceOfStudent.addReasonAbsent(tempReason[i]);
}
tempCovid = fileScanner.nextLine();
String covid[] = tempCovid.split(",");
for (int i = 0; i < covid.length; i++ ) {
attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
}
tempReasonCovid = fileScanner.nextLine();
String reasonCov[] = tempReasonCovid.split(",");
for (int i = 0; i < reasonCov.length; i++ ) {
attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
}

dates = new ArrayList<String>();

tempDate = fileScanner.nextLine();
String date[] = tempDate.split(",");
for (int i = 0; i < date.length; i++ ) {
dates.add(date[i]);
}

guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
guardianOnePhoneNumber = fileScanner.nextLine();
guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
```

```
                guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
                    guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
                guardianTwoPhoneNumber = fileScanner.nextLine();
                guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
                guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

                emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
                emergencyContactOneHomeNumber = (fileScanner.nextLine());
                emergencyContactOneCellNumber = (fileScanner.nextLine());
                emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoHomeNumber = (fileScanner.nextLine());
                emergencyContactTwoCellNumber = (fileScanner.nextLine());

                healthFactorOne = (fileScanner.nextLine()).toLowerCase();
                healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

                Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,countryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardia-
nOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail, guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardi-
anTwoCallAtWork, emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCellNumber,emergencyContactT-
woFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFac-
torOnePlanOfCareRequired, healthFactorOneMedicationsRequired, healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree,
healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                listOfStudents.add(tempS);
                }
                fileScanner.close();

                return listOfStudents;
                }

                //write to teacher file method
                public static void closeFileTwo() {
                //make new print writer called pw
                PrintWriter pw = null;
                //try and catch statement to assign print writer to file
                try {
                pw = new PrintWriter(new File("teacher.txt"));
                //if file not found then display error and exit program
                } catch (FileNotFoundException e) {
                System.err.print("couldn't open file for writing!");
                System.exit(0);
                }
                //iterate through teacher information array
                for (int y = 0; y < teacherInformation.size(); y++) {
                                //print the teachers information to file
                pw.println(teacherInformation.get(y).getFirstName());
        pw.println(teacherInformation.get(y).getLastName());
                pw.println(teacherInformation.get(y).getPassword());
                }
                //close print writer when done
                pw.close();
                }

        public static void store(int index, String date) {
        PrintWriter pw = null;
                try {
                pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                pw.println(index);
                pw.println(date);
                pw.close();
                } catch (FileNotFoundException e) {
                System.err.print("couldn't open file for writing!");
                System.exit(0);
                }
        }
        public static void store3(String date) {
        PrintWriter pw = null;
                try {
                pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                pw.println(date);
                pw.close();
                } catch (FileNotFoundException e) {
                   System.err.print("couldn't open file for writing!");
                System.exit(0);
                }
        }
        public static void store4(int index) {
        PrintWriter pw = null;
                try {
                pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                pw.println(index);
                pw.close();
                } catch (FileNotFoundException e) {
                System.err.print("couldn't open file for writing!");
                System.exit(0);
                }
        }

        //reading from file for teacher.txt method
        public static ArrayList <Teacher> fileTwoOpen() {
        //try and catch statement to assign a scanner to the text file
                try {
                fileScanner = new Scanner(new File("../marchbreakia/teacher.txt"));
                //print out error and close program if not found
                } catch (FileNotFoundException e) {
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
                }

                //variables to hold information from file
                String fname, lname, password;
                //while scanner has a next line, keep reading from file
                while (fileScanner.hasNextLine()) {


                //read from file line by line and assign each line to its own variable
```

```
                        //these lines are lower cased strings too maintain ambiguity except for password
                        fname = (fileScanner.nextLine()).toLowerCase();
                        lname = (fileScanner.nextLine()).toLowerCase();
                        password = fileScanner.nextLine();
                        //make new teacher object and store with information read from file
                        Teacher tempT = new Teacher(fname, lname, password);
                        teacherInformation.add(tempT);
                        }
                        //close scanner
                        fileScanner.close();
                        //return the teacher information
                        return teacherInformation;
                }

}
```

CLASS: Emergency1.java

```
package com.example.test;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.html.*;
import com.vaadin.flow.component.orderedlayout.Scroller;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.router.Route;
import java.util.*;
import java.time.format.DateTimeFormatter;
import java.time.LocalDateTime;
import java.io.*;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;

@Route(layout = Welcome.class)
public class Emergency1 extends VerticalLayout {

          public static ArrayList <Student> listOfStudents = new ArrayList <Student>();
          public static Scanner fileScanner;

          public Emergency1 () {
                    listOfStudents.removeAll(listOfStudents);
                    listOfStudents = fileOneOpen();
                    int index = index();

                                        // Header
                                        Header header = new Header();
                                        header.getStyle()
                                                .set("align-items", "center")
                                                .set("border-bottom", "1px solid var(--lumo-contrast-20pct)")
                                                .set("display", "flex")
                                                .set("padding", "var(--lumo-space-m)");

                                        //capitalize first letter of student's name and store
                                        String studentName =  listOfStudents.get(index).getFullName();

                                        H2 editEmployee = new H2(studentName + "'s Information");
                                        editEmployee.getStyle().set("margin", "0");

                                        header.add(editEmployee);
                                        add(header);

                                        // Personal information
                                        H3 personalTitle = new H3("Guardian Information");
                                        Section personalInformation = new Section(personalTitle);

                                                Paragraph guardianOne = new Paragraph ("Guardian One: " + listOfStudents.get(index).get-
GuardianOneFirstName()+ " " + listOfStudents.get(index).getGuardianOneLastName());
                                                guardianOne.setWidthFull();
                                                personalInformation.add(guardianOne);
                                                Paragraph phoneOne = new Paragraph ("Phone Number: " + listOfStudents.get(index).get-
GuardianOnePhoneNumber());

                                                phoneOne.setWidthFull();
                                                personalInformation.add(phoneOne);
                                                if(listOfStudents.get(index).isGuardianOneCallAtWork() == false) {
                                            Paragraph callOne = new Paragraph("Note: This guardian does not like to be called at work.");
                                            callOne.setWidthFull();
                                            personalInformation.add(callOne);
                                                }

                                                 if(!(listOfStudents.get(index).getGuardianTwoFirstName().equals("n/a"))) {
                                                        Paragraph guardianTwo = new Paragraph("Guardian Two: " + listOfStudents.get(in-
dex).getGuardianTwoFirstName()+ " " + listOfStudents.get(index).getGuardianTwoLastName());
                                                        guardianTwo.setWidthFull();
                                                        personalInformation.add(guardianTwo);
                                                        Paragraph phoneTwo = new Paragraph("Phone Number: " + listOfStudents.get(in-
dex).getGuardianTwoPhoneNumber());

                                                        phoneTwo.setWidthFull();
                                                        personalInformation.add(phoneTwo);
                                                        if(listOfStudents.get(index).isGuardianTwoCallAtWork() == false) {
                                                        Paragraph callTwo = new Paragraph("Note: This guardian does not like to be called
at work.");

                                                        callTwo.setWidthFull();
                                                        personalInformation.add(callTwo);

                                                        }
                                                        }

                                        // Emergency Contact Information
                                        H3 emergencyContactTitle = new H3("Emergency Contact Information");
```

```
                                        Paragraph contactOne = new Paragraph("Contact One: " + listOfStudents.get(index).getEmergencyCon-
tactOneFirstName() + " " + listOfStudents.get(index).getEmergencyContactOneLastName());
                                        contactOne.setWidthFull();
                                        Paragraph relationshipOne = new Paragraph("Relationship: " + listOfStudents.get(index).getEmer-
gencyContactOneRelationship());
                                        relationshipOne.setWidthFull();
                                        Paragraph homeOne = new Paragraph("Home Number: " + listOfStudents.get(index).getEmergencyContac-
tOneHomeNumber());
                                        homeOne.setWidthFull();
                                        Paragraph cellOne = new Paragraph("Cell Number: " + listOfStudents.get(index).getEmergencyContac-
tOneCellNumber());
                                        cellOne.setWidthFull();
                                        Section emergencyContactInformation = new Section(emergencyContactTitle, contactOne, relation-
shipOne, homeOne, cellOne);

                                        if(!(listOfStudents.get(index).getEmergencyContactTwoFirstName().equals("n/a"))) {
                                              Paragraph contactTwo = new Paragraph("Contact Two: " + listOfStudents.get(index).getEmer-
gencyContactTwoFirstName() + " " + listOfStudents.get(index).getEmergencyContactTwoLastName());
                                              contactTwo.setWidthFull();
                                              emergencyContactInformation.add(contactTwo);
                                              Paragraph relationshipTwo = new Paragraph("Relationship: " + listOfStudents.get(in-
dex).getEmergencyContactTwoRelationship());
                                              relationshipTwo.setWidthFull();
                                              emergencyContactInformation.add(relationshipTwo);
                                              Paragraph homeTwo = new Paragraph("Home Number: " + listOfStudents.get(index).getEmergen-
cyContactTwoHomeNumber());
                                              homeTwo.setWidthFull();
                                              emergencyContactInformation.add(homeTwo);
                                              Paragraph cellTwo = new Paragraph("Cell Number: " + listOfStudents.get(index).getEmergen-
cyContactTwoCellNumber());
                                              cellTwo.setWidthFull();
                                              emergencyContactInformation.add(cellTwo);
                                        }

                                      // Emergency Contact Information
                                        H3 healthInformationTitle = new H3("Health Information");
                                        Section healthInformation = new Section(healthInformationTitle);
                                        if (listOfStudents.get(index).getHealthFactorOne().equals("n/a")) {
                                              Paragraph none = new Paragraph("No health factors were provided.");
                                              none.setWidthFull();
                                              healthInformation.add(none);
                                        }
                                        if(!(listOfStudents.get(index).getHealthFactorOne().equals("n/a"))) {
                                            Paragraph healthOne = new Paragraph("Health Factor 1: " + listOfStudents.get(index).getH-
ealthFactorOne());

                                            healthOne.setWidthFull();
                                            healthInformation.add(healthOne);
                                            if ((listOfStudents.get(index).isHealthFactorOneLifeThreatening()) == true) {
                                              Paragraph threateningOne = new Paragraph("Life Threatening: yes");
                                              threateningOne.setWidthFull();
                                              healthInformation.add(threateningOne);
                                            } else {
                                               Paragraph threateningOne = new Paragraph("Life Threatening: no");
                                                         threateningOne.setWidthFull();
                                                         healthInformation.add(threateningOne);
                                            }
                                            if ((listOfStudents.get(index).isHealthFactorOnePlanOfCareRequired()) == true) {
                                              Paragraph careOne = new Paragraph ("Plan Of Care Required: yes");
                                              careOne.setWidthFull();
                                              healthInformation.add(careOne);
                                            } else {
                                               Paragraph careOne = new Paragraph ("Plan Of Care Required: no");
                                                         careOne.setWidthFull();
                                                         healthInformation.add(careOne);
                                            }
                                            if ((listOfStudents.get(index).isHealthFactorOneMedicationsRequired()) == true) {
                                              Paragraph medicationsOne = new Paragraph("Medications Required: yes");
                                              medicationsOne.setWidthFull();
                                              healthInformation.add(medicationsOne);
                                            } else {
                                               Paragraph medicationsOne = new Paragraph("Medications Required: no");
                                                         medicationsOne.setWidthFull();
                                                         healthInformation.add(medicationsOne);
                                            }

                                          }
                                          if(!(listOfStudents.get(index).getHealthFactorTwo().equals("n/a"))) {
                                                Paragraph healthTwo = new Paragraph("Health Factor 2: " + listOfStudents.get(in-
dex).getHealthFactorTwo());

                                                      healthTwo.setWidthFull();
                                                      healthInformation.add(healthTwo);
                                            if ((listOfStudents.get(index).isHealthFactorTwoLifeThreatening()) == true) {
                                               Paragraph threateningTwo = new Paragraph("Life Threatening: yes");
                                                         threateningTwo.setWidthFull();
                                                         healthInformation.add(threateningTwo);
                                            } else {
                                               Paragraph threateningTwo = new Paragraph("Life Threatening: no");
                                                         threateningTwo.setWidthFull();
                                                         healthInformation.add(threateningTwo);
                                            }
                                            if ((listOfStudents.get(index).isHealthFactorTwoPlanOfCareRequired()) == true) {
                                               Paragraph careTwo = new Paragraph ("Plan Of Care Required: yes");
                                                         careTwo.setWidthFull();
                                                         healthInformation.add(careTwo);
                                            } else {
                                               Paragraph careTwo = new Paragraph ("Plan Of Care Required: no");
                                                         careTwo.setWidthFull();
                                                         healthInformation.add(careTwo);
                                            }
                                            if ((listOfStudents.get(index).isHealthFactorTwoMedicationsRequired()) == true) {
```

```
                Paragraph medicationsTwo = new Paragraph("Medications Required: yes");
                        medicationsTwo.setWidthFull();
                        healthInformation.add(medicationsTwo);
                } else {
                    Paragraph medicationsTwo = new Paragraph("Medications Required: no");
                        medicationsTwo.setWidthFull();
                        healthInformation.add(medicationsTwo);
                }
            }
            if(!(listOfStudents.get(index).getHealthFactorThree().equals("n/a"))) {
                  Paragraph healthThree = new Paragraph("Health Factor 3: " + listOfStudents.get(in-
dex).getHealthFactorThree());
                        healthThree.setWidthFull();
                        healthInformation.add(healthThree);
                if ((listOfStudents.get(index).isHealthFactorThreeLifeThreatening()) == true) {
                    Paragraph threateningThree = new Paragraph("Life Threatening: yes");
                        threateningThree.setWidthFull();
                        healthInformation.add(threateningThree);
                } else {
                    Paragraph threateningThree = new Paragraph("Life Threatening: no");
                        threateningThree.setWidthFull();
                        healthInformation.add(threateningThree);
                }
                if ((listOfStudents.get(index).isHealthFactorThreePlanOfCareRequired()) == true) {
                    Paragraph careThree = new Paragraph ("Plan Of Care Required: yes");
                        careThree.setWidthFull();
                        healthInformation.add(careThree);
                } else {
                    Paragraph careThree = new Paragraph ("Plan Of Care Required: no");
                        careThree.setWidthFull();
                        healthInformation.add(careThree);
                }
                if ((listOfStudents.get(index).isHealthFactorThreeMedicationsRequired()) == true) {
                    Paragraph medicationsThree = new Paragraph("Medications Required: yes");
                        medicationsThree.setWidthFull();
                        healthInformation.add(medicationsThree);
                } else {
                    Paragraph medicationsThree = new Paragraph("Medications Required: no");
                        medicationsThree.setWidthFull();
                        healthInformation.add(medicationsThree);
                }
            }


            Scroller scroller = new Scroller(new Div(personalInformation, emergencyContactInformation,
healthInformation));

            scroller.setScrollDirection(Scroller.ScrollDirection.VERTICAL);
            scroller.getStyle()
                    .set("border-bottom", "1px solid var(--lumo-contrast-20pct)")
                    .set("padding", "var(--lumo-space-m)");
            add(scroller);

            // Footer

            //Done button to navigate back to studentInfo class
            Button done = new Button("Done", e -> {
                  UI.getCurrent().navigate("studentInfo");
            });

            //Adding styling and themes to button for visual appeal
            done.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
            done.getStyle().set("margin-right", "var(--lumo-space-s)");

            //anotherStudent button to open a dialog
            Button anotherStudent = new Button("Another Student", l ->{

                  //making and naming the dialog
                  Dialog dialog = new Dialog();
                  dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                  //opening the dialog and adding it to the view
                  VerticalLayout dialogLayout = createDialogLayout(dialog);
                  dialog.add(dialogLayout);
                  dialog.open();
                  add(dialog);
            });

            //adding theming to anotherStudent button
            anotherStudent.addThemeVariants(ButtonVariant.LUMO_TERTIARY);

            Footer footer = new Footer(done, anotherStudent);
            footer.getStyle().set("padding", "var(--lumo-space-wide-m)");
            add(footer);

            setAlignItems(Alignment.STRETCH);
            //setHeight("400px");
            //setMaxWidth("100%");
            setPadding(false);
            setSpacing(false);
            //setWidth("360px");
            getStyle().set("border", "1px solid var(--lumo-contrast-20pct)");
        }

        //dialog for entering another student's information
    private static VerticalLayout createDialogLayout(Dialog dialog) {
        H2 headline = new H2("Enter Student Information");
        headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                .set("font-size", "1.5em").set("font-weight", "bold");

        TextField firstNameField = new TextField("First Name");
        TextField lastNameField = new TextField("Last Name");
```

```
        VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                lastNameField);
        fieldLayout.setSpacing(false);
        fieldLayout.setPadding(false);
        fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

        Button cancelButton = new Button("Cancel", e -> dialog.close());
        Button saveButton = new Button("Done", e -> {
           int index = -2;
            boolean found = false;
           for (int i = 0; i < listOfStudents.size(); i++) {
                if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOfStu-
dents.get(i).getLastName())) {
                  index = i;
                  found = true;
                  store(index);
                                dialog.close();
                  UI.getCurrent().navigate("emergency1");
                  UI.getCurrent().getPage().reload();
                  break;
                }
              }

              if (found == false) {
                        Notification.show("Invalid name entered.",
                          3000, Notification.Position.MIDDLE);
              }


        });
        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
        buttonLayout
                .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

        VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                buttonLayout);
        dialogLayout.setPadding(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

        return dialogLayout;
    }



          //access stored index of student in temp file
                              public static int index () {
                                        int index = -1;
                                         try {
                                             fileScanner = new Scanner(new File("temp.txt"));
                                             index = Integer.parseInt(fileScanner.nextLine());

                                                    fileScanner.close();
                                               } catch (FileNotFoundException e) {
                                                 System.err.println("File not found! Choosing to quit now...");

                                                 System.exit(0);
                                               }
                                         return index;
                              }

                              //store student index to temp file
                              public static void store(int index) {
                                          PrintWriter pw = null;
                             try {
                                pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                                pw.println(index);
                                pw.close();
                             } catch (FileNotFoundException e) {
                               System.err.print("couldn't open file for writing!");
                               System.exit(0);
                             }
                                }



                               public static ArrayList <Student> fileOneOpen() {
                                      try {
                                        fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
                                      } catch (FileNotFoundException e) {
                                        System.err.println("File not found! Choosing to quit now...");

                                        System.exit(0);
                                      }

                                      //programChosen  - CHECK CONSTRUCTORS
                                      //add health factors to printing out in emergency situation stuff

                                      String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tem-
pAttendance, tempReasonAttendance, tempCovid, tempReasonCovid;

                                       Boolean[] dourDoneOrNot;
                                       Boolean todayDourDoneOrNot;
                                       int[] quarterNumDoneMonth;
                                       int todayQuartersDone, currentQuarter;
                                       Boolean[] numOfDourSaparasDoneMonth;
                                       Boolean todayDourSaparaDoneOrNot;
                                       int todayDourSaparaDone;
```

```
                int dourCurrentSapara, dourNextFill;

                String programChosen;
                String lastRecord;

                Boolean[] sabaqDoneOrNot;
                Boolean todaySabaqDoneOrNot;
                int[] linesMemorized;
                int todayLinesMemorized;
                int[] mistakesMade;
                int todayMistakesMade;
                Boolean[] numOfSaparasDoneMonth;
                Boolean todaySaparaFinished;
                int[] nameOfSaparasDoneMonth;
                int totalSaparasDone;
                int todaySaparaDone;
                String saparasDone;
                int currentSaparaMemorizing;
                int saparaNextFill = 0;

              int age;

              String tempDate;
              ArrayList<String> dates;


              String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
              String guardianOneEmail;
              Boolean guardianOneCallAtWork;
              String guardianTwoFirstName, guardianTwoLastName;
              String guardianTwoPhoneNumber;
              String guardianTwoEmail;
              Boolean guardianTwoCallAtWork;

              String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
              String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
              String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
              String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

              String healthFactorOne;
              Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedica-
tionsRequired;
              String healthFactorTwo;
              Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedica-
tionsRequired;
              String healthFactorThree;
              Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedica-
tionsRequired;


              Attendance attendanceOfStudent;
              StudentProgress progressOfStudent;

              while (fileScanner.hasNextLine()) {
                     dourDoneOrNot = new Boolean[30];
                          quarterNumDoneMonth = new int[30];
                         numOfDourSaparasDoneMonth = new Boolean[30];
                         sabaqDoneOrNot = new Boolean[30];
                          linesMemorized = new int[30];
                          mistakesMade = new int[30];
                          numOfSaparasDoneMonth = new Boolean[30];
                         nameOfSaparasDoneMonth = new int[30];

                firstName = (fileScanner.nextLine()).toLowerCase();
                middleName = (fileScanner.nextLine()).toLowerCase();
                lastName = (fileScanner.nextLine()).toLowerCase();
                address = (fileScanner.nextLine()).toLowerCase();
                dateOfBirth = fileScanner.nextLine();
                age = Integer.parseInt(fileScanner.nextLine());
                postalCode = (fileScanner.nextLine()).toLowerCase();
                language = (fileScanner.nextLine()).toLowerCase();
                countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                //progress of student
                programChosen = (fileScanner.nextLine()).toLowerCase();
                progressOfStudent = new StudentProgress();
                progressOfStudent.setProgramChosen(programChosen);

                lastRecord = (fileScanner.nextLine());
                progressOfStudent.setLastRecord(lastRecord);


                String tempDourDoneOrNot = fileScanner.nextLine();
                String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                for (int i = 0; i < strDourDoneOrNot.length; i++) {
                  dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                }
          progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                String tempQuarterNumDoneMonth = fileScanner.nextLine();
                String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                  quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                }
          progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuar-
ter(currentQuarter);

                String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
```

```
                numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
            }
        progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

            dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCur-
rentSapara(dourCurrentSapara);

            dourNextFill = Integer.parseInt(fileScanner.nextLine());
        progressOfStudent.setOpenDourNextFill(dourNextFill);

            DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
               LocalDateTime firstNow = LocalDateTime.now();
              String alreadyDone = firstFormatter.format(firstNow);

            if (!(alreadyDone.equals(lastRecord))) {
              if (programChosen.equals("hafiz")) {
                 Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 int holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());
                  temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());

                 todayDourDoneOrNot = false;
                 todayDourSaparaDoneOrNot = false;
                 todayQuartersDone = 0;
                 todayDourSaparaDone = 0;
                 progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
        progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
        progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
        progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

               } else {
                 Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 int holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());
                  temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());
                 todayDourDoneOrNot = false;
                  todayQuartersDone = 0;
                  todayDourSaparaDoneOrNot = false;
                 todayDourSaparaDone = 0;
                  todaySabaqDoneOrNot = false;
                 todayLinesMemorized = 0;
                 todayMistakesMade = 0;
                 todaySaparaFinished = false;
                 todaySaparaDone = 0;
                 progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
        progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
        progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
        progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
        progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
        progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
        progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
        progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
        progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
            }


            } else {
              todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
              progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

              todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
              progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

              todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
             progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

             todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
             progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

              if (!(programChosen.equals("hafiz"))) {

            todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.set-
OpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);

                todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
        progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
        progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
        progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
        progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
               } else {
                 Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 int holder = Integer.parseInt(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                 holder = Integer.parseInt(fileScanner.nextLine());
```

```
      }
    }


    if (!(programChosen.equals("hafiz"))) {
      String tempSabaqDoneOrNot = fileScanner.nextLine();
    String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
    for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
      sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
    }
progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

    String tempLinesMemorized = fileScanner.nextLine();
    String strLinesMemorized[] = tempLinesMemorized.split(",");
    for (int i = 0; i < strLinesMemorized.length; i++) {
      linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
    }
    progressOfStudent.setOpenLinesMemorized(linesMemorized);

    String tempMistakesMade = fileScanner.nextLine();
    String strMistakesMade[] = tempMistakesMade.split(",");
    for (int i = 0; i < strMistakesMade.length; i++) {
      mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
    }
    progressOfStudent.setOpenMistakesMade(mistakesMade);

    String tempNumOfSaparasFinished = fileScanner.nextLine();
    String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
    for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
     numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
    }
    progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

    String tempNameOfSaparasFinished = fileScanner.nextLine();
    String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
    for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
     nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
    }
    progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

    totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
    progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

    saparasDone = fileScanner.nextLine();
    progressOfStudent.setOpenSaparasDone(saparasDone);

    currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

    saparaNextFill = Integer.parseInt(fileScanner.nextLine());
    progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
    } else {
      String hold = fileScanner.nextLine();
      hold = fileScanner.nextLine();
      hold = fileScanner.nextLine();
      hold = fileScanner.nextLine();
      hold = fileScanner.nextLine();
      hold = fileScanner.nextLine();
      hold = fileScanner.nextLine();
      hold = fileScanner.nextLine();
      hold = fileScanner.nextLine();
    }


    //attendance

    tempAttendance = fileScanner.nextLine();
    String attendance[] = tempAttendance.split(",");
    attendanceOfStudent = new Attendance();
    for (int i = 0; i < attendance.length; i++ ) {
      attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
      }
    tempReasonAttendance = fileScanner.nextLine();
    String tempReason[] = tempReasonAttendance.split(",");
    for (int i = 0; i < tempReason.length; i++ ) {
      attendanceOfStudent.addReasonAbsent(tempReason[i]);
      }
    tempCovid = fileScanner.nextLine();
    String covid[] = tempCovid.split(",");
    for (int i = 0; i < covid.length; i++ ) {
      attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
      }
    tempReasonCovid = fileScanner.nextLine();
    String reasonCov[] = tempReasonCovid.split(",");
    for (int i = 0; i < reasonCov.length; i++ ) {
      attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
      }

      dates = new ArrayList<String>();

      tempDate = fileScanner.nextLine();
    String date[] = tempDate.split(",");
    for (int i = 0; i < date.length; i++ ) {
      dates.add(date[i]);
      }

      guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
      guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
      guardianOnePhoneNumber = fileScanner.nextLine();
      guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
```

```
                guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
                guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
                guardianTwoPhoneNumber = fileScanner.nextLine();
                guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
                guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

                emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
                emergencyContactOneHomeNumber = (fileScanner.nextLine());
                emergencyContactOneCellNumber = (fileScanner.nextLine());
                emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoHomeNumber = (fileScanner.nextLine());
                emergencyContactTwoCellNumber = (fileScanner.nextLine());

                healthFactorOne = (fileScanner.nextLine()).toLowerCase();
                healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

            Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode,
language,countryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePho-
neNumber,guardianOneEmail, guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardi-
anTwoCallAtWork, emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber,
emergencyContactOneCellNumber,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactT-
woHomeNumber, emergencyContactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorO-
neMedicationsRequired, healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired,
healthFactorThree, healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                listOfStudents.add(tempS);
            }
            fileScanner.close();

            return listOfStudents;
        }

}
```

CLASS: StudentInfo.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.router.Route;

@Route(value = "studentInfo", layout = Welcome.class)
public class StudentInfo extends VerticalLayout {
          static Scanner fileScanner;
          static ArrayList <Student> listOfStudents = new ArrayList <Student>();


          public StudentInfo() {
                    listOfStudents.removeAll(listOfStudents);
                    listOfStudents = fileOneOpen();

                    H2 intro = new H2 ("Student Information");

                    TextField fname = new TextField();
              fname.setLabel("First Name");
              fname.setRequiredIndicatorVisible(true);
              fname.setErrorMessage("This field is required");

              TextField lname = new TextField();
              lname.setLabel("Last Name");
              lname.setRequiredIndicatorVisible(true);
              lname.setErrorMessage("This field is required");
              lname.setHeight("75px");

        Button saveButton = new Button("Done", e -> {
           int index = -2;
            boolean found = false;
           for (int i = 0; i < listOfStudents.size(); i++) {
                if (fname.getValue().equals(listOfStudents.get(i).getFirstName()) && lname.getValue().equals(listOfStudents.get(i).getLast-
Name())) {
                   index = i;
                   found = true;
                   store(index);
```

```
                UI.getCurrent().navigate("emergency1");
                break;
              }
            }

            if (found == false) {
                      Notification.show("Invalid name entered.",
                        3000, Notification.Position.MIDDLE);
            }


        });
        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        saveButton.setWidth("200px");

        setSizeFull();
        setJustifyContentMode(JustifyContentMode.CENTER);
        setDefaultHorizontalComponentAlignment(Alignment.CENTER);
        getStyle().set("text-align", "center");
        add(intro, fname, lname, saveButton);

          }



           public static void store(int index) {
                      PrintWriter pw = null;
          try {
             pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
             pw.println(index);
             pw.close();
          } catch (FileNotFoundException e) {
            System.err.print("couldn't open file for writing!");
            System.exit(0);
          }
             }

             public static ArrayList <Student> fileOneOpen() {
                   try {
                    fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
                  } catch (FileNotFoundException e) {
                    System.err.println("File not found! Choosing to quit now...");

                    System.exit(0);
                  }

                  //programChosen  - CHECK CONSTRUCTORS
                  //add health factors to printing out in emergency situation stuff

                  String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tem-
pReasonAttendance, tempCovid, tempReasonCovid;

                   Boolean[] dourDoneOrNot;
                   Boolean todayDourDoneOrNot;
                   int[] quarterNumDoneMonth;
                   int todayQuartersDone, currentQuarter;
                   Boolean[] numOfDourSaparasDoneMonth;
                   Boolean todayDourSaparaDoneOrNot;
                   int todayDourSaparaDone;
                   int dourCurrentSapara, dourNextFill;

                   String programChosen;
                   String lastRecord;

                   Boolean[] sabaqDoneOrNot;
                   Boolean todaySabaqDoneOrNot;
                   int[] linesMemorized;
                   int todayLinesMemorized;
                   int[] mistakesMade;
                   int todayMistakesMade;
                   Boolean[] numOfSaparasDoneMonth;
                   Boolean todaySaparaFinished;
                   int[] nameOfSaparasDoneMonth;
                   int totalSaparasDone;
                   int todaySaparaDone;
                   String saparasDone;
                   int currentSaparaMemorizing;
                   int saparaNextFill = 0;

                  int age;

                  String tempDate;
                  ArrayList<String> dates;


                  String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
                  String guardianOneEmail;
                  Boolean guardianOneCallAtWork;
                  String guardianTwoFirstName, guardianTwoLastName;
                  String guardianTwoPhoneNumber;
                  String guardianTwoEmail;
                  Boolean guardianTwoCallAtWork;

                  String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
                  String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
                  String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
                  String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

                  String healthFactorOne;
                  Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
                  String healthFactorTwo;
```

```
                Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
                String healthFactorThree;
                Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

                Attendance attendanceOfStudent;
                StudentProgress progressOfStudent;

                while (fileScanner.hasNextLine()) {
                       dourDoneOrNot = new Boolean[30];
                            quarterNumDoneMonth = new int[30];
                           numOfDourSaparasDoneMonth = new Boolean[30];
                           sabaqDoneOrNot = new Boolean[30];
                            linesMemorized = new int[30];
                            mistakesMade = new int[30];
                            numOfSaparasDoneMonth = new Boolean[30];
                           nameOfSaparasDoneMonth = new int[30];

                  firstName = (fileScanner.nextLine()).toLowerCase();
                  middleName = (fileScanner.nextLine()).toLowerCase();
                  lastName = (fileScanner.nextLine()).toLowerCase();
                  address = (fileScanner.nextLine()).toLowerCase();
                  dateOfBirth = fileScanner.nextLine();
                  age = Integer.parseInt(fileScanner.nextLine());
                  postalCode = (fileScanner.nextLine()).toLowerCase();
                  language = (fileScanner.nextLine()).toLowerCase();
                  countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                  //progress of student
                  programChosen = (fileScanner.nextLine()).toLowerCase();
                  progressOfStudent = new StudentProgress();
                  progressOfStudent.setProgramChosen(programChosen);

                  lastRecord = (fileScanner.nextLine());
                  progressOfStudent.setLastRecord(lastRecord);


                  String tempDourDoneOrNot = fileScanner.nextLine();
                  String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                  for (int i = 0; i < strDourDoneOrNot.length; i++) {
                    dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                  }
            progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                  String tempQuarterNumDoneMonth = fileScanner.nextLine();
                  String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                  for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                    quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                  }
            progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                  currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuarter(currentQuarter);

                  String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                  String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                  for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                    numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                  }
            progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                  dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCur-
rentSapara);

                  dourNextFill = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenDourNextFill(dourNextFill);

                  DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                     LocalDateTime firstNow = LocalDateTime.now();
                    String alreadyDone = firstFormatter.format(firstNow);

                  if (!(alreadyDone.equals(lastRecord))) {
                    if (programChosen.equals("hafiz")) {
                      Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      int holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                       temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());

                      todayDourDoneOrNot = false;
                      todayDourSaparaDoneOrNot = false;
                      todayQuartersDone = 0;
                      todayDourSaparaDone = 0;
                      progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                    } else {
                      Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      int holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                       temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      todayDourDoneOrNot = false;
```

```
                    todayQuartersDone = 0;
                    todayDourSaparaDoneOrNot = false;
                   todayDourSaparaDone = 0;
                    todaySabaqDoneOrNot = false;
                   todayLinesMemorized = 0;
                   todayMistakesMade = 0;
                   todaySaparaFinished = false;
                   todaySaparaDone = 0;
                   progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
              progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
              progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
              progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
              progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
              progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
              progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
              progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
              progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                   }


                   } else {
                     todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                     progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                     todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                     progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                     todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                    progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

                    todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                    progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                     if (!(programChosen.equals("hafiz"))) {

                   todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(to-
daySabaqDoneOrNot);

                     todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
              progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                     todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
              progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                     todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
              progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                     todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
              progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                    } else {
                     Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                     int holder = Integer.parseInt(fileScanner.nextLine());
                     holder = Integer.parseInt(fileScanner.nextLine());
                     temporary = Boolean.parseBoolean(fileScanner.nextLine());
                     holder = Integer.parseInt(fileScanner.nextLine());
                    }
                   }



                   if (!(programChosen.equals("hafiz"))) {
                     String tempSabaqDoneOrNot = fileScanner.nextLine();
                   String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
                   for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
                     sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
                   }
              progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

                   String tempLinesMemorized = fileScanner.nextLine();
                   String strLinesMemorized[] = tempLinesMemorized.split(",");
                   for (int i = 0; i < strLinesMemorized.length; i++) {
                     linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
                   }
                   progressOfStudent.setOpenLinesMemorized(linesMemorized);

                   String tempMistakesMade = fileScanner.nextLine();
                   String strMistakesMade[] = tempMistakesMade.split(",");
                   for (int i = 0; i < strMistakesMade.length; i++) {
                     mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
                   }
                   progressOfStudent.setOpenMistakesMade(mistakesMade);

                   String tempNumOfSaparasFinished = fileScanner.nextLine();
                   String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
                   for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
                    numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
                   }
                   progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

                   String tempNameOfSaparasFinished = fileScanner.nextLine();
                   String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
                   for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
                    nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
                   }
                   progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

                   totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
                   progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

                   saparasDone = fileScanner.nextLine();
                   progressOfStudent.setOpenSaparasDone(saparasDone);
```

```
            currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
        progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

            saparaNextFill = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
            } else {
              String hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
            }


            //attendance

            tempAttendance = fileScanner.nextLine();
            String attendance[] = tempAttendance.split(",");
            attendanceOfStudent = new Attendance();
            for (int i = 0; i < attendance.length; i++ ) {
              attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
              }
            tempReasonAttendance = fileScanner.nextLine();
            String tempReason[] = tempReasonAttendance.split(",");
            for (int i = 0; i < tempReason.length; i++ ) {
              attendanceOfStudent.addReasonAbsent(tempReason[i]);
              }
            tempCovid = fileScanner.nextLine();
            String covid[] = tempCovid.split(",");
            for (int i = 0; i < covid.length; i++ ) {
              attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
              }
            tempReasonCovid = fileScanner.nextLine();
            String reasonCov[] = tempReasonCovid.split(",");
            for (int i = 0; i < reasonCov.length; i++ ) {
              attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
              }

              dates = new ArrayList<String>();

              tempDate = fileScanner.nextLine();
            String date[] = tempDate.split(",");
            for (int i = 0; i < date.length; i++ ) {
              dates.add(date[i]);
              }

              guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
              guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
              guardianOnePhoneNumber = fileScanner.nextLine();
              guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
              guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
              guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
              guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
              guardianTwoPhoneNumber = fileScanner.nextLine();
              guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
              guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

              emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
              emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
              emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
              emergencyContactOneHomeNumber = (fileScanner.nextLine());
              emergencyContactOneCellNumber = (fileScanner.nextLine());
              emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
              emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
              emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
              emergencyContactTwoHomeNumber = (fileScanner.nextLine());
              emergencyContactTwoCellNumber = (fileScanner.nextLine());

              healthFactorOne = (fileScanner.nextLine()).toLowerCase();
              healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
              healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorThree = (fileScanner.nextLine()).toLowerCase();
              healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

            Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,coun-
tryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail,
guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardianTwoCallAtWork, emergencyCon-
tactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCell-
Number,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyCon-
tactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired,
healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree,
healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
              listOfStudents.add(tempS);
           }
           fileScanner.close();

           return listOfStudents;
          }
}
```

CLASS: Started.java

```
package com.example.test;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.Route;

@Route("started")
public class started extends VerticalLayout {
    public started() {
          H1 intro = new H1 ("How Are You Feeling Today?");
        intro.setMinWidth("700px");

        Image happy = new Image("images/happy.png", "Mood: Happy");
           happy.setWidth("300px");
           happy.setHeight("280px");
        Button happyButton = new Button(happy, e -> {
           UI.getCurrent().navigate("happyMood");
        });
        happyButton.addThemeVariants(ButtonVariant.LUMO_ICON);
        happyButton.setWidth("300px");
        happyButton.setHeight("280px");

        Image okay = new Image("images/okay.png", "Mood: Okay");
           okay.setWidth("300px");
           okay.setHeight("280px");
        Button okayButton = new Button(okay, e -> {
           UI.getCurrent().navigate("okayMood");
        });
        okayButton.addThemeVariants(ButtonVariant.LUMO_ICON);
        okayButton.setWidth("300px");
        okayButton.setHeight("280px");

        Image sad = new Image("images/sad.png", "Mood: Sad");
           sad.setWidth("300px");
           sad.setHeight("280px");
        Button sadButton = new Button(sad, e -> {
           UI.getCurrent().navigate("sadMood");
        });
        sadButton.addThemeVariants(ButtonVariant.LUMO_ICON);
        sadButton.setWidth("300px");
        sadButton.setHeight("280px");
        HorizontalLayout images = new HorizontalLayout(happyButton, okayButton, sadButton);

        Button cancelButton  = new Button("Back", e-> {
           UI.getCurrent().navigate("emergencyOrNot");
        });
        cancelButton.addThemeVariants(ButtonVariant.LUMO_TERTIARY);

        add(intro, images, cancelButton);

        addClassName("centered-content");
        setSizeFull();
        setJustifyContentMode(JustifyContentMode.CENTER);
        setDefaultHorizontalComponentAlignment(Alignment.CENTER);
        getStyle().set("text-align", "center");
    }

}
```

CLASS: HappyMood.java

```
package com.example.test;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.router.Route;

@Route("happyMood")
public class HappyMood extends VerticalLayout{
          public HappyMood() {
                    Image happy = new Image("images/happy.png", "Mood: Happy");
           happy.setWidth("200px");
           happy.setHeight("185px");
                    H2 intro = new H2 ("Nice to hear that you are having feeling happy! A good dua to read in times of happiness is:");
                    intro.setWidth("600px");
                    Image dua = new Image("images/happyWrite.png", "Mood: Happy");
           dua.setWidth("300px");
           Paragraph meaning = new Paragraph("Subhaanallaah Allaahu 'Akbar\r\n"
                              + "Glory is to Allah. Allah is the Most Great.");
           Button continuee = new Button("Continue", e->{
                    UI.getCurrent().navigate("welcome");
           });
        continuee.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        Button cancelButton  = new Button("Back", e-> {
           UI.getCurrent().navigate("started");
        });
```

```
        cancelButton.addThemeVariants(ButtonVariant.LUMO_TERTIARY);
        HorizontalLayout btns = new HorizontalLayout(cancelButton, continuee);
        btns
                .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                add(happy,intro, dua, meaning, btns);

                addClassName("centered-content");
                setSizeFull();
        setJustifyContentMode(JustifyContentMode.CENTER);
        setDefaultHorizontalComponentAlignment(Alignment.CENTER);
        getStyle().set("text-align", "center");
          }
}
```

CLASS: OkayMood.java

```
package com.example.test;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.router.Route;

@Route("okayMood")
public class OkayMood extends VerticalLayout {
          public OkayMood() {
          Image happy = new Image("images/okay.png", "Mood: Okay");
          happy.setWidth("210px");
          happy.setHeight("197px");
          H2 intro = new H2 (" Having an average day I see! \r\n"
                              + "A good dua to promote happiness and a peaceful life is:");
          intro.setWidth("600px");
          Image dua = new Image("images/okayWrite.png", "Mood: Okay");
          dua.setHeight("100px");
          Paragraph meaning = new Paragraph("Allahummaj alissa adata tagmuruna tunsina ma abkana wama ahzanana O Allah, allow happiness to sur-
round us, making us forget what made us cry and what made us sad");
          meaning.setWidth("520px");
          Button continuee = new Button("Continue", e->{
                    UI.getCurrent().navigate("welcome");
          });
    continuee.addThemeVariants(ButtonVariant.LUMO_PRIMARY, ButtonVariant.LUMO_SUCCESS);
    Button cancelButton  = new Button("Back", e-> {
          UI.getCurrent().navigate("started");
    });
    cancelButton.addThemeVariants(ButtonVariant.LUMO_TERTIARY, ButtonVariant.LUMO_SUCCESS);
    HorizontalLayout btns = new HorizontalLayout(cancelButton, continuee);
    btns
           .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

          add(happy,intro, dua, meaning, btns);

          addClassName("centered-content");
          setSizeFull();
    setJustifyContentMode(JustifyContentMode.CENTER);
    setDefaultHorizontalComponentAlignment(Alignment.CENTER);
    getStyle().set("text-align", "center");

}
}
```

CLASS: SadMood.java

```
package com.example.test;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.Image;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.Route;

@Route("sadMood")
public class SadMood extends VerticalLayout{
          public SadMood() {
          Image sad = new Image("images/sad.png", "Mood: Sad");
          sad.setWidth("194px");
          sad.setHeight("176px");
          H2 intro = new H2 ("Oh no! Having a pretty bad day I see...\r\n"
                              + "A good dua to say in times of hardship and distress is: ");
          intro.setWidth("700px");
          Image dua = new Image("images/sadWrite.png", "Mood: Sad");
          dua.setHeight("80px");
          Paragraph meaning = new Paragraph("Allahumma inni a’udhu bika minal-hammi wal-Ḥuzni wal-‘ajazi wal-kasli wal-bukhli wal-jubni wa
ḍala’id-dayni wa ghalabatir-rijal.\r\n"
                              + "“O Allah, I take refuge in You from anxiety and sorrow, weakness and laziness, miserliness and cowardice, the
burden of debts and from being overpowered by men.” (Bukhari)");
          meaning.setWidth("470px");
          Button continuee = new Button("Continue", e->{
                    UI.getCurrent().navigate("welcome");
          });
          continuee.addThemeVariants(ButtonVariant.LUMO_PRIMARY, ButtonVariant.LUMO_ERROR);
```

```
    Button cancelButton  = new Button("Back", e-> {
           UI.getCurrent().navigate("started");
    });
    cancelButton.addThemeVariants(ButtonVariant.LUMO_TERTIARY, ButtonVariant.LUMO_ERROR);
    HorizontalLayout btns = new HorizontalLayout(cancelButton, continuee);
    btns
            .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

           add(sad,intro, dua, meaning, btns);

           addClassName("centered-content");
           setSizeFull();
    setJustifyContentMode(JustifyContentMode.CENTER);
    setDefaultHorizontalComponentAlignment(Alignment.CENTER);
    getStyle().set("text-align", "center");
}
}
```

CLASS: Welcome.java

```
package com.example.test;

import com.vaadin.flow.component.applayout.AppLayout;
import com.vaadin.flow.component.applayout.DrawerToggle;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.Span;
import com.vaadin.flow.component.icon.Icon;
import com.vaadin.flow.component.icon.VaadinIcon;
import com.vaadin.flow.component.tabs.Tab;
import com.vaadin.flow.component.tabs.Tabs;
import com.vaadin.flow.router.Route;
import com.vaadin.flow.router.RouterLink;

@Route("welcome")
public class Welcome extends AppLayout {

          public Welcome() {

                    DrawerToggle toggle = new DrawerToggle();

              H1 title = new H1("Miftahul Quran Academy");
              title.getStyle()
                .set("font-size", "var(--lumo-font-size-l)")
                .set("margin", "0");

              Tabs tabs = getTabs();

              addToDrawer(tabs);
              addToNavbar(toggle, title);

              H1 intro = new H1 ("Welcome!");
             intro.addClassName("centered-content");
      intro.setSizeFull();
      intro.getStyle().set("text-align", "center");
              setContent(intro);

          }

           private Tabs getTabs() {
                        Tabs tabs = new Tabs();
                        tabs.add(
                          createTab(VaadinIcon.HOME, "Home", 0),
                          createTab(VaadinIcon.BULLETS, "Main Menu", 2),
                          createTab(VaadinIcon.EXCLAMATION_CIRCLE, "Emergency", 1),
                          createTab(VaadinIcon.SIGN_OUT, "Log Out", 7)
                        );
                        tabs.setOrientation(Tabs.Orientation.VERTICAL);
                        return tabs;
                      }

                      private Tab createTab(VaadinIcon viewIcon, String viewName, int i) {
                        Icon icon = viewIcon.create();
                        icon.getStyle()
                                .set("box-sizing", "border-box")
                                .set("margin-inline-end", "var(--lumo-space-m)")
                                .set("margin-inline-start", "var(--lumo-space-xs)")
                                .set("padding", "var(--lumo-space-xs)");

                        if (i == 1) {
                        RouterLink link = new RouterLink();
                        link.add(icon, new Span(viewName));
                        link.setRoute(StudentInfo.class);

                        link.setTabIndex(-1);

                        return new Tab(link);
                        } else if (i == 0) {
                               RouterLink link = new RouterLink();
                                   link.add(icon, new Span(viewName));
                                   link.setRoute(Welcome2.class);

                                   link.setTabIndex(-1);



                                   return new Tab(link);
                        } else if (i == 2){
                               RouterLink link = new RouterLink();
                                   link.add(icon, new Span(viewName));
                                   link.setRoute(Menu.class);

                                   link.setTabIndex(-1);
```

```
                        return new Tab(link);

                    } else {
                        RouterLink link = new RouterLink();
                            link.add(icon, new Span(viewName));
                            link.setRoute(MainView.class);

                            link.setTabIndex(-1);

                            return new Tab(link);
                    }
                    }

}
```

CLASS: Welcome2.java

```
package com.example.test;

import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.Route;

@Route(value = "welcome2", layout = Welcome.class)
public class Welcome2 extends VerticalLayout {

          public Welcome2 () {

                    H1 intro = new H1 ("Welcome!");
                setSizeFull();
                 setJustifyContentMode(JustifyContentMode.CENTER);
                 setDefaultHorizontalComponentAlignment(Alignment.CENTER);
                 getStyle().set("text-align", "center");
                       add(intro);


          }
}
```

CLASS: AttendanceRun.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.checkbox.Checkbox;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H4;
import com.vaadin.flow.component.html.H5;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.router.Route;

import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.Component;
import com.vaadin.flow.component.Focusable;
import com.vaadin.flow.component.Key;
import com.vaadin.flow.component.grid.editor.Editor;
import com.vaadin.flow.data.binder.Binder;


@Route(value = "attendance", layout = Welcome.class)
public class AttendanceRun extends VerticalLayout {
          /**
           *
           */
          private static final long serialVersionUID = 1L;
          static Scanner fileScanner;
          static ArrayList <Student> listOfStudents = new ArrayList <Student>();
          // private Span status;
          public AttendanceRun() {

                    for (int i = 0; i < listOfStudents.size(); i++) {
                              listOfStudents.get(i).setTempAttendance(true);
                              listOfStudents.get(i).setTempScreening(true);
                              listOfStudents.get(i).setTempReason("");
                              listOfStudents.get(i).setTempReason2("");
                    }

                    listOfStudents.removeAll(listOfStudents);
                    listOfStudents = fileOneOpen();
                     //check if attendance for this day has already been done
                int index = -1;
                 DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                 LocalDateTime firstNow2 = LocalDateTime.now(ZoneId.systemDefault());
                String alreadyDoneAttendance = firstFormatter1.format(firstNow2);
```

```
           for (int k = 0; k < listOfStudents.get(0).getDate().size(); k++) {
            if ((listOfStudents.get(0).getDate().get(k)).equals(alreadyDoneAttendance)) {
              index = k;
            }
          }

          if (index != -1) {
          H1 done = new H1("\nAttendance for today is already complete.");
          addClassName("centered-content");
          done.setWidth("500px");

          Button incomplete = new Button("Back", e -> {
                UI.getCurrent().navigate("menu");
          });

              incomplete.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
              incomplete.setMinWidth("250px");
            incomplete.addClickShortcut(Key.ENTER);

            add(done, incomplete);

            setSizeFull();
            setJustifyContentMode(JustifyContentMode.CENTER);
            setDefaultHorizontalComponentAlignment(Alignment.CENTER);
            getStyle().set("text-align", "center");
        } else {
// ValidationMessage lastNameValidationMessage = new ValidationMessage();
// ValidationMessage emailValidationMessage = new ValidationMessage();

 H2 intro = new H2 ("Attendance");
 intro.setMinWidth("700px");
 intro.setSizeFull();
 intro.getStyle().set("text-align", "center");
 add(intro);

 DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
 LocalDateTime firstNow = LocalDateTime.now(ZoneId.systemDefault());
 String alreadyDone = firstFormatter.format(firstNow);

 H4 date = new H4("Today's date is " + alreadyDone);
 add(date);

 Grid<Student> grid = new Grid<>(Student.class, false);
 Grid.Column<Student> nameColumn =
              grid.addColumn(Student::getFullName)
              .setHeader("Student")
         .setWidth("215px").setFlexGrow(0);

 Editor<Student> editor = grid.getEditor();
 Binder<Student> binder = new Binder<>(Student.class);
 editor.setBinder(binder);

 //the grid column for student attendance - check boxes
 Grid.Column<Student> presentOrAbsentColumn = grid.addComponentColumn(
              m_customer -> {
                        //make a new check box
                        Checkbox m_checkbox = new Checkbox();
                        //set value to true for check box and the temporary variable
                        //that holds the screening for that particular student
                        m_checkbox.setValue(true);
                        m_customer.setTempAttendance(true);

                        //if check box value changes listener
                        m_checkbox.addValueChangeListener(event -> {
                                  //if it now equals to false (unselected)
                                  if (m_checkbox.getValue() == false) {
                                            //change the display of the check box to match
                                            m_checkbox.setValue(false);
                                            //change the temporary variable to match
                                            m_customer.setTempAttendance(false);
                                            //if it now equals to true (selected)
                                  } else {
                                            //change the display of the check box to match
                                            m_checkbox.setValue(true);
                                            //change the temporary variable to match
                                            m_customer.setTempAttendance(true);
                                  }

                        });
                        //return the check box to add to grid
                        return m_checkbox;

              }
              //add header and change width
              ).setHeader("Present/Absent").setWidth("30px");



 Grid.Column<Student> reasonAbsent = grid.addColumn(Student::getTempReason)
         .setHeader("Reason for Absence").setWidth("140px").setFlexGrow(1);
 TextField reasonAB = new TextField();
 reasonAB.setWidthFull();
 addCloseHandler(reasonAB, editor);
 binder.forField(reasonAB)
        // .withStatusLabel(lastNameValidationMessage)
         .bind(Student::getTempReason, Student::setTempReason);
 reasonAbsent.setEditorComponent(reasonAB);

//the grid column for COVID screening - check boxes
 Grid.Column<Student> covidScreeningOrNotColumn = grid.addComponentColumn(
              m_customer -> {
```

```
                        //make a new check box
                        Checkbox m_checkbox = new Checkbox();
                        //set value to true for check box and the temporary variable
                        //that holds the screening for that particular student
                        m_checkbox.setValue(true);
                        m_customer.setTempScreening(true);

                        //if check box value changes listener
                        m_checkbox.addValueChangeListener(event -> {

                                  //if it now equals to false (unselected)
                                  if (m_checkbox.getValue() == false) {
                                            //change the display of the check box to match
                                            m_checkbox.setValue(false);
                                            //change the temporary variable to match
                                            m_customer.setTempScreening(false);
                                            //if it now equals to true (selected)
                                  } else {
                                            //change the display of the check box to match
                                            m_checkbox.setValue(true);
                                            //change the temporary variable to match
                                            m_customer.setTempScreening(true);
                                  }

                        });
                        //return the check box to store into the variable
                        return m_checkbox;

              }
              //apply styling
              ).setHeader("COVID Screening").setWidth("50px");

Grid.Column<Student> reasonScreening = grid.addColumn(Student::getTempReason2)
        .setHeader("Reason for Incomplete Screening").setWidth("140px").setFlexGrow(1);

TextField reasonB = new TextField();
reasonB.setWidthFull();
addCloseHandler(reasonB, editor);
binder.forField(reasonB)
       // .withStatusLabel(lastNameValidationMessage)
        .bind(Student::getTempReason2, Student::setTempReason2);
reasonScreening.setEditorComponent(reasonB);

Button cancelButton  = new Button("Back", e-> {
   Dialog dialog = new Dialog();
    dialog.getElement().setAttribute("aria-label", "Unsaved Changes");

    VerticalLayout dialogLayout = createDialogLayout(dialog);
    dialog.add(dialogLayout);
    dialog.open();
    add(dialog);
    listOfStudents.removeAll(listOfStudents);
});

Button save = new Button("Done", e-> {

   for (int i = 0; i < listOfStudents.size();i++) {
             listOfStudents.get(i).addDate(alreadyDone);
             listOfStudents.get(i).addAttendance((listOfStudents.get(i).getTempAttendance()));
             listOfStudents.get(i).addCovidScreening(listOfStudents.get(i).getTempScreening());
             if (listOfStudents.get(i).getTempAttendance() == true) {
                       listOfStudents.get(i).addReasonAbsent("n/a");
             } else {
                       //listOfStudents.get(i).setProgressOfStudentDaily("absent");
            listOfStudents.get(i).setTodaySabaqDoneOrNot(false);
            listOfStudents.get(i).setTodayDourDoneOrNot(false);
            if (listOfStudents.get(i).getTempReason().equals(null)) {
               listOfStudents.get(i).addReasonAbsent("no reason provided.");
            } else {
               listOfStudents.get(i).addReasonAbsent(listOfStudents.get(i).getTempReason());
            }
             }

             if (listOfStudents.get(i).getTempScreening() == true) {
                       listOfStudents.get(i).addReasonCovidScreening("n/a");
             } else {
                       if (listOfStudents.get(i).getTempReason2().equals(null)) {
                                 listOfStudents.get(i).addReasonCovidScreening("no reason provided.");
                       } else {
                                 listOfStudents.get(i).addReasonCovidScreening(listOfStudents.get(i).getTempReason2());
                       }
             }

    }
   for (int y = 0; y < listOfStudents.size(); y++) {

   for (int k = 0; k < listOfStudents.get(y).getReasonCovidScreening().size(); k++) {



   }
   }
   closeFileOne(listOfStudents);
   UI.getCurrent().navigate("menu");
});
cancelButton.addThemeVariants(ButtonVariant.LUMO_TERTIARY);
save.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
HorizontalLayout temp = new HorizontalLayout(cancelButton, save);
temp.setSizeFull();
temp.getStyle().set("text-align", "center");
```

```
        grid.addItemDoubleClickListener(e -> {
            editor.editItem(e.getItem());
            Component editorComponent = e.getColumn().getEditorComponent();
            if (editorComponent instanceof Focusable) {
                ((Focusable) editorComponent).focus();
            }
        });

        /*

        editor.addCancelListener(e -> {
            lastNameValidationMessage.setText("");
            emailValidationMessage.setText("");
        });
*/
        grid.setItems(listOfStudents);

        getThemeList().clear();
        getThemeList().add("spacing-s");
        add(grid, temp);
               }


           }

            private static VerticalLayout createDialogLayout(Dialog dialog) {
                   H2 headline = new H2("Unsaved Changes");
                   headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                           .set("font-size", "1.5em").set("font-weight", "bold");

                   H5 message = new H5("There are unsaved changes. Do you want to continue editing or dicard them?");

                   Button cancelButton = new Button("Discard", e -> {
                      for (int i = 0; i < listOfStudents.size(); i++) {
                                listOfStudents.get(i).setTempReason("");
                                listOfStudents.get(i).setTempReason2("");
                      }
                      UI.getCurrent().navigate("menu");
                      dialog.close();
                      });
                   Button saveButton = new Button("Continue", e -> {
                      dialog.close();
                      });
                   saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                   HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                           saveButton);
                   buttonLayout
                           .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                   VerticalLayout dialogLayout = new VerticalLayout(headline, message,
                           buttonLayout);
                   dialogLayout.setPadding(false);
                   dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                   dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                   return dialogLayout;
               }

            private static void addCloseHandler(Component textField,
                       Editor<Student> editor) {
                   textField.getElement().addEventListener("keydown", e -> editor.cancel())
                           .setFilter("event.code === 'Escape'");
               }

            public static void closeFileOne(ArrayList <Student> listOfStudents) { //COME BACK
                         PrintWriter pw = null;
                               try {
                                  pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                               } catch (FileNotFoundException e) {
                                 System.err.print("couldn't open file for writing!");
                                 System.exit(0);
                               }



                               for (int y = 0; y < listOfStudents.size(); y++) {

                                 if (y == 0) {
                                pw.println(listOfStudents.get(y).getFirstName());
                                 } else {
                                   pw.println(listOfStudents.get(y).getFirstName());
                                 }
                                 pw.println(listOfStudents.get(y).getMiddleName());
                                pw.println(listOfStudents.get(y).getLastName());
                                pw.println(listOfStudents.get(y).getAddress());
                                pw.println(listOfStudents.get(y).getDateOfBirth());
                                pw.println(listOfStudents.get(y).getAge());
                                pw.println(listOfStudents.get(y).getPostalCode());
                                pw.println(listOfStudents.get(y).getLanguage());
                                pw.println(listOfStudents.get(y).getCountryOfBirth());
                           pw.println(listOfStudents.get(y).getProgramChosen());
                                 pw.println(listOfStudents.get(y).getLastRecord());


                                String holder = "";
                               for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
                                    if (k == 0) {
                                holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
                                } else {
                                holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
```

```
      }
      }
        pw.println(holder);

        holder = "";
 for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
       if (k == 0) {
        holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
       } else {
       holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
       }
       }
        pw.println(holder);

pw.println(listOfStudents.get(y).getCurrentQuarter());

holder = "";
        for (int k = 0; k < listOfStudents.get(y).getNumOfDourSaparasDoneMonth().length; k++) {
       if (k == 0) {
        holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
       } else {
       holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
       }
       }
        pw.println(holder);
  pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStudents.get(y).getDourNextFill());

  DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
     LocalDateTime firstNow = LocalDateTime.now();
    String alreadyDone = firstFormatter.format(firstNow);

  if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
      pw.println(false);
      pw.println(-1);
      pw.println(false);
      pw.println(-1);
      pw.println(false);
      pw.println(-1);
      pw.println(-1);
      pw.println(false);
      pw.println(-1);
  } else {
       pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
       pw.println(listOfStudents.get(y).getTodayQuartersDone());
       pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
       pw.println(listOfStudents.get(y).getTodayDourSaparaDone());

    if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
          pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
          pw.println(listOfStudents.get(y).getTodayLinesMemorized());
          pw.println(listOfStudents.get(y).getTodayMistakesMade());
          pw.println(listOfStudents.get(y).isTodaySaparaFinished());
          pw.println(listOfStudents.get(y).getTodaySaparaDone());
    } else {
      pw.println(false);
      pw.println(-1);
      pw.println(-1);
      pw.println(false);
      pw.println(-1);
    }
  }



  if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
        holder = "";
    for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
        if (k == 0) {
         holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
        } else {
        holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
        }
        }
    pw.println(holder);

    holder = "";
  for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
        if (k == 0) {
         holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
        } else {
        holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
        }
        }
  pw.println(holder);

  holder = "";
  for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
        if (k == 0) {
         holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
        } else {
        holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
        }
        }
  pw.println(holder);

  holder = "";
    for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
        if (k == 0) {
         holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
        } else {
        holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
```

```
          }
          }
        pw.println(holder);

        holder = "";
        for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++) {
            if (k == 0) {
             holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
            } else {
            holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
            }
            }
        pw.println(holder);
     pw.println(listOfStudents.get(y).getTotalSaparasDone());
         pw.println(listOfStudents.get(y).getSaparasDone());
pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStudents.get(y).getSaparaNextFill());
      } else {
        pw.println(false);
        pw.println(0);
        pw.println(0);
        pw.println(false);
        pw.println(0);
        pw.println(0);
        pw.println(0);
        pw.println(0);
        pw.println(0);
      }


      //attendance
     //printing to file for attendance
      holder = "";
            for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

            if (k == 0) {
             holder = "" + (listOfStudents.get(y).getAttendance().get(k));

            } else {
            holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));

            }
               }
             pw.println(holder);

             holder = "";
            //printing to file for reason absent
            for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

             if (d == 0) {
              holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

            } else {
            holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

            }
              }
            pw.println(holder);
            //printing to file for covid screening
            holder = "";
            for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

             if (r == 0) {
              holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

            } else {
            holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

            }
          }
            pw.println(holder);

          //printing to file for reason covid screening was not done
            holder = "";
            for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++) {

             if (p == 0) {
              holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

            } else {
            holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreening().get(p));

            }
          }
            pw.println(holder);

          //printing to file for dates
            holder = "";
          for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

             if (z == 0) {
              holder = ""+(listOfStudents.get(y).getDate().get(z));

            } else {
            holder = holder + ("," + listOfStudents.get(y).getDate().get(z));

            }
          }
          pw.println(holder);
          pw.println(listOfStudents.get(y).getGuardianOneFirstName());
          pw.println(listOfStudents.get(y).getGuardianOneLastName());
          pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
```

```
                        pw.println(listOfStudents.get(y).getGuardianOneEmail());
                        pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
                        pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
                        pw.println(listOfStudents.get(y).getGuardianTwoLastName());
                        pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
                        pw.println(listOfStudents.get(y).getGuardianTwoEmail());
                        pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

                        pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

                        pw.println(listOfStudents.get(y).getHealthFactorOne());
                        pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorTwo());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorThree());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                            }
                    pw.close();


                            }


        public static ArrayList <Student> fileOneOpen() {
               try {
                fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
              } catch (FileNotFoundException e) {
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
              }

              //programChosen  - CHECK CONSTRUCTORS
              //add health factors to printing out in emergency situation stuff

             String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tem-
pReasonAttendance, tempCovid, tempReasonCovid;

              Boolean[] dourDoneOrNot;
              Boolean todayDourDoneOrNot;
              int[] quarterNumDoneMonth;
              int todayQuartersDone, currentQuarter;
              Boolean[] numOfDourSaparasDoneMonth;
              Boolean todayDourSaparaDoneOrNot;
              int todayDourSaparaDone;
              int dourCurrentSapara, dourNextFill;

              String programChosen;
              String lastRecord;

              Boolean[] sabaqDoneOrNot;
              Boolean todaySabaqDoneOrNot;
              int[] linesMemorized;
              int todayLinesMemorized;
              int[] mistakesMade;
              int todayMistakesMade;
              Boolean[] numOfSaparasDoneMonth;
              Boolean todaySaparaFinished;
              int[] nameOfSaparasDoneMonth;
              int totalSaparasDone;
              int todaySaparaDone;
              String saparasDone;
              int currentSaparaMemorizing;
              int saparaNextFill = 0;

             int age;

             String tempDate;
             ArrayList<String> dates;


             String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
             String guardianOneEmail;
             Boolean guardianOneCallAtWork;
             String guardianTwoFirstName, guardianTwoLastName;
             String guardianTwoPhoneNumber;
             String guardianTwoEmail;
             Boolean guardianTwoCallAtWork;

             String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
             String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
             String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
             String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

             String healthFactorOne;
```

```
                Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
                String healthFactorTwo;
                Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
                String healthFactorThree;
                Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

                Attendance attendanceOfStudent;
                StudentProgress progressOfStudent;

                while (fileScanner.hasNextLine()) {
                       dourDoneOrNot = new Boolean[30];
                            quarterNumDoneMonth = new int[30];
                           numOfDourSaparasDoneMonth = new Boolean[30];
                           sabaqDoneOrNot = new Boolean[30];
                            linesMemorized = new int[30];
                            mistakesMade = new int[30];
                            numOfSaparasDoneMonth = new Boolean[30];
                           nameOfSaparasDoneMonth = new int[30];

                  firstName = (fileScanner.nextLine()).toLowerCase();
                  middleName = (fileScanner.nextLine()).toLowerCase();
                  lastName = (fileScanner.nextLine()).toLowerCase();
                  address = (fileScanner.nextLine()).toLowerCase();
                  dateOfBirth = fileScanner.nextLine();
                  age = Integer.parseInt(fileScanner.nextLine());
                  postalCode = (fileScanner.nextLine()).toLowerCase();
                  language = (fileScanner.nextLine()).toLowerCase();
                  countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                  //progress of student
                  programChosen = (fileScanner.nextLine()).toLowerCase();
                  progressOfStudent = new StudentProgress();
                  progressOfStudent.setProgramChosen(programChosen);

                  lastRecord = (fileScanner.nextLine());
                  progressOfStudent.setLastRecord(lastRecord);


                  String tempDourDoneOrNot = fileScanner.nextLine();
                  String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                  for (int i = 0; i < strDourDoneOrNot.length; i++) {
                    dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                  }
            progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                  String tempQuarterNumDoneMonth = fileScanner.nextLine();
                  String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                  for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                    quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                  }
            progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                  currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuarter(currentQuarter);

                  String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                  String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                  for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                    numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                  }
            progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                  dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCur-
rentSapara);

                  dourNextFill = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenDourNextFill(dourNextFill);

                  DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                     LocalDateTime firstNow = LocalDateTime.now();
                    String alreadyDone = firstFormatter.format(firstNow);

                  if (!(alreadyDone.equals(lastRecord))) {
                    if (programChosen.equals("hafiz")) {
                      Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      int holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                       temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());

                      todayDourDoneOrNot = false;
                      todayDourSaparaDoneOrNot = false;
                      todayQuartersDone = 0;
                      todayDourSaparaDone = 0;
                      progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                    } else {
                      Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      int holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                       temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
```

```
                holder = Integer.parseInt(fileScanner.nextLine());
                todayDourDoneOrNot = false;
                 todayQuartersDone = 0;
                 todayDourSaparaDoneOrNot = false;
                todayDourSaparaDone = 0;
                 todaySabaqDoneOrNot = false;
                todayLinesMemorized = 0;
                todayMistakesMade = 0;
                todaySaparaFinished = false;
                todaySaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
            progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
            progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                }


                } else {
                  todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                  progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                  todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                  progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                  todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                 progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

                 todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                 progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                  if (!(programChosen.equals("hafiz"))) {

                todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(to-
daySabaqDoneOrNot);

                  todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                  todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                  todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
            progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                  todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                } else {
                  Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                  int holder = Integer.parseInt(fileScanner.nextLine());
                  holder = Integer.parseInt(fileScanner.nextLine());
                  temporary = Boolean.parseBoolean(fileScanner.nextLine());
                  holder = Integer.parseInt(fileScanner.nextLine());
                }
              }



              if (!(programChosen.equals("hafiz"))) {
                String tempSabaqDoneOrNot = fileScanner.nextLine();
              String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
              for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
                sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
              }
            progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

              String tempLinesMemorized = fileScanner.nextLine();
              String strLinesMemorized[] = tempLinesMemorized.split(",");
              for (int i = 0; i < strLinesMemorized.length; i++) {
                linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
              }
              progressOfStudent.setOpenLinesMemorized(linesMemorized);

              String tempMistakesMade = fileScanner.nextLine();
              String strMistakesMade[] = tempMistakesMade.split(",");
              for (int i = 0; i < strMistakesMade.length; i++) {
                mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
              }
              progressOfStudent.setOpenMistakesMade(mistakesMade);

              String tempNumOfSaparasFinished = fileScanner.nextLine();
              String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
              for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
               numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
              }
              progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

              String tempNameOfSaparasFinished = fileScanner.nextLine();
              String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
              for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
               nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
              }
              progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

              totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
              progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);
```

```
            saparasDone = fileScanner.nextLine();
            progressOfStudent.setOpenSaparasDone(saparasDone);

            currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
        progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

            saparaNextFill = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
            } else {
              String hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
              hold = fileScanner.nextLine();
            }


            //attendance

            tempAttendance = fileScanner.nextLine();
            String attendance[] = tempAttendance.split(",");
            attendanceOfStudent = new Attendance();
            for (int i = 0; i < attendance.length; i++ ) {
              attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
              }
            tempReasonAttendance = fileScanner.nextLine();
            String tempReason[] = tempReasonAttendance.split(",");
            for (int i = 0; i < tempReason.length; i++ ) {
              attendanceOfStudent.addReasonAbsent(tempReason[i]);
              }
            tempCovid = fileScanner.nextLine();
            String covid[] = tempCovid.split(",");
            for (int i = 0; i < covid.length; i++ ) {
              attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
              }
            tempReasonCovid = fileScanner.nextLine();
            String reasonCov[] = tempReasonCovid.split(",");
            for (int i = 0; i < reasonCov.length; i++ ) {
              attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
              }

              dates = new ArrayList<String>();

              tempDate = fileScanner.nextLine();
            String date[] = tempDate.split(",");
            for (int i = 0; i < date.length; i++ ) {
              dates.add(date[i]);
              }

              guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
              guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
              guardianOnePhoneNumber = fileScanner.nextLine();
              guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
              guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
              guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
              guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
              guardianTwoPhoneNumber = fileScanner.nextLine();
              guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
              guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

              emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
              emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
              emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
              emergencyContactOneHomeNumber = (fileScanner.nextLine());
              emergencyContactOneCellNumber = (fileScanner.nextLine());
              emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
              emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
              emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
              emergencyContactTwoHomeNumber = (fileScanner.nextLine());
              emergencyContactTwoCellNumber = (fileScanner.nextLine());

              healthFactorOne = (fileScanner.nextLine()).toLowerCase();
              healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
              healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorThree = (fileScanner.nextLine()).toLowerCase();
              healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
              healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

            Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,countryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail, guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardianTwoCallAtWork, emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCellNumber,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired, healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree, healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
              listOfStudents.add(tempS);
          }
          fileScanner.close();

          return listOfStudents;
```

```
            }

            public static void store(int index) {
                        PrintWriter pw = null;
        try {
           pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
           pw.println(index);
           pw.close();
        } catch (FileNotFoundException e) {
          System.err.print("couldn't open file for writing!");
          System.exit(0);
        }
            }

}
```

CLASS: MenuAllStudentT.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.Component;
import com.vaadin.flow.component.Focusable;
import com.vaadin.flow.component.Key;
import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.checkbox.Checkbox;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.grid.editor.Editor;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H4;
import com.vaadin.flow.component.html.H5;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.data.binder.Binder;
import com.vaadin.flow.router.Route;

@Route(value = "menuAllStudentT", layout = Welcome.class)
public class MenuAllStudentT extends VerticalLayout {

           static Scanner fileScanner;
           static ArrayList <Student> listOfStudents = new ArrayList <Student>();

           public MenuAllStudentT() {

                      listOfStudents.removeAll(listOfStudents);
                      listOfStudents = fileOneOpen();
                      for (int i = 0; i < listOfStudents.size(); i++) {
                                 listOfStudents.get(i).setTempAttendance(listOfStudents.get(i).getAttendance().get(listOfStudents.get(i).getAt-
tendance().size()-1));
                                 listOfStudents.get(i).setTempScreening(listOfStudents.get(i).getCovidScreening().get(listOfStu-
dents.get(i).getCovidScreening().size()-1));
                                 listOfStudents.get(i).setTempReason(listOfStudents.get(i).getReasonAbsent().get(listOfStudents.get(i).getRea-
sonAbsent().size()-1));
                                 listOfStudents.get(i).setTempReason2(listOfStudents.get(i).getReasonCovidScreening().get(listOfStu-
dents.get(i).getReasonCovidScreening().size()-1));
                      }
                       //check if attendance for this day has already been done
                    int index = -1;
                     DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                     LocalDateTime firstNow2 = LocalDateTime.now(ZoneId.systemDefault());
                    String alreadyDoneAttendance = firstFormatter1.format(firstNow2);

                    for (int k = 0; k < listOfStudents.get(0).getDate().size(); k++) {
                     if ((listOfStudents.get(0).getDate().get(k)).equals(alreadyDoneAttendance)) {
                       index = k;
                     }
                   }

                   if (index == -1) {
                   H1 done = new H1("\nAttendance for today is incomplete.");
                   addClassName("centered-content");
                   done.setWidth("500px");

                   Button incomplete = new Button("Back", e -> {
                          UI.getCurrent().navigate("menu");
                   });

                       incomplete.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                       incomplete.setMinWidth("250px");
                     incomplete.addClickShortcut(Key.ENTER);

                     add(done, incomplete);

                     setSizeFull();
                     setJustifyContentMode(JustifyContentMode.CENTER);
                     setDefaultHorizontalComponentAlignment(Alignment.CENTER);
                     getStyle().set("text-align", "center");
                  } else {
```

```
        H2 intro = new H2 ("Attendance");
        intro.setMinWidth("700px");
        intro.setSizeFull();
        intro.getStyle().set("text-align", "center");
        add(intro);

        DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDateTime firstNow = LocalDateTime.now(ZoneId.systemDefault());
        String alreadyDone = firstFormatter.format(firstNow);

        H4 date = new H4("Today's date is " + alreadyDone);
        add(date);

        Grid<Student> grid = new Grid<>(Student.class, false);
        Grid.Column<Student> nameColumn =
                    grid.addColumn(Student::getFullName)
                    .setHeader("Student")
                .setWidth("215px").setFlexGrow(0);

        Editor<Student> editor = grid.getEditor();
        Binder<Student> binder = new Binder<>(Student.class);
        editor.setBinder(binder);

        Grid.Column<Student> presentOrAbsentColumn = grid.addComponentColumn(
                    m_customer -> {
                                Checkbox m_checkbox = new Checkbox();
                                m_checkbox.setValue(m_customer.getTempAttendance());

                                m_checkbox.addValueChangeListener(event -> {
                                        if (m_checkbox.getValue() == false) {
                                                m_checkbox.setValue(false);
                                                m_customer.setTempAttendance(false);
                                        } else {
                                                m_checkbox.setValue(true);
                                                m_customer.setTempAttendance(true);
                                        }

                                });
                                return m_checkbox;

                    }
                    ).setHeader("Present/Absent").setWidth("30px");


        Grid.Column<Student> reasonAbsent = grid.addColumn(Student::getTempReason)
                .setHeader("Reason for Absence").setWidth("140px").setFlexGrow(1);
        TextField reasonAB = new TextField();
        reasonAB.setWidthFull();
        addCloseHandler(reasonAB, editor);
        binder.forField(reasonAB)
                .bind(Student::getTempReason, Student::setTempReason);
        reasonAbsent.setEditorComponent(reasonAB);

        Grid.Column<Student> covidScreeningOrNotColumn = grid.addComponentColumn(
                    m_customer -> {
                                Checkbox m_checkbox = new Checkbox();
                                m_checkbox.setValue(m_customer.getTempScreening());

                                m_checkbox.addValueChangeListener(event -> {
                                        if (m_checkbox.getValue() == false) {
                                                m_checkbox.setValue(false);
                                                m_customer.setTempScreening(false);
                                        } else {
                                                m_checkbox.setValue(true);
                                                m_customer.setTempScreening(true);
                                        }

                                });

                                return m_checkbox;

                    }
                    ).setHeader("COVID Screening").setWidth("50px");

        Grid.Column<Student> reasonScreening = grid.addColumn(Student::getTempReason2)
                .setHeader("Reason for Incomplete Screening").setWidth("140px").setFlexGrow(1);

        TextField reasonB = new TextField();
        reasonB.setWidthFull();
        addCloseHandler(reasonB, editor);
        binder.forField(reasonB)
               // .withStatusLabel(lastNameValidationMessage)
                .bind(Student::getTempReason2, Student::setTempReason2);
        reasonScreening.setEditorComponent(reasonB);

        Button cancelButton  = new Button("Back", e-> {
           Dialog dialog = new Dialog();
            dialog.getElement().setAttribute("aria-label", "Unsaved Changes");

            VerticalLayout dialogLayout = createDialogLayout(dialog);
            dialog.add(dialogLayout);
            dialog.open();
            add(dialog);
            listOfStudents.removeAll(listOfStudents);
        });

        Button save = new Button("Done", e-> {
           for (int i = 0; i < listOfStudents.size();i++) {
                    listOfStudents.get(i).getAttendance().set((listOfStudents.get(i).getAttendance().size()-1),(listOfStudents.get(i).getTem-
pAttendance()));
```

```
                    listOfStudents.get(i).getCovidScreening().set((listOfStudents.get(i).getCovidScreening().size()-1) ,(listOfStu-
dents.get(i).getTempScreening()));
                    if (listOfStudents.get(i).getTempAttendance() == true) {
                              listOfStudents.get(i).getReasonAbsent().set((listOfStudents.get(i).getReasonAbsent().size()-1),("n/a"));
                    } else {
                    //        listOfStudents.get(i).setProgressOfStudentDaily("absent");
                    listOfStudents.get(i).setTodaySabaqDoneOrNot(false);
                    listOfStudents.get(i).setTodayDourDoneOrNot(false);
                    if (listOfStudents.get(i).getTempReason().equals("")) {
                      listOfStudents.get(i).getReasonAbsent().set((listOfStudents.get(i).getReasonAbsent().size()-1) ,("no reason pro-
vided."));
                    } else {
                      listOfStudents.get(i).getReasonAbsent().set((listOfStudents.get(i).getReasonAbsent().size()-1) ,(listOfStu-
dents.get(i).getTempReason()));
                    }
                    }

                    if (listOfStudents.get(i).getTempScreening() == true) {
                              listOfStudents.get(i).getReasonCovidScreening().set((listOfStudents.get(i).getReasonCovidScreening().size()-
1) ,("n/a"));
                    } else {
                              if (listOfStudents.get(i).getTempReason2().equals("")) {
                                        listOfStudents.get(i).getReasonCovidScreening().set((listOfStudents.get(i).getReasonCovidScreen-
ing().size()-1),("no reason provided."));
                              } else {
                                        listOfStudents.get(i).getReasonCovidScreening().set((listOfStudents.get(i).getReasonCovidScreen-
ing().size()-1) , (listOfStudents.get(i).getTempReason2()));
                              }
                    }

          }
          closeFileOne(listOfStudents);
          UI.getCurrent().navigate("menu");
        });
        cancelButton.addThemeVariants(ButtonVariant.LUMO_TERTIARY);
        save.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        HorizontalLayout temp = new HorizontalLayout(cancelButton, save);
        temp.setSizeFull();
        temp.getStyle().set("text-align", "center");


        grid.addItemDoubleClickListener(e -> {
            editor.editItem(e.getItem());
            Component editorComponent = e.getColumn().getEditorComponent();
            if (editorComponent instanceof Focusable) {
                ((Focusable) editorComponent).focus();
            }
        });
        grid.setItems(listOfStudents);

        getThemeList().clear();
        getThemeList().add("spacing-s");
        add(grid, temp);
              }
          }

            private static VerticalLayout createDialogLayout(Dialog dialog) {
                  H2 headline = new H2("Unsaved Changes");
                  headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                          .set("font-size", "1.5em").set("font-weight", "bold");

                  H5 message = new H5("There are unsaved changes. Do you want to continue editing or dicard them?");

                  Button cancelButton = new Button("Discard", e -> {
                     for (int i = 0; i < listOfStudents.size(); i++) {
                              listOfStudents.get(i).setTempReason("");
                              listOfStudents.get(i).setTempReason2("");
                     }
                     UI.getCurrent().navigate("menu");
                     dialog.close();
                     });
                  Button saveButton = new Button("Continue", e -> {
                     dialog.close();
                     });
                  saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                  HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                          saveButton);
                  buttonLayout
                          .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                  VerticalLayout dialogLayout = new VerticalLayout(headline, message,
                          buttonLayout);
                  dialogLayout.setPadding(false);
                  dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                  dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                  return dialogLayout;
              }

            private static void addCloseHandler(Component textField,
                    Editor<Student> editor) {
                  textField.getElement().addEventListener("keydown", e -> editor.cancel())
                          .setFilter("event.code === 'Escape'");
              }

            public static void closeFileOne(ArrayList <Student> listOfStudents) {
                        PrintWriter pw = null;
                              try {
                                 pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                              } catch (FileNotFoundException e) {
                                System.err.print("couldn't open file for writing!");
```

```
            System.exit(0);
          }


          for (int y = 0; y < listOfStudents.size(); y++) {

            if (y == 0) {
           pw.println(listOfStudents.get(y).getFirstName());
            } else {
              pw.println(listOfStudents.get(y).getFirstName());
            }
            pw.println(listOfStudents.get(y).getMiddleName());
           pw.println(listOfStudents.get(y).getLastName());
           pw.println(listOfStudents.get(y).getAddress());
           pw.println(listOfStudents.get(y).getDateOfBirth());
           pw.println(listOfStudents.get(y).getAge());
           pw.println(listOfStudents.get(y).getPostalCode());
           pw.println(listOfStudents.get(y).getLanguage());
           pw.println(listOfStudents.get(y).getCountryOfBirth());
pw.println(listOfStudents.get(y).getProgramChosen());
            pw.println(listOfStudents.get(y).getLastRecord());


       String holder = "";
      for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
           if (k == 0) {
 holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
          } else {
          holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
          }
          }
            pw.println(holder);

            holder = "";
    for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
           if (k == 0) {
            holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
          } else {
          holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
          }
          }
            pw.println(holder);

   pw.println(listOfStudents.get(y).getCurrentQuarter());

   holder = "";
           for (int k = 0; k < listOfStudents.get(y).getNumOfDourSaparasDoneMonth().length; k++) {
           if (k == 0) {
            holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
          } else {
          holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
          }
          }
            pw.println(holder);
     pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStudents.get(y).getDourNextFill());

    DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
       LocalDateTime firstNow = LocalDateTime.now();
      String alreadyDone = firstFormatter.format(firstNow);

    if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
        pw.println(false);
        pw.println(-1);
        pw.println(false);
        pw.println(-1);
        pw.println(false);
        pw.println(-1);
        pw.println(-1);
        pw.println(false);
        pw.println(-1);
    } else {
         pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
         pw.println(listOfStudents.get(y).getTodayQuartersDone());
         pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
         pw.println(listOfStudents.get(y).getTodayDourSaparaDone());

      if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
            pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
            pw.println(listOfStudents.get(y).getTodayLinesMemorized());
            pw.println(listOfStudents.get(y).getTodayMistakesMade());
            pw.println(listOfStudents.get(y).isTodaySaparaFinished());
            pw.println(listOfStudents.get(y).getTodaySaparaDone());
      } else {
        pw.println(false);
        pw.println(-1);
        pw.println(-1);
        pw.println(false);
        pw.println(-1);
      }
    }


    if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
          holder = "";
      for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
          if (k == 0) {
           holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
          } else {
          holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
```

```
            }
            }
          pw.println(holder);

          holder = "";
        for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
              }
              }
        pw.println(holder);

        holder = "";
        for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
              }
              }
        pw.println(holder);

        holder = "";
          for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
              }
              }
          pw.println(holder);

          holder = "";
          for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
              }
              }
          pw.println(holder);
        pw.println(listOfStudents.get(y).getTotalSaparasDone());
            pw.println(listOfStudents.get(y).getSaparasDone());
pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStudents.get(y).getSaparaNextFill());
        } else {
          pw.println(false);
          pw.println(0);
          pw.println(0);
          pw.println(false);
          pw.println(0);
          pw.println(0);
          pw.println(0);
          pw.println(0);
          pw.println(0);
        }


        //attendance
       //printing to file for attendance
        holder = "";
              for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getAttendance().get(k));

              } else {
              holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));

              }
                 }
               pw.println(holder);

               holder = "";
              //printing to file for reason absent
              for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

               if (d == 0) {
                holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

              } else {
              holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

              }
                }
              pw.println(holder);
              //printing to file for covid screening
              holder = "";
              for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

               if (r == 0) {
                holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

              } else {
              holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

              }
            }
              pw.println(holder);
```

```
                        //printing to file for reason covid screening was not done
                          holder = "";
                          for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++) {

                           if (p == 0) {
                            holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

                          } else {
                          holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreening().get(p));

                          }
                        }
                          pw.println(holder);

                        //printing to file for dates
                          holder = "";
                        for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

                           if (z == 0) {
                            holder = ""+(listOfStudents.get(y).getDate().get(z));

                          } else {
                          holder = holder + ("," + listOfStudents.get(y).getDate().get(z));

                          }
                        }
                        pw.println(holder);
                        pw.println(listOfStudents.get(y).getGuardianOneFirstName());
                        pw.println(listOfStudents.get(y).getGuardianOneLastName());
                        pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
                        pw.println(listOfStudents.get(y).getGuardianOneEmail());
                        pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
                        pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
                        pw.println(listOfStudents.get(y).getGuardianTwoLastName());
                        pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
                        pw.println(listOfStudents.get(y).getGuardianTwoEmail());
                        pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

                        pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

                        pw.println(listOfStudents.get(y).getHealthFactorOne());
                        pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorTwo());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorThree());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                              }
                    pw.close();


                              }


        public static ArrayList <Student> fileOneOpen() {
               try {
                fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
              } catch (FileNotFoundException e) {
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
              }

              //programChosen  - CHECK CONSTRUCTORS
              //add health factors to printing out in emergency situation stuff

             String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tem-
pReasonAttendance, tempCovid, tempReasonCovid;

              Boolean[] dourDoneOrNot;
              Boolean todayDourDoneOrNot;
              int[] quarterNumDoneMonth;
              int todayQuartersDone, currentQuarter;
              Boolean[] numOfDourSaparasDoneMonth;
              Boolean todayDourSaparaDoneOrNot;
              int todayDourSaparaDone;
              int dourCurrentSapara, dourNextFill;

              String programChosen;
              String lastRecord;

              Boolean[] sabaqDoneOrNot;
              Boolean todaySabaqDoneOrNot;
              int[] linesMemorized;
              int todayLinesMemorized;
              int[] mistakesMade;
```

```
                int todayMistakesMade;
                Boolean[] numOfSaparasDoneMonth;
                Boolean todaySaparaFinished;
                int[] nameOfSaparasDoneMonth;
                int totalSaparasDone;
                int todaySaparaDone;
                String saparasDone;
                int currentSaparaMemorizing;
                int saparaNextFill = 0;

               int age;

               String tempDate;
               ArrayList<String> dates;


               String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
               String guardianOneEmail;
               Boolean guardianOneCallAtWork;
               String guardianTwoFirstName, guardianTwoLastName;
               String guardianTwoPhoneNumber;
               String guardianTwoEmail;
               Boolean guardianTwoCallAtWork;

               String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
               String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
               String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
               String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

               String healthFactorOne;
               Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
               String healthFactorTwo;
               Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
               String healthFactorThree;
               Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

               Attendance attendanceOfStudent;
               StudentProgress progressOfStudent;

               while (fileScanner.hasNextLine()) {
                      dourDoneOrNot = new Boolean[30];
                           quarterNumDoneMonth = new int[30];
                          numOfDourSaparasDoneMonth = new Boolean[30];
                          sabaqDoneOrNot = new Boolean[30];
                           linesMemorized = new int[30];
                           mistakesMade = new int[30];
                           numOfSaparasDoneMonth = new Boolean[30];
                          nameOfSaparasDoneMonth = new int[30];

                 firstName = (fileScanner.nextLine()).toLowerCase();
                 middleName = (fileScanner.nextLine()).toLowerCase();
                 lastName = (fileScanner.nextLine()).toLowerCase();
                 address = (fileScanner.nextLine()).toLowerCase();
                 dateOfBirth = fileScanner.nextLine();
                 age = Integer.parseInt(fileScanner.nextLine());
                 postalCode = (fileScanner.nextLine()).toLowerCase();
                 language = (fileScanner.nextLine()).toLowerCase();
                 countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                 //progress of student
                 programChosen = (fileScanner.nextLine()).toLowerCase();
                 progressOfStudent = new StudentProgress();
                 progressOfStudent.setProgramChosen(programChosen);

                 lastRecord = (fileScanner.nextLine());
                 progressOfStudent.setLastRecord(lastRecord);


                 String tempDourDoneOrNot = fileScanner.nextLine();
                 String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                 for (int i = 0; i < strDourDoneOrNot.length; i++) {
                   dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                 }
            progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                 String tempQuarterNumDoneMonth = fileScanner.nextLine();
                 String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                 for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                   quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                 }
            progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                 currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuarter(currentQuarter);

                 String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                 String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                 for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                   numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                 }
            progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                 dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCur-
rentSapara);

                 dourNextFill = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenDourNextFill(dourNextFill);

                 DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                    LocalDateTime firstNow = LocalDateTime.now();
                   String alreadyDone = firstFormatter.format(firstNow);
```

```
                    if (!(alreadyDone.equals(lastRecord))) {
                      if (programChosen.equals("hafiz")) {
                        Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        int holder = Integer.parseInt(fileScanner.nextLine());
                        temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                         temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                        temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());

                        todayDourDoneOrNot = false;
                        todayDourSaparaDoneOrNot = false;
                        todayQuartersDone = 0;
                        todayDourSaparaDone = 0;
                        progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
                    progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
                    progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
                    progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                      } else {
                        Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        int holder = Integer.parseInt(fileScanner.nextLine());
                        temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                         temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                        temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                        todayDourDoneOrNot = false;
                         todayQuartersDone = 0;
                         todayDourSaparaDoneOrNot = false;
                        todayDourSaparaDone = 0;
                         todaySabaqDoneOrNot = false;
                        todayLinesMemorized = 0;
                        todayMistakesMade = 0;
                        todaySaparaFinished = false;
                        todaySaparaDone = 0;
                        progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
                    progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
                    progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
                    progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
                    progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
                    progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
                    progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
                    progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
                    progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                    }


                    } else {
                      todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                      progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                      todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                      progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                      todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                     progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

                     todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                     progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                      if (!(programChosen.equals("hafiz"))) {

                    todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(to-
daySabaqDoneOrNot);

                        todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
                    progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                        todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
                    progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                        todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
                    progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                        todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
                    progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                      } else {
                        Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        int holder = Integer.parseInt(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                        temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                      }
                    }



                    if (!(programChosen.equals("hafiz"))) {
                      String tempSabaqDoneOrNot = fileScanner.nextLine();
                    String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
                    for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
                      sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
                    }
                    progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

                    String tempLinesMemorized = fileScanner.nextLine();
```

```
      String strLinesMemorized[] = tempLinesMemorized.split(",");
      for (int i = 0; i < strLinesMemorized.length; i++) {
        linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
      }
      progressOfStudent.setOpenLinesMemorized(linesMemorized);

      String tempMistakesMade = fileScanner.nextLine();
      String strMistakesMade[] = tempMistakesMade.split(",");
      for (int i = 0; i < strMistakesMade.length; i++) {
        mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
      }
      progressOfStudent.setOpenMistakesMade(mistakesMade);

      String tempNumOfSaparasFinished = fileScanner.nextLine();
      String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
      for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
       numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

      String tempNameOfSaparasFinished = fileScanner.nextLine();
      String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
      for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
       nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

      totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

      saparasDone = fileScanner.nextLine();
      progressOfStudent.setOpenSaparasDone(saparasDone);

      currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

      saparaNextFill = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
      } else {
        String hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
      }


      //attendance

      tempAttendance = fileScanner.nextLine();
      String attendance[] = tempAttendance.split(",");
      attendanceOfStudent = new Attendance();
      for (int i = 0; i < attendance.length; i++ ) {
        attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
        }
      tempReasonAttendance = fileScanner.nextLine();
      String tempReason[] = tempReasonAttendance.split(",");
      for (int i = 0; i < tempReason.length; i++ ) {
        attendanceOfStudent.addReasonAbsent(tempReason[i]);
        }
      tempCovid = fileScanner.nextLine();
      String covid[] = tempCovid.split(",");
      for (int i = 0; i < covid.length; i++ ) {
        attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
        }
      tempReasonCovid = fileScanner.nextLine();
      String reasonCov[] = tempReasonCovid.split(",");
      for (int i = 0; i < reasonCov.length; i++ ) {
        attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
        }

        dates = new ArrayList<String>();

        tempDate = fileScanner.nextLine();
      String date[] = tempDate.split(",");
      for (int i = 0; i < date.length; i++ ) {
        dates.add(date[i]);
        }

        guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
        guardianOnePhoneNumber = fileScanner.nextLine();
        guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
        guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
        guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoPhoneNumber = fileScanner.nextLine();
        guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
        guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

        emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneHomeNumber = (fileScanner.nextLine());
        emergencyContactOneCellNumber = (fileScanner.nextLine());
        emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
```

```
                    emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactTwoHomeNumber = (fileScanner.nextLine());
                    emergencyContactTwoCellNumber = (fileScanner.nextLine());

                    healthFactorOne = (fileScanner.nextLine()).toLowerCase();
                    healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                    healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                    healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

                  Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,coun-
tryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail,
guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardianTwoCallAtWork, emergencyCon-
tactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCell-
Number,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyCon-
tactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired,
healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree,
healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                    listOfStudents.add(tempS);
                  }
                  fileScanner.close();

                  return listOfStudents;
                 }

             public static void store(int index) {
                        PrintWriter pw = null;
     try {
        pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
        pw.println(index);
        pw.close();
     } catch (FileNotFoundException e) {
       System.err.print("couldn't open file for writing!");
       System.exit(0);
     }
            }
}
```

CLASS: MenuAllStudentP.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.Component;
import com.vaadin.flow.component.Focusable;
import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.checkbox.Checkbox;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.grid.Grid;
import com.vaadin.flow.component.grid.editor.Editor;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H4;
import com.vaadin.flow.component.html.H5;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.data.binder.Binder;
import com.vaadin.flow.router.Route;

@Route(value = "menuAllStudentP", layout = Welcome.class)
public class MenuAllStudentP extends VerticalLayout {

           static Scanner fileScanner;
           static ArrayList <Student> listOfStudents = new ArrayList <Student>();
           static String date;

           public MenuAllStudentP() {

                    //read from files
                    listOfStudents.removeAll(listOfStudents);
                    listOfStudents = fileOneOpen();
                    date = info();

                     H2 intro = new H2 ("Attendance");
                   intro.setMinWidth("700px");
                   intro.setSizeFull();
                   intro.getStyle().set("text-align", "center");
                   add(intro);

                   H4 date3 = new H4("Date: " + date);
                   add(date3);

                   ArrayList <Student> tempS = new ArrayList <Student>();
                   tempS.removeAll(tempS);
                   ArrayList <Integer> index = new ArrayList <Integer> ();
```

```
        ArrayList <Integer> indexOfDate = new ArrayList <Integer> ();

        for (int i = 0; i < listOfStudents.size(); i++) {
           for (int k = 0; k < listOfStudents.get(i).getDate().size(); k++) {
             if (listOfStudents.get(i).getDate().get(k).equals(date)) {
                        tempS.add(listOfStudents.get(i));
                        index.add(i);
                        indexOfDate.add(k);
             }
  }
}


        for (int i = 0; i < tempS.size(); i++) {
                                tempS.get(i).setTempAttendance(tempS.get(i).getAttendance().get(indexOfDate.get(i)));
                                tempS.get(i).setTempScreening(tempS.get(i).getCovidScreening().get(indexOfDate.get(i)));
                                tempS.get(i).setTempReason(tempS.get(i).getReasonAbsent().get(indexOfDate.get(i)));
                                tempS.get(i).setTempReason2(tempS.get(i).getReasonCovidScreening().get(indexOfDate.get(i)));
                    }

        Grid<Student> grid = new Grid<>(Student.class, false);
        Grid.Column<Student> nameColumn =
                    grid.addColumn(Student::getFullName)
                    .setHeader("Student")
                .setWidth("215px").setFlexGrow(0);

        Editor<Student> editor = grid.getEditor();
        Binder<Student> binder = new Binder<>(Student.class);
        editor.setBinder(binder);

        Grid.Column<Student> presentOrAbsentColumn = grid.addComponentColumn(
                    m_customer -> {
                                Checkbox m_checkbox = new Checkbox();
                                m_checkbox.setValue(m_customer.getTempAttendance());

                                m_checkbox.addValueChangeListener(event -> {
                                            if (m_checkbox.getValue() == false) {
                                                        m_checkbox.setValue(false);
                                                        m_customer.setTempAttendance(false);
                                            } else {
                                                        m_checkbox.setValue(true);
                                                        m_customer.setTempAttendance(true);
                                            }

                                });
                                return m_checkbox;

                    }
                    ).setHeader("Present/Absent").setWidth("30px");


        Grid.Column<Student> reasonAbsent = grid.addColumn(Student::getTempReason)
                .setHeader("Reason for Absence").setWidth("140px").setFlexGrow(1);
        TextField reasonAB = new TextField();
        reasonAB.setWidthFull();
        addCloseHandler(reasonAB, editor);
        binder.forField(reasonAB)
                .bind(Student::getTempReason, Student::setTempReason);
        reasonAbsent.setEditorComponent(reasonAB);

        Grid.Column<Student> covidScreeningOrNotColumn = grid.addComponentColumn(
                    m_customer -> {
                                Checkbox m_checkbox = new Checkbox();
                                m_checkbox.setValue(m_customer.getTempScreening());

                                m_checkbox.addValueChangeListener(event -> {
                                            if (m_checkbox.getValue() == false) {
                                                        m_checkbox.setValue(false);
                                                        m_customer.setTempScreening(false);
                                            } else {
                                                        m_checkbox.setValue(true);
                                                        m_customer.setTempScreening(true);
                                            }

                                });

                                return m_checkbox;

                    }
                    ).setHeader("COVID Screening").setWidth("50px");

        Grid.Column<Student> reasonScreening = grid.addColumn(Student::getTempReason2)
                .setHeader("Reason for Incomplete Screening").setWidth("140px").setFlexGrow(1);

        TextField reasonB = new TextField();
        reasonB.setWidthFull();
        addCloseHandler(reasonB, editor);
        binder.forField(reasonB)
                .bind(Student::getTempReason2, Student::setTempReason2);
        reasonScreening.setEditorComponent(reasonB);

        Button cancelButton  = new Button("Back", e-> {
           Dialog dialog = new Dialog();
            dialog.getElement().setAttribute("aria-label", "Unsaved Changes");

            VerticalLayout dialogLayout = createDialogLayout(dialog);
            dialog.add(dialogLayout);
            dialog.open();
            add(dialog);
            tempS.removeAll(tempS);
            listOfStudents.removeAll(listOfStudents);
```

```
                });

                Button save = new Button("Done", e-> {
                   for (int i = 0; i < tempS.size();i++) {
                             tempS.get(i).getAttendance().set((indexOfDate.get(i)),(tempS.get(i).getTempAttendance()));
                             tempS.get(i).getCovidScreening().set((indexOfDate.get(i)) ,(tempS.get(i).getTempScreening()));
                             if (tempS.get(i).getTempAttendance() == true) {
                                       tempS.get(i).getReasonAbsent().set((indexOfDate.get(i)),("n/a"));
                             } else {
                                       //tempS.get(i).setProgressOfStudentDaily("absent");
                                       tempS.get(i).setTodaySabaqDoneOrNot(false);
                                       tempS.get(i).setTodayDourDoneOrNot(false);
                                      if (tempS.get(i).getTempReason().equals("")) {
                                         tempS.get(i).getReasonAbsent().set((indexOfDate.get(i)) ,("no reason provided."));
                                      } else {
                                         tempS.get(i).getReasonAbsent().set((indexOfDate.get(i)) ,(tempS.get(i).getTempReason()));
                                      }
                             }

                             if (tempS.get(i).getTempScreening() == true) {
                                       tempS.get(i).getReasonCovidScreening().set((indexOfDate.get(i)) ,("n/a"));
                             } else {
                                       if (tempS.get(i).getTempReason2().equals("")) {
                                                 tempS.get(i).getReasonCovidScreening().set((indexOfDate.get(i)),("no reason provided."));
                                       } else {
                                                 tempS.get(i).getReasonCovidScreening().set((indexOfDate.get(i)) , (tempS.get(i).getTem-
pReason2()));
                                       }
                             }

                    }

                   for (int i = 0; i < listOfStudents.size(); i++) {
                             for (int k = 0; k < index.size(); k++) {
                                       if (index.get(k) == i) {
                                                 listOfStudents.set(i, tempS.get(k));
                                       }
                             }
                   }
                   closeFileOne(listOfStudents);
                   UI.getCurrent().navigate("menu");
                });
                cancelButton.addThemeVariants(ButtonVariant.LUMO_TERTIARY);
                save.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout temp = new HorizontalLayout(cancelButton, save);
                temp.setSizeFull();
                temp.getStyle().set("text-align", "center");


                grid.addItemDoubleClickListener(e -> {
                    editor.editItem(e.getItem());
                    Component editorComponent = e.getColumn().getEditorComponent();
                    if (editorComponent instanceof Focusable) {
                        ((Focusable) editorComponent).focus();
                    }
                });
                grid.setItems(tempS);

                getThemeList().clear();
                getThemeList().add("spacing-s");
                add(grid, temp);
                      }



          //access stored index of student in temp file
                             public static String info () {
                                       date = "";
                                       try {
                                            fileScanner = new Scanner(new File("temp.txt"));
                                            date = fileScanner.nextLine();
                                              fileScanner.close();
                                          } catch (FileNotFoundException e) {
                                            System.err.println("File not found! Choosing to quit now...");

                                            System.exit(0);
                                          }
                                       return date;
                             }


                              public static void closeFileOne(ArrayList <Student> listOfStudents) {
                                           PrintWriter pw = null;
                                                 try {
                                                    pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                                                 } catch (FileNotFoundException e) {
                                                   System.err.print("couldn't open file for writing!");
                                                   System.exit(0);
                                                 }



                                                 for (int y = 0; y < listOfStudents.size(); y++) {

                                                   if (y == 0) {
                                                  pw.println(listOfStudents.get(y).getFirstName());
                                                   } else {
                                                     pw.println(listOfStudents.get(y).getFirstName());
                                                   }
                                                   pw.println(listOfStudents.get(y).getMiddleName());
                                                  pw.println(listOfStudents.get(y).getLastName());
```

```
                pw.println(listOfStudents.get(y).getAddress());
                pw.println(listOfStudents.get(y).getDateOfBirth());
                pw.println(listOfStudents.get(y).getAge());
                pw.println(listOfStudents.get(y).getPostalCode());
                pw.println(listOfStudents.get(y).getLanguage());
                pw.println(listOfStudents.get(y).getCountryOfBirth());
pw.println(listOfStudents.get(y).getProgramChosen());
                 pw.println(listOfStudents.get(y).getLastRecord());


             String holder = "";
            for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
                 if (k == 0) {
 holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
                } else {
                holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
                }
                }
                  pw.println(holder);

                  holder = "";
           for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
                 if (k == 0) {
                  holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
                } else {
                holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
                }
                }
                  pw.println(holder);

          pw.println(listOfStudents.get(y).getCurrentQuarter());

          holder = "";
                  for (int k = 0; k < listOfStudents.get(y).getNumOfDourSaparasDoneMonth().length; k++) {
                 if (k == 0) {
                  holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
                } else {
                holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
                }
                }
                  pw.println(holder);
           pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStudents.get(y).get-
DourNextFill());

           DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
              LocalDateTime firstNow = LocalDateTime.now();
             String alreadyDone = firstFormatter.format(firstNow);

           if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
               pw.println(false);
               pw.println(-1);
               pw.println(false);
               pw.println(-1);
               pw.println(false);
               pw.println(-1);
               pw.println(-1);
               pw.println(false);
               pw.println(-1);
           } else {
                pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
                pw.println(listOfStudents.get(y).getTodayQuartersDone());
                pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
                pw.println(listOfStudents.get(y).getTodayDourSaparaDone());

             if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
                    pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
                    pw.println(listOfStudents.get(y).getTodayLinesMemorized());
                    pw.println(listOfStudents.get(y).getTodayMistakesMade());
                    pw.println(listOfStudents.get(y).isTodaySaparaFinished());
                    pw.println(listOfStudents.get(y).getTodaySaparaDone());
             } else {
               pw.println(false);
               pw.println(-1);
               pw.println(-1);
               pw.println(false);
               pw.println(-1);
             }
           }



           if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
                  holder = "";
             for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
                 if (k == 0) {
                  holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
                } else {
                holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
                }
                }
             pw.println(holder);

             holder = "";
           for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
                 if (k == 0) {
                  holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
                } else {
                holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
                }
                }
           pw.println(holder);
```

```
                holder = "";
                for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
                      if (k == 0) {
                       holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
                      } else {
                      holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
                      }
                      }
                pw.println(holder);

                holder = "";
                  for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
                      if (k == 0) {
                       holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
                      } else {
                      holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
                      }
                      }
                  pw.println(holder);

                  holder = "";
                  for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++) {
                      if (k == 0) {
                       holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
                      } else {
                      holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
                      }
                      }
                  pw.println(holder);
                pw.println(listOfStudents.get(y).getTotalSaparasDone());
                    pw.println(listOfStudents.get(y).getSaparasDone());
            pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStu-
dents.get(y).getSaparaNextFill());

                } else {
                  pw.println(false);
                  pw.println(0);
                  pw.println(0);
                  pw.println(false);
                  pw.println(0);
                  pw.println(0);
                  pw.println(0);
                  pw.println(0);
                  pw.println(0);
                }


                //attendance
               //printing to file for attendance
                holder = "";
                      for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

                      if (k == 0) {
                       holder = "" + (listOfStudents.get(y).getAttendance().get(k));

                     } else {
                     holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));

                     }
                        }
                      pw.println(holder);

                      holder = "";
                     //printing to file for reason absent
                     for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

                      if (d == 0) {
                       holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

                     } else {
                     holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

                     }
                       }
                     pw.println(holder);
                     //printing to file for covid screening
                     holder = "";
                     for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

                      if (r == 0) {
                       holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

                     } else {
                     holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

                     }
                    }
                      pw.println(holder);

                    //printing to file for reason covid screening was not done
                      holder = "";
                      for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++) {

                       if (p == 0) {
                       holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

                     } else {
                     holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreening().get(p));

                     }
                    }
```

```
                    pw.println(holder);

                   //printing to file for dates
                    holder = "";
                   for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

                      if (z == 0) {
                       holder = ""+(listOfStudents.get(y).getDate().get(z));

                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getDate().get(z));

                    }
                   }
                   pw.println(holder);
                   pw.println(listOfStudents.get(y).getGuardianOneFirstName());
                   pw.println(listOfStudents.get(y).getGuardianOneLastName());
                   pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
                   pw.println(listOfStudents.get(y).getGuardianOneEmail());
                   pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
                   pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
                   pw.println(listOfStudents.get(y).getGuardianTwoLastName());
                   pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
                   pw.println(listOfStudents.get(y).getGuardianTwoEmail());
                   pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

                   pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
                   pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
                   pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
                   pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
                   pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
                   pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
                   pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
                   pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
                   pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
                   pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

                   pw.println(listOfStudents.get(y).getHealthFactorOne());
                   pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
                   pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
                   pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
                   pw.println(listOfStudents.get(y).getHealthFactorTwo());
                   pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
                   pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
                   pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
                   pw.println(listOfStudents.get(y).getHealthFactorThree());
                   pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
                   pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
                   pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                        }
               pw.close();


                        }


          public static ArrayList <Student> fileOneOpen() {
                 try {
                  fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
                } catch (FileNotFoundException e) {
                  System.err.println("File not found! Choosing to quit now...");

                  System.exit(0);
                }

                //programChosen  - CHECK CONSTRUCTORS
                //add health factors to printing out in emergency situation stuff

               String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tem-
pAttendance, tempReasonAttendance, tempCovid, tempReasonCovid;

                Boolean[] dourDoneOrNot;
                Boolean todayDourDoneOrNot;
                int[] quarterNumDoneMonth;
                int todayQuartersDone, currentQuarter;
                Boolean[] numOfDourSaparasDoneMonth;
                Boolean todayDourSaparaDoneOrNot;
                int todayDourSaparaDone;
                int dourCurrentSapara, dourNextFill;

                String programChosen;
                String lastRecord;

                Boolean[] sabaqDoneOrNot;
                Boolean todaySabaqDoneOrNot;
                int[] linesMemorized;
                int todayLinesMemorized;
                int[] mistakesMade;
                int todayMistakesMade;
                Boolean[] numOfSaparasDoneMonth;
                Boolean todaySaparaFinished;
                int[] nameOfSaparasDoneMonth;
                int totalSaparasDone;
                int todaySaparaDone;
                String saparasDone;
                int currentSaparaMemorizing;
                int saparaNextFill = 0;

               int age;
```

```
                                String tempDate;
                                ArrayList<String> dates;


                                String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
                                String guardianOneEmail;
                                Boolean guardianOneCallAtWork;
                                String guardianTwoFirstName, guardianTwoLastName;
                                String guardianTwoPhoneNumber;
                                String guardianTwoEmail;
                                Boolean guardianTwoCallAtWork;

                                String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
                                String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
                                String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
                                String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

                                String healthFactorOne;
                                Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedica-
tionsRequired;
                                String healthFactorTwo;
                                Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedica-
tionsRequired;
                                String healthFactorThree;
                                Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedica-
tionsRequired;


                                Attendance attendanceOfStudent;
                                StudentProgress progressOfStudent;

                                while (fileScanner.hasNextLine()) {
                                       dourDoneOrNot = new Boolean[30];
                                            quarterNumDoneMonth = new int[30];
                                           numOfDourSaparasDoneMonth = new Boolean[30];
                                           sabaqDoneOrNot = new Boolean[30];
                                            linesMemorized = new int[30];
                                            mistakesMade = new int[30];
                                            numOfSaparasDoneMonth = new Boolean[30];
                                           nameOfSaparasDoneMonth = new int[30];

                                  firstName = (fileScanner.nextLine()).toLowerCase();
                                  middleName = (fileScanner.nextLine()).toLowerCase();
                                  lastName = (fileScanner.nextLine()).toLowerCase();
                                  address = (fileScanner.nextLine()).toLowerCase();
                                  dateOfBirth = fileScanner.nextLine();
                                  age = Integer.parseInt(fileScanner.nextLine());
                                  postalCode = (fileScanner.nextLine()).toLowerCase();
                                  language = (fileScanner.nextLine()).toLowerCase();
                                  countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                                  //progress of student
                                  programChosen = (fileScanner.nextLine()).toLowerCase();
                                  progressOfStudent = new StudentProgress();
                                  progressOfStudent.setProgramChosen(programChosen);

                                  lastRecord = (fileScanner.nextLine());
                                  progressOfStudent.setLastRecord(lastRecord);


                                  String tempDourDoneOrNot = fileScanner.nextLine();
                                  String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                                  for (int i = 0; i < strDourDoneOrNot.length; i++) {
                                    dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                                  }
                              progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                                  String tempQuarterNumDoneMonth = fileScanner.nextLine();
                                  String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                                  for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                                    quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                                  }
                              progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                                  currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuar-
ter(currentQuarter);

                                  String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                                  String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                                  for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                                    numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                                  }
                              progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                                  dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCur-
rentSapara(dourCurrentSapara);

                                  dourNextFill = Integer.parseInt(fileScanner.nextLine());
                              progressOfStudent.setOpenDourNextFill(dourNextFill);

                                  DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                                     LocalDateTime firstNow = LocalDateTime.now();
                                    String alreadyDone = firstFormatter.format(firstNow);

                                  if (!(alreadyDone.equals(lastRecord))) {
                                    if (programChosen.equals("hafiz")) {
                                      Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                      int holder = Integer.parseInt(fileScanner.nextLine());
                                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                      holder = Integer.parseInt(fileScanner.nextLine());
                                       temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                      holder = Integer.parseInt(fileScanner.nextLine());
```

```
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());

                todayDourDoneOrNot = false;
                todayDourSaparaDoneOrNot = false;
                todayQuartersDone = 0;
                todayDourSaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

              } else {
                Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                int holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                todayDourDoneOrNot = false;
                 todayQuartersDone = 0;
                 todayDourSaparaDoneOrNot = false;
                todayDourSaparaDone = 0;
                 todaySabaqDoneOrNot = false;
                todayLinesMemorized = 0;
                todayMistakesMade = 0;
                todaySaparaFinished = false;
                todaySaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
            progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
            progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
            }


            } else {
              todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
              progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

              todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
              progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

              todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
             progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

             todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
             progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

              if (!(programChosen.equals("hafiz"))) {

            todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.set-
OpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);

                todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
            progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
              } else {
                Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                int holder = Integer.parseInt(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
              }
            }



            if (!(programChosen.equals("hafiz"))) {
              String tempSabaqDoneOrNot = fileScanner.nextLine();
            String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
            for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
              sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
            }
            progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

            String tempLinesMemorized = fileScanner.nextLine();
            String strLinesMemorized[] = tempLinesMemorized.split(",");
            for (int i = 0; i < strLinesMemorized.length; i++) {
              linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
            }
            progressOfStudent.setOpenLinesMemorized(linesMemorized);

            String tempMistakesMade = fileScanner.nextLine();
            String strMistakesMade[] = tempMistakesMade.split(",");
```

```
      for (int i = 0; i < strMistakesMade.length; i++) {
        mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
      }
      progressOfStudent.setOpenMistakesMade(mistakesMade);

      String tempNumOfSaparasFinished = fileScanner.nextLine();
      String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
      for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
       numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

      String tempNameOfSaparasFinished = fileScanner.nextLine();
      String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
      for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
       nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

      totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

      saparasDone = fileScanner.nextLine();
      progressOfStudent.setOpenSaparasDone(saparasDone);

      currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

      saparaNextFill = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
      } else {
        String hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
      }


      //attendance

      tempAttendance = fileScanner.nextLine();
      String attendance[] = tempAttendance.split(",");
      attendanceOfStudent = new Attendance();
      for (int i = 0; i < attendance.length; i++ ) {
        attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
        }
      tempReasonAttendance = fileScanner.nextLine();
      String tempReason[] = tempReasonAttendance.split(",");
      for (int i = 0; i < tempReason.length; i++ ) {
        attendanceOfStudent.addReasonAbsent(tempReason[i]);
        }
      tempCovid = fileScanner.nextLine();
      String covid[] = tempCovid.split(",");
      for (int i = 0; i < covid.length; i++ ) {
        attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
        }
      tempReasonCovid = fileScanner.nextLine();
      String reasonCov[] = tempReasonCovid.split(",");
      for (int i = 0; i < reasonCov.length; i++ ) {
        attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
        }

        dates = new ArrayList<String>();

        tempDate = fileScanner.nextLine();
      String date[] = tempDate.split(",");
      for (int i = 0; i < date.length; i++ ) {
        dates.add(date[i]);
        }

        guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
        guardianOnePhoneNumber = fileScanner.nextLine();
        guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
        guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
        guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoPhoneNumber = fileScanner.nextLine();
        guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
        guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

        emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneHomeNumber = (fileScanner.nextLine());
        emergencyContactOneCellNumber = (fileScanner.nextLine());
        emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoHomeNumber = (fileScanner.nextLine());
        emergencyContactTwoCellNumber = (fileScanner.nextLine());

        healthFactorOne = (fileScanner.nextLine()).toLowerCase();
        healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
```

```
                                        healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                                        healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                                        healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                                        healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                                        healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                                        healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                                        healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                                        healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

                                    Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode,
language,countryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePho-
neNumber,guardianOneEmail, guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardi-
anTwoCallAtWork, emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber,
emergencyContactOneCellNumber,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactT-
woHomeNumber, emergencyContactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorO-
neMedicationsRequired, healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired,
healthFactorThree, healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                                        listOfStudents.add(tempS);
                                    }
                                    fileScanner.close();

                                    return listOfStudents;
                                }

            private static VerticalLayout createDialogLayout(Dialog dialog) {
                  H2 headline = new H2("Unsaved Changes");
                  headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                          .set("font-size", "1.5em").set("font-weight", "bold");

                  H5 message = new H5("There are unsaved changes. Do you want to continue editing or dicard them?");

                  Button cancelButton = new Button("Discard", e -> {
                     for (int i = 0; i < listOfStudents.size(); i++) {
                              listOfStudents.get(i).setTempReason("");
                              listOfStudents.get(i).setTempReason2("");
                     }
                     UI.getCurrent().navigate("menu");
                     dialog.close();
                     });
                  Button saveButton = new Button("Continue", e -> {
                     dialog.close();
                     });
                  saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                  HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                          saveButton);
                  buttonLayout
                          .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                  VerticalLayout dialogLayout = new VerticalLayout(headline, message,
                          buttonLayout);
                  dialogLayout.setPadding(false);
                  dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                  dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                  return dialogLayout;
              }

            private static void addCloseHandler(Component textField,
                      Editor<Student> editor) {
                  textField.getElement().addEventListener("keydown", e -> editor.cancel())
                          .setFilter("event.code === 'Escape'");
              }

}
```

CLASS: MenuAStudentP.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.datepicker.DatePicker;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.html.Footer;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H4;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.select.Select;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.router.Route;

@Route(value = "menuAStudentP", layout = Welcome.class)
public class MenuAStudentP extends VerticalLayout {

           static Scanner fileScanner;
           static ArrayList <Student> listOfStudents = new ArrayList <Student>();
           static int index;
           static String date;

           public MenuAStudentP() {
```

```
             //read from files
            listOfStudents.removeAll(listOfStudents);
            listOfStudents = fileOneOpen();
            info();

            H2 intro = new H2("Attendance for " + listOfStudents.get(index).getFullName());
intro.setMinWidth("700px");
intro.setSizeFull();
intro.getStyle().set("text-align", "center");

H4 date1 = new H4("Date: " + date);
            date1.setMinWidth("700px");
date1.setSizeFull();
date1.getStyle().set("text-align", "center");



      //find index of date in the date arraylist of student
      int dateIndex = -1;
      for (int i = 0; i < listOfStudents.get(index).getDate().size(); i++) {
            if (listOfStudents.get(index).getDate().get(i).equals(date)) {
                  dateIndex = i;
            }
      }

      Select<String> select = new Select<>();
      select.setLabel("Attendance");
      select.setItems("Present", "Absent");
      if ((listOfStudents.get(index).getAttendance().get(dateIndex)) == true) {
            select.setValue("Present");
      } else {
            select.setValue("Absent");
      }

      TextField textField = new TextField("Reason Absent");
             textField.setPlaceholder(listOfStudents.get(index).getReasonAbsent().get(dateIndex));

             Select<String> select2 = new Select<>();
                select2.setLabel("COVID Screening");
                select2.setItems("Complete", "Incomplete");
                if ((listOfStudents.get(index).getCovidScreening().get(dateIndex)) == true) {
                      select2.setValue("Complete");
                } else {
                      select2.setValue("Incomplete");
                }

             TextField textField2 = new TextField("Reason Incomplete");
             textField2.setPlaceholder(listOfStudents.get(index).getReasonCovidScreening().get(dateIndex));

            HorizontalLayout edit = new HorizontalLayout(select, textField, select2, textField2);
          edit.setSizeFull();
          edit.getStyle().set("text-align", "center");
          edit.setJustifyContentMode(JustifyContentMode.CENTER);
          edit.setHeight("200px");


       // Footer
                Button done = new Button("Done", e -> {
                       int ind = -1;

                      for (int k = 0; k < listOfStudents.get(index).getDate().size(); k++) {
                     if ((listOfStudents.get(index).getDate().get(k)).equals(date)) {
                     ind = k;
                     boolean set;
                     if (select.getValue() == "Present") {
                        set = true;
                     } else {
                        set = false;
                     }
                     listOfStudents.get(index).getAttendance().set(ind, set);
                     listOfStudents.get(index).getReasonAbsent().set(ind, textField.getValue());
                     boolean set2;
                     if (select2.getValue() == "Present") {
                        set2 = true;
                     } else {
                        set2 = false;
                     }
                     listOfStudents.get(index).getCovidScreening().set(ind, set2);
                     listOfStudents.get(index).getReasonCovidScreening().set(ind, textField2.getValue());
                   }
                  }
                      closeFileOne(listOfStudents);
                      listOfStudents.removeAll(listOfStudents);
                         UI.getCurrent().navigate("menu");
                });
                done.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                done.getStyle().set("margin-right", "var(--lumo-space-s)");

                Button anotherStudent = new Button("Another Student", l ->{
                      closeFileOne(listOfStudents);
                      Dialog dialog = new Dialog();
                      dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                      VerticalLayout dialogLayout = createDialogLayout(dialog);
                      dialog.add(dialogLayout);
                      dialog.open();
                      add(dialog);
                });
                anotherStudent.addThemeVariants(ButtonVariant.LUMO_TERTIARY);
```

```
                Footer footer = new Footer(done, anotherStudent);
                footer.getStyle().set("padding", "var(--lumo-space-wide-m)");
            footer.setSizeFull();
            setJustifyContentMode(JustifyContentMode.CENTER);
            setDefaultHorizontalComponentAlignment(Alignment.CENTER);
            getStyle().set("text-align", "center");

                add(intro, date1, edit, footer);

                setAlignItems(Alignment.STRETCH);
                   setPadding(false);
                   setSpacing(false);
                   getStyle().set("border", "1px solid var(--lumo-contrast-20pct)");


        }

        //store student index to temp file
             public static void store(int index, String date) {
                        PrintWriter pw = null;
         try {
            pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
            pw.println(index);
            pw.println(date);
            pw.close();
         } catch (FileNotFoundException e) {
           System.err.print("couldn't open file for writing!");
           System.exit(0);
         }
             }

        //access stored index of student in temp file
              public static void info () {
                        index = -1;
                         try {
                             fileScanner = new Scanner(new File("temp.txt"));
                             index = Integer.parseInt(fileScanner.nextLine());
                             date = fileScanner.nextLine();

                                    fileScanner.close();
                               } catch (FileNotFoundException e) {
                                 System.err.println("File not found! Choosing to quit now...");

                                 System.exit(0);
                               }
            }

            //dialog for entering another student's information
                                  private static VerticalLayout createDialogLayout(Dialog dialog) {
                               H2 headline = new H2("Enter Student Information");
                               headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                                       .set("font-size", "1.5em").set("font-weight", "bold");

                               TextField firstNameField = new TextField("First Name");
                               TextField lastNameField = new TextField("Last Name");
                               DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                               DatePicker singleFormatI18n = new DatePicker("Pick a Date");
                               VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                                       lastNameField, singleFormatI18n);
                               fieldLayout.setSpacing(false);
                               fieldLayout.setPadding(false);
                               fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                               Button cancelButton = new Button("Cancel", e -> dialog.close());
                               Button saveButton = new Button("Done", e -> {
                                  int index = -2;
                                   boolean found = false;
                                   boolean found2 = false;
                                  for (int i = 0; i < listOfStudents.size(); i++) {
                                       if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.get-
Value().equals(listOfStudents.get(i).getLastName())) {
                                                found = true;
                                               for (int k = 0; k < listOfStudents.get(i).getDate().size(); k++) {
                                                 if (listOfStudents.get(i).getDate().get(k).equals(firstFormatter1.format(single-
FormatI18n.getValue()))) {
                                                           found2 = true;
                                                           index = i;
                                                     store(index, firstFormatter1.format(singleFormatI18n.getValue()));
                                                                dialog.close();
                                                     UI.getCurrent().navigate("menuAStudentP");
                                                     UI.getCurrent().getPage().reload();
                                                     break;
                                                 }
                                             }
                                                if (found2 == true) {
                                                          break;
                                                }
                                            }
                                          if (!((found == true) && (found2 == true))) {
                                                     if (found == true) {
                                                               Notification.show("Invalid date entered.",
                                                                  3000, Notification.Position.MIDDLE);
                                                     } else {
                                                      Notification.show("Invalid name entered.",
                                                        3000, Notification.Position.MIDDLE);
                                                     }
                                                     }


                                       });
```

```
        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
        buttonLayout
                .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

        VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                buttonLayout);
        dialogLayout.setPadding(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

        return dialogLayout;
    }


        public static void closeFileOne(ArrayList <Student> listOfStudents) {
                    PrintWriter pw = null;
                          try {
                             pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                          } catch (FileNotFoundException e) {
                            System.err.print("couldn't open file for writing!");
                            System.exit(0);
                          }



                          for (int y = 0; y < listOfStudents.size(); y++) {

                            if (y == 0) {
                           pw.println(listOfStudents.get(y).getFirstName());
                            } else {
                              pw.println(listOfStudents.get(y).getFirstName());
                            }
                            pw.println(listOfStudents.get(y).getMiddleName());
                           pw.println(listOfStudents.get(y).getLastName());
                           pw.println(listOfStudents.get(y).getAddress());
                           pw.println(listOfStudents.get(y).getDateOfBirth());
                           pw.println(listOfStudents.get(y).getAge());
                           pw.println(listOfStudents.get(y).getPostalCode());
                           pw.println(listOfStudents.get(y).getLanguage());
                           pw.println(listOfStudents.get(y).getCountryOfBirth());
                      pw.println(listOfStudents.get(y).getProgramChosen());
                            pw.println(listOfStudents.get(y).getLastRecord());


                         String holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
                            if (k == 0) {
                     holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
                            } else {
                            holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
                            }
                            }
                              pw.println(holder);

                              holder = "";
                       for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
                            if (k == 0) {
                              holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
                            } else {
                            holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
                            }
                            }
                              pw.println(holder);

                      pw.println(listOfStudents.get(y).getCurrentQuarter());

                      holder = "";
                               for (int k = 0; k < listOfStudents.get(y).getNumOfDourSapa-
rasDoneMonth().length; k++) {

                               if (k == 0) {
                                holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
                            } else {
                            holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
                            }
                            }
                              pw.println(holder);
                        pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStu-
dents.get(y).getDourNextFill());

                        DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                           LocalDateTime firstNow = LocalDateTime.now();
                          String alreadyDone = firstFormatter.format(firstNow);

                        if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
                            pw.println(false);
                            pw.println(-1);
                            pw.println(false);
                            pw.println(-1);
                            pw.println(false);
                            pw.println(-1);
                            pw.println(-1);
                            pw.println(false);
                            pw.println(-1);
                        } else {
                             pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
                             pw.println(listOfStudents.get(y).getTodayQuartersDone());
                             pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
                             pw.println(listOfStudents.get(y).getTodayDourSaparaDone());
```

```
      if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
             pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
             pw.println(listOfStudents.get(y).getTodayLinesMemorized());
             pw.println(listOfStudents.get(y).getTodayMistakesMade());
             pw.println(listOfStudents.get(y).isTodaySaparaFinished());
             pw.println(listOfStudents.get(y).getTodaySaparaDone());
      } else {
        pw.println(false);
        pw.println(-1);
        pw.println(-1);
        pw.println(false);
        pw.println(-1);
      }
    }


    if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
          holder = "";
      for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
          if (k == 0) {
           holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
          } else {
          holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
          }
          }
      pw.println(holder);

      holder = "";
    for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
          if (k == 0) {
           holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
          } else {
          holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
          }
          }
    pw.println(holder);

    holder = "";
    for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
          if (k == 0) {
           holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
          } else {
          holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
          }
          }
    pw.println(holder);

    holder = "";
      for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
          if (k == 0) {
           holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
          } else {
          holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
          }
          }
      pw.println(holder);

      holder = "";
      for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++)
{

          if (k == 0) {
           holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
          } else {
          holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
          }
          }
      pw.println(holder);
    pw.println(listOfStudents.get(y).getTotalSaparasDone());
        pw.println(listOfStudents.get(y).getSaparasDone());
pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStu-
dents.get(y).getSaparaNextFill());

    } else {
      pw.println(false);
      pw.println(0);
      pw.println(0);
      pw.println(false);
      pw.println(0);
      pw.println(0);
      pw.println(0);
      pw.println(0);
      pw.println(0);
    }


    //attendance
   //printing to file for attendance
    holder = "";
          for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

          if (k == 0) {
           holder = "" + (listOfStudents.get(y).getAttendance().get(k));

          } else {
          holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));

          }
             }
           pw.println(holder);

           holder = "";
```

```
                //printing to file for reason absent
                for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

                 if (d == 0) {
                  holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

                } else {
                holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

                }
                  }
                pw.println(holder);
                //printing to file for covid screening
                holder = "";
                for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

                 if (r == 0) {
                  holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

                } else {
                holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

                }
              }
                pw.println(holder);

              //printing to file for reason covid screening was not done
                holder = "";
                for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++)
{


                 if (p == 0) {
                  holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

                } else {
                holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreen-
ing().get(p));


                }
              }
                pw.println(holder);

              //printing to file for dates
                holder = "";
              for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

                 if (z == 0) {
                  holder = ""+(listOfStudents.get(y).getDate().get(z));

                } else {
                holder = holder + ("," + listOfStudents.get(y).getDate().get(z));

                }
              }
              pw.println(holder);
              pw.println(listOfStudents.get(y).getGuardianOneFirstName());
              pw.println(listOfStudents.get(y).getGuardianOneLastName());
              pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
              pw.println(listOfStudents.get(y).getGuardianOneEmail());
              pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
              pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
              pw.println(listOfStudents.get(y).getGuardianTwoLastName());
              pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
              pw.println(listOfStudents.get(y).getGuardianTwoEmail());
              pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

              pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
              pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
              pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
              pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
              pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

              pw.println(listOfStudents.get(y).getHealthFactorOne());
              pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
              pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
              pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
              pw.println(listOfStudents.get(y).getHealthFactorTwo());
              pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
              pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
              pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
              pw.println(listOfStudents.get(y).getHealthFactorThree());
              pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
              pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
              pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                    }
            pw.close();


                    }

    public static ArrayList <Student> fileOneOpen() {
           try {
            fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
          } catch (FileNotFoundException e) {
```

```
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
            }

            //programChosen  - CHECK CONSTRUCTORS
            //add health factors to printing out in emergency situation stuff

            String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, coun-
tryOfBirth, tempAttendance, tempReasonAttendance, tempCovid, tempReasonCovid;

             Boolean[] dourDoneOrNot;
             Boolean todayDourDoneOrNot;
             int[] quarterNumDoneMonth;
             int todayQuartersDone, currentQuarter;
             Boolean[] numOfDourSaparasDoneMonth;
             Boolean todayDourSaparaDoneOrNot;
             int todayDourSaparaDone;
             int dourCurrentSapara, dourNextFill;

             String programChosen;
             String lastRecord;

             Boolean[] sabaqDoneOrNot;
             Boolean todaySabaqDoneOrNot;
             int[] linesMemorized;
             int todayLinesMemorized;
             int[] mistakesMade;
             int todayMistakesMade;
             Boolean[] numOfSaparasDoneMonth;
             Boolean todaySaparaFinished;
             int[] nameOfSaparasDoneMonth;
             int totalSaparasDone;
             int todaySaparaDone;
             String saparasDone;
             int currentSaparaMemorizing;
             int saparaNextFill = 0;

            int age;

            String tempDate;
            ArrayList<String> dates;


            String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
            String guardianOneEmail;
            Boolean guardianOneCallAtWork;
            String guardianTwoFirstName, guardianTwoLastName;
            String guardianTwoPhoneNumber;
            String guardianTwoEmail;
            Boolean guardianTwoCallAtWork;

            String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelation-
ship;
            String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
            String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelation-
ship;
            String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

            String healthFactorOne;
            Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedi-
cationsRequired;
            String healthFactorTwo;
            Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMed-
icationsRequired;
            String healthFactorThree;
            Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactor-
ThreeMedicationsRequired;

            Attendance attendanceOfStudent;
            StudentProgress progressOfStudent;

            while (fileScanner.hasNextLine()) {
                   dourDoneOrNot = new Boolean[30];
                        quarterNumDoneMonth = new int[30];
                       numOfDourSaparasDoneMonth = new Boolean[30];
                       sabaqDoneOrNot = new Boolean[30];
                        linesMemorized = new int[30];
                        mistakesMade = new int[30];
                        numOfSaparasDoneMonth = new Boolean[30];
                       nameOfSaparasDoneMonth = new int[30];

              firstName = (fileScanner.nextLine()).toLowerCase();
              middleName = (fileScanner.nextLine()).toLowerCase();
              lastName = (fileScanner.nextLine()).toLowerCase();
              address = (fileScanner.nextLine()).toLowerCase();
              dateOfBirth = fileScanner.nextLine();
              age = Integer.parseInt(fileScanner.nextLine());
              postalCode = (fileScanner.nextLine()).toLowerCase();
              language = (fileScanner.nextLine()).toLowerCase();
              countryOfBirth = (fileScanner.nextLine()).toLowerCase();

              //progress of student
              programChosen = (fileScanner.nextLine()).toLowerCase();
              progressOfStudent = new StudentProgress();
              progressOfStudent.setProgramChosen(programChosen);

              lastRecord = (fileScanner.nextLine());
              progressOfStudent.setLastRecord(lastRecord);
```

```
                String tempDourDoneOrNot = fileScanner.nextLine();
                String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                for (int i = 0; i < strDourDoneOrNot.length; i++) {
                  dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                }
            progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                String tempQuarterNumDoneMonth = fileScanner.nextLine();
                String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                  quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                }
            progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCur-
rentQuarter(currentQuarter);

                String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                  numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                }
            progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.set-
OpenDourCurrentSapara(dourCurrentSapara);

                dourNextFill = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenDourNextFill(dourNextFill);

                DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                   LocalDateTime firstNow = LocalDateTime.now();
                  String alreadyDone = firstFormatter.format(firstNow);

                if (!(alreadyDone.equals(lastRecord))) {
                  if (programChosen.equals("hafiz")) {
                    Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    int holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                     temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());

                    todayDourDoneOrNot = false;
                    todayDourSaparaDoneOrNot = false;
                    todayQuartersDone = 0;
                    todayDourSaparaDone = 0;
                    progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                  } else {
                    Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    int holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                     temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    todayDourDoneOrNot = false;
                     todayQuartersDone = 0;
                     todayDourSaparaDoneOrNot = false;
                    todayDourSaparaDone = 0;
                     todaySabaqDoneOrNot = false;
                    todayLinesMemorized = 0;
                    todayMistakesMade = 0;
                    todaySaparaFinished = false;
                    todaySaparaDone = 0;
                    progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
            progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
            progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                }


                } else {
                  todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                  progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                  todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                  progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                   todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                 progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

                 todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                 progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                  if (!(programChosen.equals("hafiz"))) {
```

```
            todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.set-
OpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);

            todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

            todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

            todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

            todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
          } else {
            Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
            int holder = Integer.parseInt(fileScanner.nextLine());
            holder = Integer.parseInt(fileScanner.nextLine());
            temporary = Boolean.parseBoolean(fileScanner.nextLine());
            holder = Integer.parseInt(fileScanner.nextLine());
          }
        }



        if (!(programChosen.equals("hafiz"))) {
          String tempSabaqDoneOrNot = fileScanner.nextLine();
        String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
        for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
          sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
        }
progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

        String tempLinesMemorized = fileScanner.nextLine();
        String strLinesMemorized[] = tempLinesMemorized.split(",");
        for (int i = 0; i < strLinesMemorized.length; i++) {
          linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
        }
        progressOfStudent.setOpenLinesMemorized(linesMemorized);

        String tempMistakesMade = fileScanner.nextLine();
        String strMistakesMade[] = tempMistakesMade.split(",");
        for (int i = 0; i < strMistakesMade.length; i++) {
          mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
        }
        progressOfStudent.setOpenMistakesMade(mistakesMade);

        String tempNumOfSaparasFinished = fileScanner.nextLine();
        String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
        for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
         numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
        }
        progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

        String tempNameOfSaparasFinished = fileScanner.nextLine();
        String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
        for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
         nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
        }
        progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

        totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
        progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

        saparasDone = fileScanner.nextLine();
        progressOfStudent.setOpenSaparasDone(saparasDone);

        currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

        saparaNextFill = Integer.parseInt(fileScanner.nextLine());
        progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
        } else {
          String hold = fileScanner.nextLine();
          hold = fileScanner.nextLine();
          hold = fileScanner.nextLine();
          hold = fileScanner.nextLine();
          hold = fileScanner.nextLine();
          hold = fileScanner.nextLine();
          hold = fileScanner.nextLine();
          hold = fileScanner.nextLine();
          hold = fileScanner.nextLine();
        }


        //attendance

        tempAttendance = fileScanner.nextLine();
        String attendance[] = tempAttendance.split(",");
        attendanceOfStudent = new Attendance();
        for (int i = 0; i < attendance.length; i++ ) {
          attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
          }
        tempReasonAttendance = fileScanner.nextLine();
        String tempReason[] = tempReasonAttendance.split(",");
        for (int i = 0; i < tempReason.length; i++ ) {
          attendanceOfStudent.addReasonAbsent(tempReason[i]);
          }
        tempCovid = fileScanner.nextLine();
        String covid[] = tempCovid.split(",");
        for (int i = 0; i < covid.length; i++ ) {
```

```
                attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
                }
              tempReasonCovid = fileScanner.nextLine();
              String reasonCov[] = tempReasonCovid.split(",");
              for (int i = 0; i < reasonCov.length; i++ ) {
                attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
                }

                dates = new ArrayList<String>();

                tempDate = fileScanner.nextLine();
              String date[] = tempDate.split(",");
              for (int i = 0; i < date.length; i++ ) {
                dates.add(date[i]);
                }

                guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
                guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
                guardianOnePhoneNumber = fileScanner.nextLine();
                guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
                guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
                guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
                guardianTwoPhoneNumber = fileScanner.nextLine();
                guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
                guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

                emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
                emergencyContactOneHomeNumber = (fileScanner.nextLine());
                emergencyContactOneCellNumber = (fileScanner.nextLine());
                emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoHomeNumber = (fileScanner.nextLine());
                emergencyContactTwoCellNumber = (fileScanner.nextLine());

                healthFactorOne = (fileScanner.nextLine()).toLowerCase();
                healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

              Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age,
postalCode, language,countryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePho-
neNumber,guardianOneEmail, guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardi-
anTwoCallAtWork, emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber,
emergencyContactOneCellNumber,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactT-
woHomeNumber, emergencyContactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorO-
neMedicationsRequired, healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired,
healthFactorThree, healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                listOfStudents.add(tempS);
              }
              fileScanner.close();

              return listOfStudents;
            }


}
```

CLASS: MenuAStudentT.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.datepicker.DatePicker;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.html.Footer;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H4;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.select.Select;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.router.Route;

@Route(value = "menuAStudentP", layout = Welcome.class)
public class MenuAStudentP extends VerticalLayout {

          static Scanner fileScanner;
```

```
   static ArrayList <Student> listOfStudents = new ArrayList <Student>();
   static int index;
   static String date;

   public MenuAStudentP() {

               //read from files
              listOfStudents.removeAll(listOfStudents);
              listOfStudents = fileOneOpen();
              info();

              H2 intro = new H2("Attendance for " + listOfStudents.get(index).getFullName());
intro.setMinWidth("700px");
intro.setSizeFull();
intro.getStyle().set("text-align", "center");

H4 date1 = new H4("Date: " + date);
              date1.setMinWidth("700px");
date1.setSizeFull();
date1.getStyle().set("text-align", "center");



       //find index of date in the date arraylist of student
       int dateIndex = -1;
       for (int i = 0; i < listOfStudents.get(index).getDate().size(); i++) {
              if (listOfStudents.get(index).getDate().get(i).equals(date)) {
                     dateIndex = i;
              }
       }

       Select<String> select = new Select<>();
       select.setLabel("Attendance");
       select.setItems("Present", "Absent");
       if ((listOfStudents.get(index).getAttendance().get(dateIndex)) == true) {
              select.setValue("Present");
       } else {
              select.setValue("Absent");
       }

       TextField textField = new TextField("Reason Absent");
              textField.setPlaceholder(listOfStudents.get(index).getReasonAbsent().get(dateIndex));

              Select<String> select2 = new Select<>();
                 select2.setLabel("COVID Screening");
                 select2.setItems("Complete", "Incomplete");
                 if ((listOfStudents.get(index).getCovidScreening().get(dateIndex)) == true) {
                        select2.setValue("Complete");
                 } else {
                        select2.setValue("Incomplete");
                 }

              TextField textField2 = new TextField("Reason Incomplete");
              textField2.setPlaceholder(listOfStudents.get(index).getReasonCovidScreening().get(dateIndex));

              HorizontalLayout edit = new HorizontalLayout(select, textField, select2, textField2);
          edit.setSizeFull();
          edit.getStyle().set("text-align", "center");
          edit.setJustifyContentMode(JustifyContentMode.CENTER);
          edit.setHeight("200px");


        // Footer
                  Button done = new Button("Done", e -> {
                         int ind = -1;

                        for (int k = 0; k < listOfStudents.get(index).getDate().size(); k++) {
                       if ((listOfStudents.get(index).getDate().get(k)).equals(date)) {
                       ind = k;
                       boolean set;
                       if (select.getValue() == "Present") {
                          set = true;
                       } else {
                          set = false;
                       }
                       listOfStudents.get(index).getAttendance().set(ind, set);
                       listOfStudents.get(index).getReasonAbsent().set(ind, textField.getValue());
                       boolean set2;
                       if (select2.getValue() == "Present") {
                          set2 = true;
                       } else {
                          set2 = false;
                       }
                       listOfStudents.get(index).getCovidScreening().set(ind, set2);
                       listOfStudents.get(index).getReasonCovidScreening().set(ind, textField2.getValue());
                     }
                   }
                        closeFileOne(listOfStudents);
                        listOfStudents.removeAll(listOfStudents);
                           UI.getCurrent().navigate("menu");
                  });
                  done.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                  done.getStyle().set("margin-right", "var(--lumo-space-s)");

                  Button anotherStudent = new Button("Another Student", l ->{
                         closeFileOne(listOfStudents);
                         Dialog dialog = new Dialog();
                         dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                         VerticalLayout dialogLayout = createDialogLayout(dialog);
                         dialog.add(dialogLayout);
```

```
                        dialog.open();
                        add(dialog);
                    });
                    anotherStudent.addThemeVariants(ButtonVariant.LUMO_TERTIARY);

                    Footer footer = new Footer(done, anotherStudent);
                    footer.getStyle().set("padding", "var(--lumo-space-wide-m)");
                footer.setSizeFull();
                setJustifyContentMode(JustifyContentMode.CENTER);
                setDefaultHorizontalComponentAlignment(Alignment.CENTER);
                getStyle().set("text-align", "center");

                    add(intro, date1, edit, footer);

                    setAlignItems(Alignment.STRETCH);
                       setPadding(false);
                       setSpacing(false);
                       getStyle().set("border", "1px solid var(--lumo-contrast-20pct)");


            }

            //store student index to temp file
                public static void store(int index, String date) {
                            PrintWriter pw = null;
                 try {
                    pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                    pw.println(index);
                    pw.println(date);
                    pw.close();
                 } catch (FileNotFoundException e) {
                   System.err.print("couldn't open file for writing!");
                   System.exit(0);
                 }
                     }

            //access stored index of student in temp file
                  public static void info () {
                            index = -1;
                             try {
                                 fileScanner = new Scanner(new File("temp.txt"));
                                 index = Integer.parseInt(fileScanner.nextLine());
                                 date = fileScanner.nextLine();

                                        fileScanner.close();
                                   } catch (FileNotFoundException e) {
                                     System.err.println("File not found! Choosing to quit now...");

                                     System.exit(0);
                                   }
                    }

                    //dialog for entering another student's information
                                       private static VerticalLayout createDialogLayout(Dialog dialog) {
                                   H2 headline = new H2("Enter Student Information");
                                   headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                                           .set("font-size", "1.5em").set("font-weight", "bold");

                                   TextField firstNameField = new TextField("First Name");
                                   TextField lastNameField = new TextField("Last Name");
                                   DateTimeFormatter firstFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                                   DatePicker singleFormatI18n = new DatePicker("Pick a Date");
                                   VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                                           lastNameField, singleFormatI18n);
                                   fieldLayout.setSpacing(false);
                                   fieldLayout.setPadding(false);
                                   fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                                   Button cancelButton = new Button("Cancel", e -> dialog.close());
                                   Button saveButton = new Button("Done", e -> {
                                      int index = -2;
                                       boolean found = false;
                                       boolean found2 = false;
                                      for (int i = 0; i < listOfStudents.size(); i++) {
                                           if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.get-
Value().equals(listOfStudents.get(i).getLastName())) {
                                                found = true;
                                               for (int k = 0; k < listOfStudents.get(i).getDate().size(); k++) {
                                                 if (listOfStudents.get(i).getDate().get(k).equals(firstFormatter1.format(single-
FormatI18n.getValue()))) {
                                                           found2 = true;
                                                           index = i;
                                                     store(index, firstFormatter1.format(singleFormatI18n.getValue()));
                                                                dialog.close();
                                                     UI.getCurrent().navigate("menuAStudentP");
                                                     UI.getCurrent().getPage().reload();
                                                     break;
                                                 }
                                              }
                                               if (found2 == true) {
                                                        break;
                                               }
                                           }
                                       if (!((found == true) && (found2 == true))) {
                                                 if (found == true) {
                                                           Notification.show("Invalid date entered.",
                                                              3000, Notification.Position.MIDDLE);
                                                 } else {
                                                  Notification.show("Invalid name entered.",
                                                    3000, Notification.Position.MIDDLE);
```

```
                        }
                        }

        });
        saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                saveButton);
        buttonLayout
                .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

        VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                buttonLayout);
        dialogLayout.setPadding(false);
        dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
        dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

        return dialogLayout;
    }


        public static void closeFileOne(ArrayList <Student> listOfStudents) {
                    PrintWriter pw = null;
                          try {
                             pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                          } catch (FileNotFoundException e) {
                            System.err.print("couldn't open file for writing!");
                            System.exit(0);
                          }



                          for (int y = 0; y < listOfStudents.size(); y++) {

                            if (y == 0) {
                           pw.println(listOfStudents.get(y).getFirstName());
                            } else {
                              pw.println(listOfStudents.get(y).getFirstName());
                            }
                            pw.println(listOfStudents.get(y).getMiddleName());
                           pw.println(listOfStudents.get(y).getLastName());
                           pw.println(listOfStudents.get(y).getAddress());
                           pw.println(listOfStudents.get(y).getDateOfBirth());
                           pw.println(listOfStudents.get(y).getAge());
                           pw.println(listOfStudents.get(y).getPostalCode());
                           pw.println(listOfStudents.get(y).getLanguage());
                           pw.println(listOfStudents.get(y).getCountryOfBirth());
                      pw.println(listOfStudents.get(y).getProgramChosen());
                            pw.println(listOfStudents.get(y).getLastRecord());


                         String holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
                            if (k == 0) {
                     holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
                          } else {
                          holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
                          }
                          }
                            pw.println(holder);

                            holder = "";
                    for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
                            if (k == 0) {
                              holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
                          } else {
                          holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
                          }
                          }
                            pw.println(holder);

                      pw.println(listOfStudents.get(y).getCurrentQuarter());

                      holder = "";
                               for (int k = 0; k < listOfStudents.get(y).getNumOfDourSapa-
rasDoneMonth().length; k++) {

                              if (k == 0) {
                              holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
                          } else {
                          holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
                          }
                          }
                            pw.println(holder);
                        pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStu-
dents.get(y).getDourNextFill());


                        DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                           LocalDateTime firstNow = LocalDateTime.now();
                          String alreadyDone = firstFormatter.format(firstNow);

                        if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
                            pw.println(false);
                            pw.println(-1);
                            pw.println(false);
                            pw.println(-1);
                            pw.println(false);
                            pw.println(-1);
                            pw.println(-1);
                            pw.println(false);
                            pw.println(-1);
                        } else {
```

```
            pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
            pw.println(listOfStudents.get(y).getTodayQuartersDone());
            pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
            pw.println(listOfStudents.get(y).getTodayDourSaparaDone());

          if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
                pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
                pw.println(listOfStudents.get(y).getTodayLinesMemorized());
                pw.println(listOfStudents.get(y).getTodayMistakesMade());
                pw.println(listOfStudents.get(y).isTodaySaparaFinished());
                pw.println(listOfStudents.get(y).getTodaySaparaDone());
          } else {
            pw.println(false);
            pw.println(-1);
            pw.println(-1);
            pw.println(false);
            pw.println(-1);
          }
        }



        if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
              holder = "";
          for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
              }
              }
          pw.println(holder);

          holder = "";
        for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
              }
              }
        pw.println(holder);

        holder = "";
        for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
              }
              }
        pw.println(holder);

        holder = "";
          for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
              }
              }
          pw.println(holder);

          holder = "";
          for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++)
{

              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
              }
              }
          pw.println(holder);
        pw.println(listOfStudents.get(y).getTotalSaparasDone());
            pw.println(listOfStudents.get(y).getSaparasDone());
pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStu-
dents.get(y).getSaparaNextFill());

        } else {
          pw.println(false);
          pw.println(0);
          pw.println(0);
          pw.println(false);
          pw.println(0);
          pw.println(0);
          pw.println(0);
          pw.println(0);
          pw.println(0);
        }


        //attendance
       //printing to file for attendance
        holder = "";
              for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getAttendance().get(k));

              } else {
              holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));
```

```
                }
                   }
                 pw.println(holder);

                 holder = "";
                //printing to file for reason absent
                for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

                 if (d == 0) {
                  holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

                } else {
                holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

                }
                  }
                pw.println(holder);
                //printing to file for covid screening
                holder = "";
                for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

                 if (r == 0) {
                  holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

                } else {
                holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

                }
              }
                pw.println(holder);

              //printing to file for reason covid screening was not done
                holder = "";
                for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++)
{


                 if (p == 0) {
                  holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

                } else {
                holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreen-
ing().get(p));


                }
              }
                pw.println(holder);

              //printing to file for dates
                holder = "";
              for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

                 if (z == 0) {
                  holder = ""+(listOfStudents.get(y).getDate().get(z));

                } else {
                holder = holder + ("," + listOfStudents.get(y).getDate().get(z));

                }
              }
              pw.println(holder);
              pw.println(listOfStudents.get(y).getGuardianOneFirstName());
              pw.println(listOfStudents.get(y).getGuardianOneLastName());
              pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
              pw.println(listOfStudents.get(y).getGuardianOneEmail());
              pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
              pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
              pw.println(listOfStudents.get(y).getGuardianTwoLastName());
              pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
              pw.println(listOfStudents.get(y).getGuardianTwoEmail());
              pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

              pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
              pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
              pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
              pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
              pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
              pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

              pw.println(listOfStudents.get(y).getHealthFactorOne());
              pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
              pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
              pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
              pw.println(listOfStudents.get(y).getHealthFactorTwo());
              pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
              pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
              pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
              pw.println(listOfStudents.get(y).getHealthFactorThree());
              pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
              pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
              pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                  }
        pw.close();


                  }
```

```
public static ArrayList <Student> fileOneOpen() {
       try {
        fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
       } catch (FileNotFoundException e) {
        System.err.println("File not found! Choosing to quit now...");

        System.exit(0);
       }

       //programChosen  - CHECK CONSTRUCTORS
       //add health factors to printing out in emergency situation stuff

       String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, coun-
tryOfBirth, tempAttendance, tempReasonAttendance, tempCovid, tempReasonCovid;

        Boolean[] dourDoneOrNot;
        Boolean todayDourDoneOrNot;
        int[] quarterNumDoneMonth;
        int todayQuartersDone, currentQuarter;
        Boolean[] numOfDourSaparasDoneMonth;
        Boolean todayDourSaparaDoneOrNot;
        int todayDourSaparaDone;
        int dourCurrentSapara, dourNextFill;

        String programChosen;
        String lastRecord;

        Boolean[] sabaqDoneOrNot;
        Boolean todaySabaqDoneOrNot;
        int[] linesMemorized;
        int todayLinesMemorized;
        int[] mistakesMade;
        int todayMistakesMade;
        Boolean[] numOfSaparasDoneMonth;
        Boolean todaySaparaFinished;
        int[] nameOfSaparasDoneMonth;
        int totalSaparasDone;
        int todaySaparaDone;
        String saparasDone;
        int currentSaparaMemorizing;
        int saparaNextFill = 0;

       int age;

       String tempDate;
       ArrayList<String> dates;


       String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
       String guardianOneEmail;
       Boolean guardianOneCallAtWork;
       String guardianTwoFirstName, guardianTwoLastName;
       String guardianTwoPhoneNumber;
       String guardianTwoEmail;
       Boolean guardianTwoCallAtWork;

       String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelation-
ship;
       String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
       String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelation-
ship;
       String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;


       String healthFactorOne;
       Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedi-
cationsRequired;
       String healthFactorTwo;
       Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMed-
icationsRequired;
       String healthFactorThree;
       Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactor-
ThreeMedicationsRequired;


       Attendance attendanceOfStudent;
       StudentProgress progressOfStudent;

       while (fileScanner.hasNextLine()) {
              dourDoneOrNot = new Boolean[30];
                   quarterNumDoneMonth = new int[30];
                  numOfDourSaparasDoneMonth = new Boolean[30];
                  sabaqDoneOrNot = new Boolean[30];
                   linesMemorized = new int[30];
                   mistakesMade = new int[30];
                   numOfSaparasDoneMonth = new Boolean[30];
                  nameOfSaparasDoneMonth = new int[30];

         firstName = (fileScanner.nextLine()).toLowerCase();
         middleName = (fileScanner.nextLine()).toLowerCase();
         lastName = (fileScanner.nextLine()).toLowerCase();
         address = (fileScanner.nextLine()).toLowerCase();
         dateOfBirth = fileScanner.nextLine();
         age = Integer.parseInt(fileScanner.nextLine());
         postalCode = (fileScanner.nextLine()).toLowerCase();
         language = (fileScanner.nextLine()).toLowerCase();
         countryOfBirth = (fileScanner.nextLine()).toLowerCase();

         //progress of student
         programChosen = (fileScanner.nextLine()).toLowerCase();
         progressOfStudent = new StudentProgress();
         progressOfStudent.setProgramChosen(programChosen);
```

```
                                        lastRecord = (fileScanner.nextLine());
                                        progressOfStudent.setLastRecord(lastRecord);


                                        String tempDourDoneOrNot = fileScanner.nextLine();
                                        String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                                        for (int i = 0; i < strDourDoneOrNot.length; i++) {
                                          dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                                        }
                                    progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                                        String tempQuarterNumDoneMonth = fileScanner.nextLine();
                                        String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                                        for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                                          quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                                        }
                                    progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                                        currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCur-
rentQuarter(currentQuarter);


                                        String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                                        String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                                        for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                                          numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                                        }
                                    progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                                        dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.set-
OpenDourCurrentSapara(dourCurrentSapara);


                                        dourNextFill = Integer.parseInt(fileScanner.nextLine());
                                    progressOfStudent.setOpenDourNextFill(dourNextFill);

                                        DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                                           LocalDateTime firstNow = LocalDateTime.now();
                                          String alreadyDone = firstFormatter.format(firstNow);

                                        if (!(alreadyDone.equals(lastRecord))) {
                                          if (programChosen.equals("hafiz")) {
                                            Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                            int holder = Integer.parseInt(fileScanner.nextLine());
                                            temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                            holder = Integer.parseInt(fileScanner.nextLine());
                                             temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                            holder = Integer.parseInt(fileScanner.nextLine());
                                            holder = Integer.parseInt(fileScanner.nextLine());
                                            temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                            holder = Integer.parseInt(fileScanner.nextLine());

                                            todayDourDoneOrNot = false;
                                            todayDourSaparaDoneOrNot = false;
                                            todayQuartersDone = 0;
                                            todayDourSaparaDone = 0;
                                            progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
                                    progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
                                    progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
                                    progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                                           } else {
                                            Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                            int holder = Integer.parseInt(fileScanner.nextLine());
                                            temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                            holder = Integer.parseInt(fileScanner.nextLine());
                                             temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                            holder = Integer.parseInt(fileScanner.nextLine());
                                            holder = Integer.parseInt(fileScanner.nextLine());
                                            temporary = Boolean.parseBoolean(fileScanner.nextLine());
                                            holder = Integer.parseInt(fileScanner.nextLine());
                                            todayDourDoneOrNot = false;
                                             todayQuartersDone = 0;
                                             todayDourSaparaDoneOrNot = false;
                                            todayDourSaparaDone = 0;
                                             todaySabaqDoneOrNot = false;
                                            todayLinesMemorized = 0;
                                            todayMistakesMade = 0;
                                            todaySaparaFinished = false;
                                            todaySaparaDone = 0;
                                            progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
                                    progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
                                    progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
                                    progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
                                    progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
                                    progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
                                    progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
                                    progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
                                    progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                                        }


                                        } else {
                                          todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                                          progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                                          todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                                          progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                                          todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                                         progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
```

```
                todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                 if (!(programChosen.equals("hafiz"))) {

               todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.set-
OpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);

                   todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                   todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                   todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                   todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                 } else {
                   Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   int holder = Integer.parseInt(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());
                   temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());
                 }
               }



               if (!(programChosen.equals("hafiz"))) {
                 String tempSabaqDoneOrNot = fileScanner.nextLine();
               String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
               for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
                 sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
               }
progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

               String tempLinesMemorized = fileScanner.nextLine();
               String strLinesMemorized[] = tempLinesMemorized.split(",");
               for (int i = 0; i < strLinesMemorized.length; i++) {
                 linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
               }
               progressOfStudent.setOpenLinesMemorized(linesMemorized);

               String tempMistakesMade = fileScanner.nextLine();
               String strMistakesMade[] = tempMistakesMade.split(",");
               for (int i = 0; i < strMistakesMade.length; i++) {
                 mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
               }
               progressOfStudent.setOpenMistakesMade(mistakesMade);

               String tempNumOfSaparasFinished = fileScanner.nextLine();
               String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
               for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
                numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
               }
               progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

               String tempNameOfSaparasFinished = fileScanner.nextLine();
               String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
               for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
                nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
               }
               progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

               totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
               progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

               saparasDone = fileScanner.nextLine();
               progressOfStudent.setOpenSaparasDone(saparasDone);

               currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

               saparaNextFill = Integer.parseInt(fileScanner.nextLine());
               progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
               } else {
                 String hold = fileScanner.nextLine();
                 hold = fileScanner.nextLine();
                 hold = fileScanner.nextLine();
                 hold = fileScanner.nextLine();
                 hold = fileScanner.nextLine();
                 hold = fileScanner.nextLine();
                 hold = fileScanner.nextLine();
                 hold = fileScanner.nextLine();
                 hold = fileScanner.nextLine();
               }


               //attendance

               tempAttendance = fileScanner.nextLine();
               String attendance[] = tempAttendance.split(",");
               attendanceOfStudent = new Attendance();
               for (int i = 0; i < attendance.length; i++ ) {
                 attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
                 }
               tempReasonAttendance = fileScanner.nextLine();
               String tempReason[] = tempReasonAttendance.split(",");
               for (int i = 0; i < tempReason.length; i++ ) {
```

```
            attendanceOfStudent.addReasonAbsent(tempReason[i]);
            }
          tempCovid = fileScanner.nextLine();
          String covid[] = tempCovid.split(",");
          for (int i = 0; i < covid.length; i++ ) {
            attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
            }
          tempReasonCovid = fileScanner.nextLine();
          String reasonCov[] = tempReasonCovid.split(",");
          for (int i = 0; i < reasonCov.length; i++ ) {
            attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
            }

            dates = new ArrayList<String>();

            tempDate = fileScanner.nextLine();
          String date[] = tempDate.split(",");
          for (int i = 0; i < date.length; i++ ) {
            dates.add(date[i]);
            }

            guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
            guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
            guardianOnePhoneNumber = fileScanner.nextLine();
            guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
            guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
            guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
            guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
            guardianTwoPhoneNumber = fileScanner.nextLine();
            guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
            guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

            emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
            emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
            emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
            emergencyContactOneHomeNumber = (fileScanner.nextLine());
            emergencyContactOneCellNumber = (fileScanner.nextLine());
            emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
            emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
            emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
            emergencyContactTwoHomeNumber = (fileScanner.nextLine());
            emergencyContactTwoCellNumber = (fileScanner.nextLine());

            healthFactorOne = (fileScanner.nextLine()).toLowerCase();
            healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
            healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
            healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
            healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
            healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
            healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
            healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
            healthFactorThree = (fileScanner.nextLine()).toLowerCase();
            healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
            healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
            healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

          Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age,
postalCode, language,countryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePho-
neNumber,guardianOneEmail, guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardi-
anTwoCallAtWork, emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber,
emergencyContactOneCellNumber,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactT-
woHomeNumber, emergencyContactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorO-
neMedicationsRequired, healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired,
healthFactorThree, healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
            listOfStudents.add(tempS);
        }
       fileScanner.close();

       return listOfStudents;
      }


}
```

CLASS: MenuBRecordsV.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.html.Footer;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H3;
import com.vaadin.flow.component.html.Header;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.component.html.Section;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
```

```
import com.vaadin.flow.component.orderedlayout.Scroller;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.router.Route;

@Route(value = "menuBRecordsV", layout = Welcome.class)
public class MenuBRecordsV extends VerticalLayout{

           public static ArrayList <Student> listOfStudents = new ArrayList <Student>();
           public static Scanner fileScanner;

           public static final String PERSONAL_TITLE_ID = "personal-title";
           public static final String EMPLOYMENT_TITLE_ID = "employment-title";

           public MenuBRecordsV() {
                      listOfStudents.removeAll(listOfStudents);
                      listOfStudents = fileOneOpen();
                      int index = index();

                      // Header
               Header header = new Header();
               header.getStyle()
                       .set("align-items", "center")
                       .set("border-bottom", "1px solid var(--lumo-contrast-20pct)")
                       .set("display", "flex")
                       .set("padding", "var(--lumo-space-m)");

               //capitalize first letter of student's name and store
           String studentName =  listOfStudents.get(index).getFullName();

               H2 editEmployee = new H2(studentName + "'s Information");
               editEmployee.getStyle().set("margin", "0");

               header.add(editEmployee);
               add(header);

               //Student Information
               H3 studentTitle = new H3("Student Information");
               Section studentInformation = new Section(studentTitle);

               Paragraph firstName = new Paragraph("First Name: " + listOfStudents.get(index).getFirstName());
               firstName.setWidthFull();
                      studentInformation.add(firstName);
                      Paragraph middleName = new Paragraph("Middle Name: " + listOfStudents.get(index).getMiddleName());
       middleName.setWidthFull();
       studentInformation.add(middleName);
       Paragraph lastName = new Paragraph("Last Name: " + listOfStudents.get(index).getLastName());
       lastName.setWidthFull();
       studentInformation.add(lastName);
       Paragraph address = new Paragraph("Address: " + listOfStudents.get(index).getAddress());
       address.setWidthFull();
       studentInformation.add(address);
       Paragraph dateOfBirth = new Paragraph ("Date Of Birth: " + listOfStudents.get(index).getDateOfBirth());
       dateOfBirth.setWidthFull();
       studentInformation.add(dateOfBirth);
       Paragraph age = new Paragraph ("Age: " + listOfStudents.get(index).getAge());
       age.setWidthFull();
       studentInformation.add(age);
       Paragraph pCode = new Paragraph ("Postal Code: " + listOfStudents.get(index).getPostalCode());
       pCode.setWidthFull();
       studentInformation.add(pCode);
       Paragraph language = new Paragraph ("Language: " + listOfStudents.get(index).getLanguage());
       language.setWidthFull();
       studentInformation.add(language);
       Paragraph country = new Paragraph ("Country Of Birth: " + listOfStudents.get(index).getCountryOfBirth());
       country.setWidthFull();
       studentInformation.add(country);

                      // Personal information
                      H3 personalTitle = new H3("Guardian Information");
                      Section personalInformation = new Section(personalTitle);

                      Paragraph guardianOneFirstName = new Paragraph ("Guardian One First Name: " + listOfStudents.get(index).getGuardianOneFirst-
Name());
       guardianOneFirstName.setWidthFull();
       personalInformation.add(guardianOneFirstName);
                      Paragraph guardianOneLastName = new Paragraph ("Guardian One Last Name: " + listOfStudents.get(index).getGuardianOneLastName());
                      guardianOneLastName.setWidthFull();
                      personalInformation.add(guardianOneLastName);
                      Paragraph guardianOnePhoneNum = new Paragraph ("Guardian One Phone Number: " + listOfStudents.get(index).getGuardianOnePho-
neNumber());
       guardianOnePhoneNum.setWidthFull();
       personalInformation.add(guardianOnePhoneNum);
                      Paragraph guardianOneEmail = new Paragraph ("Guardian One Email: " + listOfStudents.get(index).getGuardianOneEmail());
       guardianOneEmail.setWidthFull();
       personalInformation.add(guardianOneEmail);
       if (listOfStudents.get(index).isGuardianOneCallAtWork() == false) {
         Paragraph guardianOneCall = new Paragraph ("Call This Guardian At Work: No");
         guardianOneCall.setWidthFull();
         personalInformation.add(guardianOneCall);
       } else {
         Paragraph guardianOneCall = new Paragraph ("Call This Guardian At Work: Yes");
         guardianOneCall.setWidthFull();
         personalInformation.add(guardianOneCall);
       }
       if (!(listOfStudents.get(index).getGuardianTwoFirstName().equals("n/a"))) {
         Paragraph guardianTwoFirstName = new Paragraph ("Guardian Two First Name: " + listOfStudents.get(index).getGuardianTwoFirstName());
         guardianTwoFirstName.setWidthFull();
         personalInformation.add(guardianTwoFirstName);
         Paragraph guardianTwoLastName = new Paragraph ("Guardian Two Last Name: " + listOfStudents.get(index).getGuardianTwoLastName());
         guardianTwoLastName.setWidthFull();
```

```
          personalInformation.add(guardianTwoLastName);
          Paragraph guardianTwoPhoneNum = new Paragraph ("Guardian Two Phone Number: " + listOfStudents.get(index).getGuardianTwoPhoneNumber());
          guardianTwoPhoneNum.setWidthFull();
          personalInformation.add(guardianTwoPhoneNum);
          Paragraph guardianTwoEmail = new Paragraph ("Guardian Two Email: " + listOfStudents.get(index).getGuardianTwoEmail());
          guardianTwoEmail.setWidthFull();
          personalInformation.add(guardianTwoEmail);
        if (listOfStudents.get(index).isGuardianOneCallAtWork() == false) {
          Paragraph guardianTwoCall = new Paragraph ("Call This Guardian At Work: No");
          guardianTwoCall.setWidthFull();
          personalInformation.add(guardianTwoCall);
        } else {
          Paragraph guardianTwoCall = new Paragraph ("Call This Guardian At Work: Yes");
          guardianTwoCall.setWidthFull();
          personalInformation.add(guardianTwoCall);
        }
        } else {
        Paragraph guardianTwoFirstName = new Paragraph ("Guardian Two First Name: " + listOfStudents.get(index).getGuardianTwoFirstName());
        guardianTwoFirstName.setWidthFull();
        personalInformation.add(guardianTwoFirstName);
        Paragraph guardianTwoLastName = new Paragraph ("Guardian Two Last Name: " + listOfStudents.get(index).getGuardianTwoLastName());
        guardianTwoLastName.setWidthFull();
        personalInformation.add(guardianTwoLastName);
        Paragraph guardianTwoPhoneNum = new Paragraph ("Guardian Two Phone Number: " + listOfStudents.get(index).getGuardianTwoPhoneNumber());
        guardianTwoPhoneNum.setWidthFull();
        personalInformation.add(guardianTwoPhoneNum);
        Paragraph guardianTwoEmail = new Paragraph ("Guardian Two Email: " + listOfStudents.get(index).getGuardianTwoEmail());
        guardianTwoEmail.setWidthFull();
        personalInformation.add(guardianTwoEmail);
        Paragraph guardianTwoCall = new Paragraph ("Call This Guardian At Work: n/a");
        guardianTwoCall.setWidthFull();
        personalInformation.add(guardianTwoCall);
        }

              // Emergency Contact Information
              H3 emergencyContactTitle = new H3("Emergency Contact Information");

              Paragraph contactOneFirstName = new Paragraph("Contact One First Name: " + listOfStudents.get(index).getEmergencyContactOneFirst-
Name());
              contactOneFirstName.setWidthFull();
              Paragraph contactOneLastName = new Paragraph ("Contact One Last Name: " + listOfStudents.get(index).getEmergencyContac-
tOneLastName());
              contactOneLastName.setWidthFull();
              Paragraph relationshipOne = new Paragraph("Relationship: " + listOfStudents.get(index).getEmergencyContactOneRelationship());
              relationshipOne.setWidthFull();
              Paragraph homeOne = new Paragraph("Home Number: " + listOfStudents.get(index).getEmergencyContactOneHomeNumber());
              homeOne.setWidthFull();
              Paragraph cellOne = new Paragraph("Cell Number: " + listOfStudents.get(index).getEmergencyContactOneCellNumber());
              cellOne.setWidthFull();

              Paragraph filler = new Paragraph ("");
              filler.setWidthFull();
              Paragraph contactTwoFirstName = new Paragraph("Contact Two First Name: " + listOfStudents.get(index).getEmergencyContactTwoFirst-
Name());
              contactTwoFirstName.setWidthFull();
              Paragraph contactTwoLastName = new Paragraph ("Contact Two Last Name: " + listOfStudents.get(index).getEmergencyContactT-
woLastName());
              contactTwoLastName.setWidthFull();
              Paragraph relationshipTwo = new Paragraph("Relationship: " + listOfStudents.get(index).getEmergencyContactTwoRelationship());
              relationshipTwo.setWidthFull();
              Paragraph homeTwo = new Paragraph("Home Number: " + listOfStudents.get(index).getEmergencyContactTwoHomeNumber());
              homeTwo.setWidthFull();
              Paragraph cellTwo = new Paragraph("Cell Number: " + listOfStudents.get(index).getEmergencyContactTwoCellNumber());
              cellTwo.setWidthFull();
              Section emergencyContactInformation = new Section(emergencyContactTitle, contactOneFirstName, contactOneLastName, relation-
shipOne, homeOne, cellOne, filler, contactTwoFirstName, contactTwoLastName, relationshipTwo, homeTwo, cellTwo);

              //Health Information
              H3 healthInformationTitle = new H3("Health Information");
              Section healthInformation = new Section(healthInformationTitle);
              if (listOfStudents.get(index).getHealthFactorOne().equals("n/a")) {
                    Paragraph none = new Paragraph("No health factors were provided.");
                    none.setWidthFull();
                    healthInformation.add(none);
              }
              if(!(listOfStudents.get(index).getHealthFactorOne().equals("n/a"))) {
                  Paragraph healthOne = new Paragraph("Health Factor 1: " + listOfStudents.get(index).getHealthFactorOne());
                  healthOne.setWidthFull();
                  healthInformation.add(healthOne);
                  if ((listOfStudents.get(index).isHealthFactorOneLifeThreatening()) == true) {
                    Paragraph threateningOne = new Paragraph("Life Threatening: yes");
                    threateningOne.setWidthFull();
                    healthInformation.add(threateningOne);
                  } else {
                     Paragraph threateningOne = new Paragraph("Life Threatening: no");
                               threateningOne.setWidthFull();
                               healthInformation.add(threateningOne);
                  }
                  if ((listOfStudents.get(index).isHealthFactorOnePlanOfCareRequired()) == true) {
                    Paragraph careOne = new Paragraph ("Plan Of Care Required: yes");
                    careOne.setWidthFull();
                    healthInformation.add(careOne);
                  } else {
                     Paragraph careOne = new Paragraph ("Plan Of Care Required: no");
                               careOne.setWidthFull();
                               healthInformation.add(careOne);
                  }
                  if ((listOfStudents.get(index).isHealthFactorOneMedicationsRequired()) == true) {
                    Paragraph medicationsOne = new Paragraph("Medications Required: yes");
                    medicationsOne.setWidthFull();
                    healthInformation.add(medicationsOne);
```

```
  } else {
     Paragraph medicationsOne = new Paragraph("Medications Required: no");
               medicationsOne.setWidthFull();
               healthInformation.add(medicationsOne);
  }

} else {
       Paragraph healthFactor1 = new Paragraph ("Health Factor 1: n/a");
       healthFactor1.setWidthFull();
       healthInformation.add(healthFactor1);
       Paragraph lifeThreatening1 = new Paragraph ("Life Threatening: n/a");
       lifeThreatening1.setWidthFull();
       healthInformation.add(lifeThreatening1);
       Paragraph care1 = new Paragraph ("Plan Of Care: n/a");
       care1.setWidthFull();
       healthInformation.add(care1);
       Paragraph medications1 = new Paragraph ("Medications Required: n/a");
       medications1.setWidthFull();
       healthInformation.add(medications1);
}
if(!(listOfStudents.get(index).getHealthFactorTwo().equals("n/a"))) {
       Paragraph healthTwo = new Paragraph("Health Factor 2: " + listOfStudents.get(index).getHealthFactorTwo());
             healthTwo.setWidthFull();
             healthInformation.add(healthTwo);
  if ((listOfStudents.get(index).isHealthFactorTwoLifeThreatening()) == true) {
     Paragraph threateningTwo = new Paragraph("Life Threatening: yes");
               threateningTwo.setWidthFull();
               healthInformation.add(threateningTwo);
  } else {
     Paragraph threateningTwo = new Paragraph("Life Threatening: no");
               threateningTwo.setWidthFull();
               healthInformation.add(threateningTwo);
  }
  if ((listOfStudents.get(index).isHealthFactorTwoPlanOfCareRequired()) == true) {
     Paragraph careTwo = new Paragraph ("Plan Of Care Required: yes");
               careTwo.setWidthFull();
               healthInformation.add(careTwo);
  } else {
     Paragraph careTwo = new Paragraph ("Plan Of Care Required: no");
               careTwo.setWidthFull();
               healthInformation.add(careTwo);
  }
  if ((listOfStudents.get(index).isHealthFactorTwoMedicationsRequired()) == true) {
     Paragraph medicationsTwo = new Paragraph("Medications Required: yes");
              medicationsTwo.setWidthFull();
              healthInformation.add(medicationsTwo);
  } else {
     Paragraph medicationsTwo = new Paragraph("Medications Required: no");
              medicationsTwo.setWidthFull();
              healthInformation.add(medicationsTwo);
  }
} else {
       Paragraph healthFactor2 = new Paragraph ("Health Factor 2: n/a");
       healthFactor2.setWidthFull();
       healthInformation.add(healthFactor2);
       Paragraph lifeThreatening3 = new Paragraph ("Life Threatening: n/a");
       lifeThreatening3.setWidthFull();
       healthInformation.add(lifeThreatening3);
       Paragraph care3 = new Paragraph ("Plan Of Care: n/a");
       care3.setWidthFull();
       healthInformation.add(care3);
       Paragraph medications3 = new Paragraph ("Medications Required: n/a");
       medications3.setWidthFull();
       healthInformation.add(medications3);
}
if(!(listOfStudents.get(index).getHealthFactorThree().equals("n/a"))) {
       Paragraph healthThree = new Paragraph("Health Factor 3: " + listOfStudents.get(index).getHealthFactorThree());
             healthThree.setWidthFull();
             healthInformation.add(healthThree);
  if ((listOfStudents.get(index).isHealthFactorThreeLifeThreatening()) == true) {
     Paragraph threateningThree = new Paragraph("Life Threatening: yes");
              threateningThree.setWidthFull();
              healthInformation.add(threateningThree);
  } else {
     Paragraph threateningThree = new Paragraph("Life Threatening: no");
              threateningThree.setWidthFull();
              healthInformation.add(threateningThree);
  }
  if ((listOfStudents.get(index).isHealthFactorThreePlanOfCareRequired()) == true) {
     Paragraph careThree = new Paragraph ("Plan Of Care Required: yes");
              careThree.setWidthFull();
              healthInformation.add(careThree);
  } else {
     Paragraph careThree = new Paragraph ("Plan Of Care Required: no");
              careThree.setWidthFull();
              healthInformation.add(careThree);
  }
  if ((listOfStudents.get(index).isHealthFactorThreeMedicationsRequired()) == true) {
     Paragraph medicationsThree = new Paragraph("Medications Required: Yes");
              medicationsThree.setWidthFull();
              healthInformation.add(medicationsThree);
  } else {
     Paragraph medicationsThree = new Paragraph("Medications Required: No");
              medicationsThree.setWidthFull();
              healthInformation.add(medicationsThree);
  }
} else {
       Paragraph healthFactor3 = new Paragraph ("Health Factor 3: n/a");
       healthFactor3.setWidthFull();
       healthInformation.add(healthFactor3);
       Paragraph lifeThreateningThree = new Paragraph ("Life Threatening: n/a");
```

```
                lifeThreateningThree.setWidthFull();
                healthInformation.add(lifeThreateningThree);
                Paragraph careThree = new Paragraph ("Plan Of Care: n/a");
                careThree.setWidthFull();
                healthInformation.add(careThree);
                Paragraph medicationsThree = new Paragraph ("Medications Required: n/a");
                medicationsThree.setWidthFull();
                healthInformation.add(medicationsThree);
            }



            // NOTE
            // We are using inline styles here to keep the example simple.
            // We recommend placing CSS in a separate style sheet and to
            // encapsulating the styling in a new component.
            Scroller scroller = new Scroller(new Div(studentInformation, personalInformation, emergencyContactInformation, healthInfor-
mation));
            scroller.setScrollDirection(Scroller.ScrollDirection.VERTICAL);
            scroller.getStyle()
                    .set("border-bottom", "1px solid var(--lumo-contrast-20pct)")
                    .set("padding", "var(--lumo-space-m)");
            add(scroller);

            // Footer
            Button done = new Button("Done", e -> {
                  UI.getCurrent().navigate("menu");
            });
            done.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
            done.getStyle().set("margin-right", "var(--lumo-space-s)");

            Button anotherStudent = new Button("Another Student", l ->{
                  Dialog dialog = new Dialog();
                  dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                  VerticalLayout dialogLayout = createDialogLayout(dialog);
                  dialog.add(dialogLayout);
                  dialog.open();
                  add(dialog);
            });
            anotherStudent.addThemeVariants(ButtonVariant.LUMO_TERTIARY);

            Footer footer = new Footer(done, anotherStudent);
            footer.getStyle().set("padding", "var(--lumo-space-wide-m)");
            add(footer);

            setAlignItems(Alignment.STRETCH);
            //setHeight("400px");
            //setMaxWidth("100%");
            setPadding(false);
            setSpacing(false);
            //setWidth("360px");
            getStyle().set("border", "1px solid var(--lumo-contrast-20pct)");
        }


         public static ArrayList <Student> fileOneOpen() {
               try {
                fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
              } catch (FileNotFoundException e) {
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
              }

              //programChosen  - CHECK CONSTRUCTORS
              //add health factors to printing out in emergency situation stuff

              String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tem-
pReasonAttendance, tempCovid, tempReasonCovid;

               Boolean[] dourDoneOrNot;
               Boolean todayDourDoneOrNot;
               int[] quarterNumDoneMonth;
               int todayQuartersDone, currentQuarter;
               Boolean[] numOfDourSaparasDoneMonth;
               Boolean todayDourSaparaDoneOrNot;
               int todayDourSaparaDone;
               int dourCurrentSapara, dourNextFill;

               String programChosen;
               String lastRecord;

               Boolean[] sabaqDoneOrNot;
               Boolean todaySabaqDoneOrNot;
               int[] linesMemorized;
               int todayLinesMemorized;
               int[] mistakesMade;
               int todayMistakesMade;
               Boolean[] numOfSaparasDoneMonth;
               Boolean todaySaparaFinished;
               int[] nameOfSaparasDoneMonth;
               int totalSaparasDone;
               int todaySaparaDone;
               String saparasDone;
               int currentSaparaMemorizing;
               int saparaNextFill = 0;

              int age;

              String tempDate;
```

```
                ArrayList<String> dates;


                String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
                String guardianOneEmail;
                Boolean guardianOneCallAtWork;
                String guardianTwoFirstName, guardianTwoLastName;
                String guardianTwoPhoneNumber;
                String guardianTwoEmail;
                Boolean guardianTwoCallAtWork;

                String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
                String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
                String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
                String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

                String healthFactorOne;
                Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
                String healthFactorTwo;
                Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
                String healthFactorThree;
                Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

                Attendance attendanceOfStudent;
                StudentProgress progressOfStudent;

                while (fileScanner.hasNextLine()) {
                       dourDoneOrNot = new Boolean[30];
                            quarterNumDoneMonth = new int[30];
                           numOfDourSaparasDoneMonth = new Boolean[30];
                           sabaqDoneOrNot = new Boolean[30];
                            linesMemorized = new int[30];
                            mistakesMade = new int[30];
                            numOfSaparasDoneMonth = new Boolean[30];
                           nameOfSaparasDoneMonth = new int[30];

                  firstName = (fileScanner.nextLine()).toLowerCase();
                  middleName = (fileScanner.nextLine()).toLowerCase();
                  lastName = (fileScanner.nextLine()).toLowerCase();
                  address = (fileScanner.nextLine()).toLowerCase();
                  dateOfBirth = fileScanner.nextLine();
                  age = Integer.parseInt(fileScanner.nextLine());
                  postalCode = (fileScanner.nextLine()).toLowerCase();
                  language = (fileScanner.nextLine()).toLowerCase();
                  countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                  //progress of student
                  programChosen = (fileScanner.nextLine()).toLowerCase();
                  progressOfStudent = new StudentProgress();
                  progressOfStudent.setProgramChosen(programChosen);

                  lastRecord = (fileScanner.nextLine());
                  progressOfStudent.setLastRecord(lastRecord);


                  String tempDourDoneOrNot = fileScanner.nextLine();
                  String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                  for (int i = 0; i < strDourDoneOrNot.length; i++) {
                    dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                  }
            progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                  String tempQuarterNumDoneMonth = fileScanner.nextLine();
                  String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                  for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                    quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                  }
            progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                  currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuarter(currentQuarter);

                  String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                  String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                  for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                    numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                  }
            progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                  dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCur-
rentSapara);

                  dourNextFill = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenDourNextFill(dourNextFill);

                  DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                     LocalDateTime firstNow = LocalDateTime.now();
                    String alreadyDone = firstFormatter.format(firstNow);

                  if (!(alreadyDone.equals(lastRecord))) {
                    if (programChosen.equals("hafiz")) {
                      Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      int holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                       temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());

                      todayDourDoneOrNot = false;
```

```
                todayDourSaparaDoneOrNot = false;
                todayQuartersDone = 0;
                todayDourSaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

              } else {
                Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                int holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                todayDourDoneOrNot = false;
                 todayQuartersDone = 0;
                 todayDourSaparaDoneOrNot = false;
                todayDourSaparaDone = 0;
                 todaySabaqDoneOrNot = false;
                todayLinesMemorized = 0;
                todayMistakesMade = 0;
                todaySaparaFinished = false;
                todaySaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
            progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
            progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                }


                } else {
                  todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                  progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                  todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                  progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                  todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                 progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

                 todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                 progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                  if (!(programChosen.equals("hafiz"))) {

                todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(to-
daySabaqDoneOrNot);

                  todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                  todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                  todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
            progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                  todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                } else {
                  Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                  int holder = Integer.parseInt(fileScanner.nextLine());
                  holder = Integer.parseInt(fileScanner.nextLine());
                  temporary = Boolean.parseBoolean(fileScanner.nextLine());
                  holder = Integer.parseInt(fileScanner.nextLine());
                }
              }



              if (!(programChosen.equals("hafiz"))) {
                String tempSabaqDoneOrNot = fileScanner.nextLine();
                String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
                for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
                  sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
                }
            progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

                String tempLinesMemorized = fileScanner.nextLine();
                String strLinesMemorized[] = tempLinesMemorized.split(",");
                for (int i = 0; i < strLinesMemorized.length; i++) {
                  linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
                }
                progressOfStudent.setOpenLinesMemorized(linesMemorized);

                String tempMistakesMade = fileScanner.nextLine();
                String strMistakesMade[] = tempMistakesMade.split(",");
                for (int i = 0; i < strMistakesMade.length; i++) {
                  mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
                }
                progressOfStudent.setOpenMistakesMade(mistakesMade);
```

```
      String tempNumOfSaparasFinished = fileScanner.nextLine();
      String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
      for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
       numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

      String tempNameOfSaparasFinished = fileScanner.nextLine();
      String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
      for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
       nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

      totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

      saparasDone = fileScanner.nextLine();
      progressOfStudent.setOpenSaparasDone(saparasDone);

      currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

      saparaNextFill = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
      } else {
        String hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
      }


      //attendance

      tempAttendance = fileScanner.nextLine();
      String attendance[] = tempAttendance.split(",");
      attendanceOfStudent = new Attendance();
      for (int i = 0; i < attendance.length; i++ ) {
        attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
        }
      tempReasonAttendance = fileScanner.nextLine();
      String tempReason[] = tempReasonAttendance.split(",");
      for (int i = 0; i < tempReason.length; i++ ) {
        attendanceOfStudent.addReasonAbsent(tempReason[i]);
        }
      tempCovid = fileScanner.nextLine();
      String covid[] = tempCovid.split(",");
      for (int i = 0; i < covid.length; i++ ) {
        attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
        }
      tempReasonCovid = fileScanner.nextLine();
      String reasonCov[] = tempReasonCovid.split(",");
      for (int i = 0; i < reasonCov.length; i++ ) {
        attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
        }

        dates = new ArrayList<String>();

        tempDate = fileScanner.nextLine();
      String date[] = tempDate.split(",");
      for (int i = 0; i < date.length; i++ ) {
        dates.add(date[i]);
        }

        guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
        guardianOnePhoneNumber = fileScanner.nextLine();
        guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
        guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
        guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoPhoneNumber = fileScanner.nextLine();
        guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
        guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

        emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneHomeNumber = (fileScanner.nextLine());
        emergencyContactOneCellNumber = (fileScanner.nextLine());
        emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoHomeNumber = (fileScanner.nextLine());
        emergencyContactTwoCellNumber = (fileScanner.nextLine());

        healthFactorOne = (fileScanner.nextLine()).toLowerCase();
        healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
        healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorThree = (fileScanner.nextLine()).toLowerCase();
```

```
                    healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

                Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,coun-
tryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail,
guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardianTwoCallAtWork, emergencyCon-
tactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCell-
Number,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyCon-
tactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired,
healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree,
healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                  listOfStudents.add(tempS);
                }
                fileScanner.close();

                return listOfStudents;
               }

                        //access stored index of student in temp file
                        public static int index () {
                                  int index = -1;
                                   try {
                                       fileScanner = new Scanner(new File("temp.txt"));
                                       index = Integer.parseInt(fileScanner.nextLine());

                                              fileScanner.close();
                                     } catch (FileNotFoundException e) {
                                       System.err.println("File not found! Choosing to quit now...");

                                       System.exit(0);
                                     }
                                    return index;
                        }

                          private static VerticalLayout createDialogLayout(Dialog dialog) {
                       H2 headline = new H2("Enter Student Information");
                       headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                               .set("font-size", "1.5em").set("font-weight", "bold");

                       //take in student's first and last name
                       TextField firstNameField = new TextField("First Name");
                       TextField lastNameField = new TextField("Last Name");
                       //styling of fields
                       VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                               lastNameField);
                       fieldLayout.setSpacing(false);
                       fieldLayout.setPadding(false);
                       fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                       //cancel button tp leave menu
                       Button cancelButton = new Button("Cancel", e -> dialog.close());
                       //done button to start search
                       Button saveButton = new Button("Done", e -> {
                          int index = -2;
                           boolean found = false;
                           //search through listOfStudents ArrayList for a match with the entered first and last names
                          for (int i = 0; i < listOfStudents.size(); i++) {
                                    //if match is found
                               if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.get-
Value().equals(listOfStudents.get(i).getLastName())) {
                                 index = i;
                                 found = true;
                                 //store index into temp.txt file and close dialog
                                 store(index);
                                             dialog.close();
                                            //reload page and navigate to desired page
                                      UI.getCurrent().getPage().reload();
                                 UI.getCurrent().navigate("menuBRecordsV");
                                 //break for loop
                                 break;
                               }
                             }
                          //if match was not found, display a warning message
                             if (found == false) {
                                       Notification.show("Invalid name entered.",
                                         3000, Notification.Position.MIDDLE);
                             }


                       });
                       saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                       HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                               saveButton);
                       buttonLayout
                               .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                       VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                               buttonLayout);
                       dialogLayout.setPadding(false);
                       dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                       dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                       return dialogLayout;
                   }

                          public static void store(int index) {
                                      PrintWriter pw = null;
                                 try {
                                    pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                                    pw.println(index);
                                    pw.close();
```

```
                        } catch (FileNotFoundException e) {
                          System.err.print("couldn't open file for writing!");
                          System.exit(0);
                        }
                           }

}
```

CLASS: MenuBRecordsE.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.accordion.Accordion;
import com.vaadin.flow.component.formlayout.FormLayout;
import com.vaadin.flow.component.formlayout.FormLayout.ResponsiveStep;
import com.vaadin.flow.component.html.Footer;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H4;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.router.Route;

import com.vaadin.flow.component.accordion.AccordionPanel;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.checkbox.Checkbox;
import com.vaadin.flow.component.datepicker.DatePicker;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.textfield.IntegerField;
import com.vaadin.flow.data.binder.Binder;

@Route(value = "menuBRecordsE", layout = Welcome.class)
public class MenuBRecordsE extends VerticalLayout {

           static Scanner fileScanner;
           static ArrayList <Student> listOfStudents = new ArrayList <Student>();
           static int index;

             private static final String studentInformation = "Student Information";
             private static final String guardianInformation = "Parent/Guardian Information";
             private static final String emergencyContacts = "Emergency Contacts";
             private static final String healthInformation = "Health Information";

           public MenuBRecordsE() {

                      //read from files
                     listOfStudents.removeAll(listOfStudents);
                     listOfStudents = fileOneOpen();
                     index();

                     //introduce
                     H2 intro = new H2 ("Student Records");
                     intro.setMinWidth("700px");
               intro.setSizeFull();
               intro.getStyle().set("text-align", "center");

               H4 intro2 = new H4("Editing For: " + listOfStudents.get(index).getFullName());
                     intro2.setMinWidth("700px");
        intro2.setSizeFull();
        intro2.getStyle().set("text-align", "center");

        add(intro, intro2);

                     //make new accordian
                     Accordion accordion = new Accordion();
                     Binder<Student> personBinder = new Binder<>(Student.class);
               personBinder.setBean(listOfStudents.get(index));

                     FormLayout studentInformationFormLayout = createFormLayout();
                     AccordionPanel studentInformationPanel = accordion.add(studentInformation, studentInformationFormLayout );
                     FormLayout guardianInformationFormLayout = createFormLayout();
        AccordionPanel guardianInformationPanel = accordion.add(guardianInformation, guardianInformationFormLayout);
        FormLayout emergencyContactsFormLayout = createFormLayout();
        AccordionPanel emergencyContactPanel = accordion.add(emergencyContacts, emergencyContactsFormLayout);
        FormLayout healthInformationFormLayout = createFormLayout();
        AccordionPanel healthInformationPanel = accordion.add(healthInformation, healthInformationFormLayout);


                         TextField fName = new TextField("First Name");
                         personBinder.forField(fName).bind(
                          Student::getFirstName,
                          Student::setFirstName);

                         TextField middleName = new TextField("Middle Name");
                         personBinder.forField(middleName).bind(
                                         Student::getMiddleName,
                                         Student::setMiddleName);

                         TextField lastName = new TextField("Last Name");
```

```
        personBinder.forField(lastName).bind(
         Student::getLastName,
         Student::setLastName
                        );

        TextField address = new TextField("Address");
        personBinder.forField(address).bind(
                        Student::getAddress,
                        Student::setAddress);


        DatePicker datePicker = new DatePicker("Date Of Birth");
        personBinder.forField(datePicker).bind(
                        Student::getDateOfBirthLocalDate,
                        Student::setDateOfBirthLocalDate
                        );

        IntegerField age = new IntegerField("Age");
        personBinder.forField(age).bind(
                        Student::getAge,
                        Student::setAge
                        );

        TextField postalCode = new TextField("Postal Code");
        personBinder.forField(postalCode).bind(
                        Student::getPostalCode,
                        Student::setPostalCode);

        TextField language = new TextField("Language");
        personBinder.forField(language).bind(
                        Student::getLanguage,
                        Student::setLanguage);

        TextField country = new TextField("Country of Birth");
        personBinder.forField(country).bind(
                        Student::getCountryOfBirth,
                        Student::setCountryOfBirth);

        studentInformationFormLayout.add(fName, middleName, lastName);
        studentInformationFormLayout.add(address, 2);
        studentInformationFormLayout.add(datePicker, age, language, country);

        studentInformationPanel.addOpenedChangeListener(e -> {
    if(e.isOpened()) {
            studentInformationPanel.setSummaryText(studentInformation);
    }
});

Button customDetailsButton = new Button("Done", (e) -> {
   guardianInformationPanel.setOpened(true);
   closeFileOne(listOfStudents);
}
);
customDetailsButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
studentInformationPanel.addContent(customDetailsButton);
accordion.setWidth("975px");

TextField gFirstName = new TextField("Guardian One First Name");
        personBinder.forField(gFirstName).bind(
         Student::getGuardianOneFirstName,
         Student::setGuardianOneFirstName);

        TextField gLastName = new TextField("Guardian One Last Name");
        personBinder.forField(gLastName).bind(
         Student::getGuardianOneLastName,
         Student::setGuardianOneLastName);

        TextField gPhoneNum = new TextField("Guardian One Phone Number");
        personBinder.forField(gPhoneNum).bind(
         Student::getGuardianOnePhoneNumber,
         Student::setGuardianOnePhoneNumber);

        TextField gEmail = new TextField("Guardian One Email");
        personBinder.forField(gEmail).bind(
         Student::getGuardianOneEmail,
         Student::setGuardianOneEmail);

Checkbox callAtWork1 = new Checkbox();
callAtWork1.setLabel("Call Guardian One at Work");
personBinder.forField(callAtWork1).bind(
         Student::isGuardianOneCallAtWork,
         Student::setGuardianOneCallAtWork);

TextField gFirstName2 = new TextField("Guardian Two First Name");
        personBinder.forField(gFirstName2).bind(
         Student::getGuardianTwoFirstName,
         Student::setGuardianTwoFirstName);

        TextField gLastName2 = new TextField("Guardian Two Last Name");
        personBinder.forField(gLastName2).bind(
         Student::getGuardianTwoLastName,
         Student::setGuardianTwoLastName);

        TextField gPhoneNum2 = new TextField("Guardian Two Phone Number");
        personBinder.forField(gPhoneNum2).bind(
         Student::getGuardianTwoPhoneNumber,
         Student::setGuardianTwoPhoneNumber);

        TextField gEmail2 = new TextField("Guardian Two Email");
        personBinder.forField(gEmail2).bind(
         Student::getGuardianTwoEmail,
```

```
                Student::setGuardianTwoEmail);

        Checkbox callAtWork2 = new Checkbox();
        callAtWork2.setLabel("Call Guardian Two at Work");
        personBinder.forField(callAtWork2).bind(
                Student::isGuardianTwoCallAtWork,
                Student::setGuardianTwoCallAtWork);

        guardianInformationFormLayout.add(gFirstName, gLastName, gPhoneNum);
        guardianInformationFormLayout.add(gEmail, 3);
        guardianInformationFormLayout.add(callAtWork1, 3);
        guardianInformationFormLayout.add(gFirstName2, gLastName2, gPhoneNum2);
        guardianInformationFormLayout.add(gEmail2, 3);
        guardianInformationFormLayout.add(callAtWork2, 3);

        guardianInformationPanel.addOpenedChangeListener(e -> {
            if(e.isOpened()) {
                    guardianInformationPanel.setSummaryText(guardianInformation);
            }
        });

        Button gInformationButton = new Button("Done", (e) -> {
           closeFileOne(listOfStudents);
           emergencyContactPanel.setOpened(true);
           }
                   );
        gInformationButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                   guardianInformationPanel.addContent(gInformationButton);

        TextField eContactOneFirstName = new TextField("Contact One First Name");
               personBinder.forField(eContactOneFirstName).bind(
                Student::getEmergencyContactOneFirstName,
                Student::setEmergencyContactOneFirstName);

               TextField eContactOneLastName = new TextField("Contact One Last Name");
               personBinder.forField(eContactOneLastName).bind(
                Student::getEmergencyContactOneLastName,
                Student::setEmergencyContactOneLastName);

               TextField eContactOneRelationship = new TextField("Contact One Relationship");
               personBinder.forField(eContactOneRelationship).bind(
                Student::getEmergencyContactOneRelationship,
                Student::setEmergencyContactOneRelationship);

               TextField eContactOneHomeNumber = new TextField("Contact One Home Number");
               personBinder.forField(eContactOneHomeNumber).bind(
                Student::getEmergencyContactOneHomeNumber,
                Student::setEmergencyContactOneHomeNumber);

               TextField eContactOneCellNumber = new TextField("Contact One Cell Number");
               personBinder.forField(eContactOneCellNumber).bind(
                Student::getEmergencyContactOneCellNumber,
                Student::setEmergencyContactOneCellNumber);

               TextField eContactTwoFirstName = new TextField("Contact Two First Name");
               personBinder.forField(eContactTwoFirstName).bind(
                Student::getEmergencyContactTwoFirstName,
                Student::setEmergencyContactTwoFirstName);

               TextField eContactTwoLastName = new TextField("Contact Two Last Name");
               personBinder.forField(eContactTwoLastName).bind(
                Student::getEmergencyContactTwoLastName,
                Student::setEmergencyContactTwoLastName);

               TextField eContactTwoRelationship = new TextField("Contact Two Relationship");
               personBinder.forField(eContactTwoRelationship).bind(
                Student::getEmergencyContactTwoRelationship,
                Student::setEmergencyContactTwoRelationship);

               TextField eContactTwoHomeNumber = new TextField("Contact Two Home Number");
               personBinder.forField(eContactTwoHomeNumber).bind(
                Student::getEmergencyContactTwoHomeNumber,
                Student::setEmergencyContactTwoHomeNumber);

               TextField eContactTwoCellNumber = new TextField("Contact Two Cell Number");
               personBinder.forField(eContactTwoCellNumber).bind(
                Student::getEmergencyContactTwoCellNumber,
                Student::setEmergencyContactTwoCellNumber);

               emergencyContactsFormLayout.add(eContactOneFirstName, eContactOneLastName, eContactOneRelationship, eContactOneHome-
Number);
               emergencyContactsFormLayout.add(eContactOneCellNumber, 2);
               emergencyContactsFormLayout.add(eContactTwoFirstName, eContactTwoLastName, eContactTwoRelationship, eContactTwoHome-
Number);
               emergencyContactsFormLayout.add(eContactTwoCellNumber, 2);

               emergencyContactPanel.addOpenedChangeListener(e -> {
            if(e.isOpened()) {
                    emergencyContactPanel.setSummaryText(emergencyContacts);
            }
        });

        Button eContactButton = new Button("Done", (e) -> {
           closeFileOne(listOfStudents);
           healthInformationPanel.setOpened(true);
           }
                   );
        eContactButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
        emergencyContactPanel.addContent(eContactButton);

               TextField healthFactorOneName = new TextField("Health Factor One");
```

```
        personBinder.forField(healthFactorOneName).bind(
         Student::getHealthFactorOne,
         Student::setHealthFactorOne);

        Checkbox lifeThreatening1 = new Checkbox();
        lifeThreatening1.setLabel("Life Threatening");
personBinder.forField(lifeThreatening1).bind(
         Student::isHealthFactorOneLifeThreatening,
         Student::setHealthFactorOneLifeThreatening);

Checkbox planOfCareRequired1 = new Checkbox();
planOfCareRequired1.setLabel("Plan Of Care Required");
personBinder.forField(planOfCareRequired1).bind(
         Student::isHealthFactorOnePlanOfCareRequired,
         Student::setHealthFactorOnePlanOfCareRequired);

Checkbox medicationsRequired1 = new Checkbox();
medicationsRequired1.setLabel("Medications Required");
personBinder.forField(medicationsRequired1).bind(
         Student::isHealthFactorOneMedicationsRequired,
         Student::setHealthFactorOneMedicationsRequired);

TextField healthFactorTwoName = new TextField("Health Factor Two");
        personBinder.forField(healthFactorTwoName).bind(
         Student::getHealthFactorTwo,
         Student::setHealthFactorTwo);

        Checkbox lifeThreatening2 = new Checkbox();
        lifeThreatening2.setLabel("Life Threatening");
personBinder.forField(lifeThreatening2).bind(
         Student::isHealthFactorTwoLifeThreatening,
         Student::setHealthFactorTwoLifeThreatening);

Checkbox planOfCareRequired2 = new Checkbox();
planOfCareRequired2.setLabel("Plan Of Care Required");
personBinder.forField(planOfCareRequired2).bind(
         Student::isHealthFactorTwoPlanOfCareRequired,
         Student::setHealthFactorTwoPlanOfCareRequired);

Checkbox medicationsRequired2 = new Checkbox();
medicationsRequired2.setLabel("Medications Required");
personBinder.forField(medicationsRequired2).bind(
         Student::isHealthFactorTwoMedicationsRequired,
         Student::setHealthFactorTwoMedicationsRequired);

TextField healthFactorThreeName = new TextField("Health Factor Three");
        personBinder.forField(healthFactorThreeName).bind(
         Student::getHealthFactorThree,
         Student::setHealthFactorThree);

        Checkbox lifeThreatening3 = new Checkbox();
        lifeThreatening3.setLabel("Life Threatening");
personBinder.forField(lifeThreatening3).bind(
         Student::isHealthFactorThreeLifeThreatening,
         Student::setHealthFactorThreeLifeThreatening);

Checkbox planOfCareRequired3 = new Checkbox();
planOfCareRequired3.setLabel("Plan Of Care Required");
personBinder.forField(planOfCareRequired3).bind(
         Student::isHealthFactorThreePlanOfCareRequired,
         Student::setHealthFactorThreePlanOfCareRequired);

Checkbox medicationsRequired3 = new Checkbox();
medicationsRequired3.setLabel("Medications Required");
personBinder.forField(medicationsRequired3).bind(
         Student::isHealthFactorThreeMedicationsRequired,
         Student::setHealthFactorThreeMedicationsRequired);

        healthInformationFormLayout.add(healthFactorOneName, 3);
        healthInformationFormLayout.add(lifeThreatening1, planOfCareRequired1, medicationsRequired1);

        healthInformationFormLayout.add(healthFactorTwoName, 3);
        healthInformationFormLayout.add(lifeThreatening2, planOfCareRequired2, medicationsRequired2);

        healthInformationFormLayout.add(healthFactorThreeName, 3);
        healthInformationFormLayout.add(lifeThreatening3, planOfCareRequired3, medicationsRequired3);

        healthInformationPanel.addOpenedChangeListener(e -> {
    if(e.isOpened()) {
            healthInformationPanel.setSummaryText(healthInformation);
    }
});

Button hInformationButton = new Button("Done", (e) -> {
   healthInformationPanel.setOpened(false);
   closeFileOne(listOfStudents);
});
hInformationButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
healthInformationPanel.addContent(hInformationButton);

// Footer
        Button done = new Button("Done", e -> {
            closeFileOne(listOfStudents);
            UI.getCurrent().navigate("menu");
        });
        done.addThemeVariants(ButtonVariant.LUMO_PRIMARY, ButtonVariant.LUMO_CONTRAST);
        done.getStyle().set("margin-right", "var(--lumo-space-s)");

        Button anotherStudent = new Button("Another Student", l ->{
            closeFileOne(listOfStudents);
            Dialog dialog = new Dialog();
```

```
                dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                VerticalLayout dialogLayout = createDialogLayout(dialog);
                dialog.add(dialogLayout);
                dialog.open();
                add(dialog);
            });
            anotherStudent.addThemeVariants(ButtonVariant.LUMO_TERTIARY, ButtonVariant.LUMO_CONTRAST);

            Footer footer = new Footer(done, anotherStudent);
            footer.getStyle().set("padding", "var(--lumo-space-wide-m)");

            //setJustifyContentMode(JustifyContentMode.CENTER);
        //setDefaultHorizontalComponentAlignment(Alignment.CENTER);
        footer.getStyle().set("text-align", "center");

            setAlignItems(Alignment.STRETCH);
            //setHeight("400px");
            //setMaxWidth("100%");
            setPadding(false);
            setSpacing(false);
            //setWidth("360px");
            getStyle().set("border", "1px solid var(--lumo-contrast-20pct)");

        VerticalLayout temp = new VerticalLayout(accordion);
        temp.setPadding(true);
        add(temp);
        add(footer);


}



//accordian methods
    private FormLayout createFormLayout() {
        FormLayout billingAddressFormLayout = new FormLayout();
        billingAddressFormLayout.setResponsiveSteps(
                    new ResponsiveStep("0", 1),
                new ResponsiveStep("320px", 2),
                new ResponsiveStep("500px", 3)
        );
        return billingAddressFormLayout;
     }

    public static void closeFileOne(ArrayList <Student> listOfStudents) {
                PrintWriter pw = null;
                    try {
                       pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                    } catch (FileNotFoundException e) {
                      System.err.print("couldn't open file for writing!");
                      System.exit(0);
                    }



                    for (int y = 0; y < listOfStudents.size(); y++) {

                      if (y == 0) {
                     pw.println(listOfStudents.get(y).getFirstName());
                      } else {
                        pw.println(listOfStudents.get(y).getFirstName());
                      }
                      pw.println(listOfStudents.get(y).getMiddleName());
                     pw.println(listOfStudents.get(y).getLastName());
                     pw.println(listOfStudents.get(y).getAddress());
                     pw.println(listOfStudents.get(y).getDateOfBirth());
                     pw.println(listOfStudents.get(y).getAge());
                     pw.println(listOfStudents.get(y).getPostalCode());
                     pw.println(listOfStudents.get(y).getLanguage());
                     pw.println(listOfStudents.get(y).getCountryOfBirth());
           pw.println(listOfStudents.get(y).getProgramChosen());
                      pw.println(listOfStudents.get(y).getLastRecord());


                   String holder = "";
                  for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
                       if (k == 0) {
             holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
                      } else {
                      holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
                      }
                      }
                        pw.println(holder);

                        holder = "";
                for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
                       if (k == 0) {
                        holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
                      } else {
                      holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
                      }
                      }
                        pw.println(holder);

                pw.println(listOfStudents.get(y).getCurrentQuarter());

                holder = "";
                        for (int k = 0; k < listOfStudents.get(y).getNumOfDourSaparasDoneMonth().length; k++) {
                       if (k == 0) {
                        holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
```

```
            } else {
            holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
            }
            }
              pw.println(holder);
        pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStudents.get(y).getDourNextFill());

        DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
           LocalDateTime firstNow = LocalDateTime.now();
          String alreadyDone = firstFormatter.format(firstNow);

        if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
            pw.println(false);
            pw.println(-1);
            pw.println(false);
            pw.println(-1);
            pw.println(false);
            pw.println(-1);
            pw.println(-1);
            pw.println(false);
            pw.println(-1);
        } else {
             pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
             pw.println(listOfStudents.get(y).getTodayQuartersDone());
             pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
             pw.println(listOfStudents.get(y).getTodayDourSaparaDone());

          if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
                pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
                pw.println(listOfStudents.get(y).getTodayLinesMemorized());
                pw.println(listOfStudents.get(y).getTodayMistakesMade());
                pw.println(listOfStudents.get(y).isTodaySaparaFinished());
                pw.println(listOfStudents.get(y).getTodaySaparaDone());
          } else {
            pw.println(false);
            pw.println(-1);
            pw.println(-1);
            pw.println(false);
            pw.println(-1);
          }
        }



        if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
              holder = "";
          for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
              }
              }
          pw.println(holder);

          holder = "";
        for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
              }
              }
        pw.println(holder);

        holder = "";
        for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
              }
              }
        pw.println(holder);

        holder = "";
          for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
              }
              }
          pw.println(holder);

          holder = "";
          for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++) {
              if (k == 0) {
               holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
              } else {
              holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
              }
              }
          pw.println(holder);
        pw.println(listOfStudents.get(y).getTotalSaparasDone());
            pw.println(listOfStudents.get(y).getSaparasDone());
    pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStudents.get(y).getSaparaNextFill());
        } else {
          pw.println(false);
          pw.println(0);
          pw.println(0);
```

```
  pw.println(false);
  pw.println(0);
  pw.println(0);
  pw.println(0);
  pw.println(0);
  pw.println(0);
 }


 //attendance
//printing to file for attendance
 holder = "";
       for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

       if (k == 0) {
        holder = "" + (listOfStudents.get(y).getAttendance().get(k));

      } else {
      holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));

      }
         }
       pw.println(holder);

       holder = "";
      //printing to file for reason absent
      for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

       if (d == 0) {
        holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

      } else {
      holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

      }
        }
      pw.println(holder);
      //printing to file for covid screening
      holder = "";
      for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

       if (r == 0) {
        holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

      } else {
      holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

      }
    }
      pw.println(holder);

    //printing to file for reason covid screening was not done
      holder = "";
      for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++) {

       if (p == 0) {
        holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

      } else {
      holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreening().get(p));

      }
    }
      pw.println(holder);

    //printing to file for dates
      holder = "";
    for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

       if (z == 0) {
        holder = ""+(listOfStudents.get(y).getDate().get(z));

      } else {
      holder = holder + ("," + listOfStudents.get(y).getDate().get(z));

      }
    }
    pw.println(holder);
    pw.println(listOfStudents.get(y).getGuardianOneFirstName());
    pw.println(listOfStudents.get(y).getGuardianOneLastName());
    pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
    pw.println(listOfStudents.get(y).getGuardianOneEmail());
    pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
    pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
    pw.println(listOfStudents.get(y).getGuardianTwoLastName());
    pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
    pw.println(listOfStudents.get(y).getGuardianTwoEmail());
    pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

    pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
    pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
    pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
    pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
    pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
    pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
    pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
    pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
    pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
    pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

    pw.println(listOfStudents.get(y).getHealthFactorOne());
```

```
                    pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
                    pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
                    pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
                    pw.println(listOfStudents.get(y).getHealthFactorTwo());
                    pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
                    pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
                    pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
                    pw.println(listOfStudents.get(y).getHealthFactorThree());
                    pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
                    pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
                    pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                        }
                pw.close();


                    }


        public static ArrayList <Student> fileOneOpen() {
             try {
              fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
            } catch (FileNotFoundException e) {
              System.err.println("File not found! Choosing to quit now...");

              System.exit(0);
            }

            //programChosen  - CHECK CONSTRUCTORS
            //add health factors to printing out in emergency situation stuff

            String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tem-
pReasonAttendance, tempCovid, tempReasonCovid;

             Boolean[] dourDoneOrNot;
             Boolean todayDourDoneOrNot;
             int[] quarterNumDoneMonth;
             int todayQuartersDone, currentQuarter;
             Boolean[] numOfDourSaparasDoneMonth;
             Boolean todayDourSaparaDoneOrNot;
             int todayDourSaparaDone;
             int dourCurrentSapara, dourNextFill;

             String programChosen;
             String lastRecord;

             Boolean[] sabaqDoneOrNot;
             Boolean todaySabaqDoneOrNot;
             int[] linesMemorized;
             int todayLinesMemorized;
             int[] mistakesMade;
             int todayMistakesMade;
             Boolean[] numOfSaparasDoneMonth;
             Boolean todaySaparaFinished;
             int[] nameOfSaparasDoneMonth;
             int totalSaparasDone;
             int todaySaparaDone;
             String saparasDone;
             int currentSaparaMemorizing;
             int saparaNextFill = 0;

            int age;

            String tempDate;
            ArrayList<String> dates;


            String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
            String guardianOneEmail;
            Boolean guardianOneCallAtWork;
            String guardianTwoFirstName, guardianTwoLastName;
            String guardianTwoPhoneNumber;
            String guardianTwoEmail;
            Boolean guardianTwoCallAtWork;

            String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
            String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
            String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
            String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

            String healthFactorOne;
            Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
            String healthFactorTwo;
            Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
            String healthFactorThree;
            Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

            Attendance attendanceOfStudent;
            StudentProgress progressOfStudent;

            while (fileScanner.hasNextLine()) {
                   dourDoneOrNot = new Boolean[30];
                        quarterNumDoneMonth = new int[30];
                       numOfDourSaparasDoneMonth = new Boolean[30];
                       sabaqDoneOrNot = new Boolean[30];
                        linesMemorized = new int[30];
                        mistakesMade = new int[30];
                        numOfSaparasDoneMonth = new Boolean[30];
                       nameOfSaparasDoneMonth = new int[30];

              firstName = (fileScanner.nextLine()).toLowerCase();
```

```
                middleName = (fileScanner.nextLine()).toLowerCase();
                lastName = (fileScanner.nextLine()).toLowerCase();
                address = (fileScanner.nextLine()).toLowerCase();
                dateOfBirth = fileScanner.nextLine();
                age = Integer.parseInt(fileScanner.nextLine());
                postalCode = (fileScanner.nextLine()).toLowerCase();
                language = (fileScanner.nextLine()).toLowerCase();
                countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                //progress of student
                programChosen = (fileScanner.nextLine()).toLowerCase();
                progressOfStudent = new StudentProgress();
                progressOfStudent.setProgramChosen(programChosen);

                lastRecord = (fileScanner.nextLine());
                progressOfStudent.setLastRecord(lastRecord);


                String tempDourDoneOrNot = fileScanner.nextLine();
                String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                for (int i = 0; i < strDourDoneOrNot.length; i++) {
                  dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                }
            progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                String tempQuarterNumDoneMonth = fileScanner.nextLine();
                String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                  quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                }
            progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuarter(currentQuarter);

                String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                  numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                }
            progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCur-
rentSapara);

                dourNextFill = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenDourNextFill(dourNextFill);

                DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                   LocalDateTime firstNow = LocalDateTime.now();
                  String alreadyDone = firstFormatter.format(firstNow);

                if (!(alreadyDone.equals(lastRecord))) {
                  if (programChosen.equals("hafiz")) {
                    Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    int holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                     temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());

                    todayDourDoneOrNot = false;
                    todayDourSaparaDoneOrNot = false;
                    todayQuartersDone = 0;
                    todayDourSaparaDone = 0;
                    progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                  } else {
                    Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    int holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                     temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                    holder = Integer.parseInt(fileScanner.nextLine());
                    todayDourDoneOrNot = false;
                     todayQuartersDone = 0;
                     todayDourSaparaDoneOrNot = false;
                    todayDourSaparaDone = 0;
                     todaySabaqDoneOrNot = false;
                    todayLinesMemorized = 0;
                    todayMistakesMade = 0;
                    todaySaparaFinished = false;
                    todaySaparaDone = 0;
                    progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
            progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
            progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                }
```

```
                    } else {
                      todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                      progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                      todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                      progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                      todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                     progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

                     todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                     progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                      if (!(programChosen.equals("hafiz"))) {

                    todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(to-
daySabaqDoneOrNot);

                        todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                        todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                        todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
                progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                        todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                      } else {
                        Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        int holder = Integer.parseInt(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                        temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                      }
                    }



                    if (!(programChosen.equals("hafiz"))) {
                      String tempSabaqDoneOrNot = fileScanner.nextLine();
                    String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
                    for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
                      sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
                    }
                progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

                    String tempLinesMemorized = fileScanner.nextLine();
                    String strLinesMemorized[] = tempLinesMemorized.split(",");
                    for (int i = 0; i < strLinesMemorized.length; i++) {
                      linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
                    }
                    progressOfStudent.setOpenLinesMemorized(linesMemorized);

                    String tempMistakesMade = fileScanner.nextLine();
                    String strMistakesMade[] = tempMistakesMade.split(",");
                    for (int i = 0; i < strMistakesMade.length; i++) {
                      mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
                    }
                    progressOfStudent.setOpenMistakesMade(mistakesMade);

                    String tempNumOfSaparasFinished = fileScanner.nextLine();
                    String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
                    for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
                     numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
                    }
                    progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

                    String tempNameOfSaparasFinished = fileScanner.nextLine();
                    String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
                    for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
                     nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
                    }
                    progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

                    totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
                    progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

                    saparasDone = fileScanner.nextLine();
                    progressOfStudent.setOpenSaparasDone(saparasDone);

                    currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

                    saparaNextFill = Integer.parseInt(fileScanner.nextLine());
                    progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
                    } else {
                      String hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                    }
```

```
                //attendance

                tempAttendance = fileScanner.nextLine();
                String attendance[] = tempAttendance.split(",");
                attendanceOfStudent = new Attendance();
                for (int i = 0; i < attendance.length; i++ ) {
                  attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
                  }
                tempReasonAttendance = fileScanner.nextLine();
                String tempReason[] = tempReasonAttendance.split(",");
                for (int i = 0; i < tempReason.length; i++ ) {
                  attendanceOfStudent.addReasonAbsent(tempReason[i]);
                  }
                tempCovid = fileScanner.nextLine();
                String covid[] = tempCovid.split(",");
                for (int i = 0; i < covid.length; i++ ) {
                  attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
                  }
                tempReasonCovid = fileScanner.nextLine();
                String reasonCov[] = tempReasonCovid.split(",");
                for (int i = 0; i < reasonCov.length; i++ ) {
                  attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
                  }

                  dates = new ArrayList<String>();

                  tempDate = fileScanner.nextLine();
                String date[] = tempDate.split(",");
                for (int i = 0; i < date.length; i++ ) {
                  dates.add(date[i]);
                  }

                  guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
                  guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
                  guardianOnePhoneNumber = fileScanner.nextLine();
                  guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
                  guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
                  guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                  guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
                  guardianTwoPhoneNumber = fileScanner.nextLine();
                  guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
                  guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

                  emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
                  emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
                  emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
                  emergencyContactOneHomeNumber = (fileScanner.nextLine());
                  emergencyContactOneCellNumber = (fileScanner.nextLine());
                  emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                  emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
                  emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
                  emergencyContactTwoHomeNumber = (fileScanner.nextLine());
                  emergencyContactTwoCellNumber = (fileScanner.nextLine());

                  healthFactorOne = (fileScanner.nextLine()).toLowerCase();
                  healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                  healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                  healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                  healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                  healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                  healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                  healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                  healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                  healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                  healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                  healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

                Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,coun-
tryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail,
guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardianTwoCallAtWork, emergencyCon-
tactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCell-
Number,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyCon-
tactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired,
healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree,
healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                  listOfStudents.add(tempS);
                }
               fileScanner.close();

               return listOfStudents;
              }

          private static VerticalLayout createDialogLayout(Dialog dialog) {
                H2 headline = new H2("Enter Student Information");
                headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                        .set("font-size", "1.5em").set("font-weight", "bold");

                TextField firstNameField = new TextField("First Name");
                TextField lastNameField = new TextField("Last Name");
                VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                        lastNameField);
                fieldLayout.setSpacing(false);
                fieldLayout.setPadding(false);
                fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

                Button cancelButton = new Button("Cancel", e -> dialog.close());
                Button saveButton = new Button("Done", e -> {
                   int index = -2;
                    boolean found = false;
                   for (int i = 0; i < listOfStudents.size(); i++) {
```

```
                        if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.getValue().equals(listOf-
Students.get(i).getLastName())) {
                          index = i;
                          found = true;
                          store(index);
                                        dialog.close();
                          UI.getCurrent().navigate("menuBRecordsV");
                          UI.getCurrent().getPage().reload();
                          break;
                        }
                      }

                      if (found == false) {
                                Notification.show("Invalid name entered.",
                                  3000, Notification.Position.MIDDLE);
                      }


                });
                saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                        saveButton);
                buttonLayout
                        .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

                VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                        buttonLayout);
                dialogLayout.setPadding(false);
                dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
                dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

                return dialogLayout;
            }

                  public static void store(int index) {
                              PrintWriter pw = null;
                    try {
                       pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
                       pw.println(index);
                       pw.close();
                    } catch (FileNotFoundException e) {
                      System.err.print("couldn't open file for writing!");
                      System.exit(0);
                    }
                       }

          //access stored index of student in temp file
                    public static void index () {
                              index = -1;
                               try {
                                   fileScanner = new Scanner(new File("temp.txt"));
                                   index = Integer.parseInt(fileScanner.nextLine());

                                         fileScanner.close();
                                 } catch (FileNotFoundException e) {
                                   System.err.println("File not found! Choosing to quit now...");

                                   System.exit(0);
                                 }
                    }

}
```

CLASS: MenuCProgressR.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.Key;
import com.vaadin.flow.component.splitlayout.SplitLayout;
import com.vaadin.flow.component.textfield.IntegerField;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.dialog.Dialog;
import com.vaadin.flow.component.html.Footer;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H4;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.component.notification.Notification;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.orderedlayout.FlexComponent;
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;
import com.vaadin.flow.router.Route;

@Route(value = "menuCProgressR", layout = Welcome.class)
public class MenuCProgressR extends VerticalLayout {

          static Scanner fileScanner;
          static ArrayList <Student> listOfStudents = new ArrayList <Student>();
          static int index = -1;
```

```
public MenuCProgressR() {

        //read from files
                listOfStudents.removeAll(listOfStudents);
                listOfStudents = fileOneOpen();
                index();

                DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
            LocalDateTime firstNow = LocalDateTime.now();
           String alreadyDone = firstFormatter.format(firstNow);

                 //check if today's progress has been recorded or not
          if ((listOfStudents.get(index).getLastRecord().equals(alreadyDone))) {
                H1 done = new H1("Today's progress for " + listOfStudents.get(index).getFullName() + " has already been rec-
orded.");
            addClassName("centered-content");
            done.setWidth("500px");

            Button incomplete = new Button("Back", e -> {
                UI.getCurrent().navigate("menu");
            });

                incomplete.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                incomplete.setMinWidth("250px");
              incomplete.addClickShortcut(Key.ENTER);

              add(done, incomplete);

              setSizeFull();
              setJustifyContentMode(JustifyContentMode.CENTER);
              setDefaultHorizontalComponentAlignment(Alignment.CENTER);
              getStyle().set("text-align", "center");
          } else {

            //make lastrecorded date into current date
                        LocalDateTime firstNow1 = LocalDateTime.now(ZoneId.systemDefault());
                    String lastRecord1 = firstFormatter.format(firstNow1);
                    listOfStudents.get(index).setLastRecord(lastRecord1);


                    //IF NOT WORKING TRY ADDING HORIZNTAL FIELD TO VERTICAL FIELD AND ADDING VERTICAL LAYOIT TO ADD EVERY
TIME
                    //if program chosen is hafiz
                    if ((listOfStudents.get(index).getProgramChosen()).equals("hafiz")) {
                        VerticalLayout dour = new VerticalLayout();
                           H1 dourTitle = new H1("Dour");
                              //dourTitle.addClassName("centered-content");
                              //dourTitle.setWidth("500px");
                              dour.add(dourTitle);
                         Paragraph info = new Paragraph ("Dour: " + listOfStudents.get(index).getFullName()+" is on quarter
number " + listOfStudents.get(index).getCurrentQuarter() + " of sapara " + listOfStudents.get(index).getDourCurrentSapara() + ".");
                         Paragraph info2 = new Paragraph("Status: Hafiz");
                         dour.add(info, info2);
                         H4 dourComplete = new H4("Was dour completed today?");
                         //dourComplete.addClassName("centered-content");
                         dour.add(dourComplete);
                         HorizontalLayout buttons = new HorizontalLayout();
                        HorizontalLayout dours = new HorizontalLayout();
                         Button complete = new Button ("Yes", e -> {
                                  listOfStudents.get(index).setTodayDourDoneOrNot(true);
                                  HorizontalLayout temp = new HorizontalLayout();
                                  IntegerField numQuarters = new IntegerField ("Enter number of quarters done");
                                  numQuarters.setWidth("250px");
                                  Button done = new Button ("Done", l -> {

                                      listOfStudents.get(index).setTodayQuartersDone(numQuarters.getValue());
                                        int pastCurrentQuarter = listOfStudents.get(index).getCurrentQuarter();
                                        int newCurrentQuarter = (numQuarters.getValue()+(listOfStudents.get(in-
dex).getCurrentQuarter()))%4;

                                        listOfStudents.get(index).setCurrentQuarter(newCurrentQuarter);
                                        if ((pastCurrentQuarter + numQuarters.getValue()) > 4) {
                                                HorizontalLayout temp2 = new HorizontalLayout();
                                          IntegerField newDour = new IntegerField ("Enter new current dour sapara");
                                          newDour.setWidth("250px");
                                          Button done2 = new Button ("Done", k -> {

                                               H4 headline = new H4("Progress for today has been recorded.");
                                               dour.add(headline);

                                                    listOfStudents.get(index).setTodayDourSaparaDone(listOfStu-
dents.get(index).getDourCurrentSapara());

                                                    listOfStudents.get(index).setDourCurrentSapara(newDour.get-
Value());

                                                    listOfStudents.get(index).setTodayDourSaparaDoneOrNot(true);
                                          });
                                          temp2.add(newDour, done2);
                                          temp2.setAlignItems(Alignment.BASELINE);
                                          temp2.setSpacing(false);
                                          dours.add(temp2);

                                          closeFileOne(listOfStudents);
                                        } else {
                                                H4 headline = new H4("Progress for today has been recorded.");
                                               dour.add(headline);
                                                listOfStudents.get(index).setTodayDourSaparaDoneOrNot(false);
                                        }

                                  });
                                  temp.setAlignItems(Alignment.BASELINE);
```

```
                                        temp.add(numQuarters, done);
                                        temp.setSpacing(false);
                                        dours.add(temp);

                                    });

                                    Button incomplete = new Button("No", e-> {
                                        listOfStudents.get(index).setTodayDourDoneOrNot(false);
                                        H4 headline = new H4("Progress for today has been recorded.");
                                dour.add(headline);
                                    });
                                    buttons.add(complete, incomplete);
                                    dour.add(buttons);
                                    dour.add(dours);
                                    dour.setDefaultHorizontalComponentAlignment(Alignment.CENTER);
                                    add(dour);


                                //if the program chosen is not hafix
                        } else {

                                //Record DOUR
                                VerticalLayout dour = new VerticalLayout();
                                          H1 dourTitle = new H1("Dour");
                                             dour.add(dourTitle);
                                    Paragraph info = new Paragraph ("Dour Progress: " + listOfStudents.get(index).getFullName()+" is on
quarter number " + listOfStudents.get(index).getCurrentQuarter() + " of sapara " + listOfStudents.get(index).getDourCurrentSapara() + ".");
                                    Paragraph info2 = new Paragraph("Status: " + listOfStudents.get(index).getProgramChosen());
                                    dour.add(info, info2);
                                    H4 dourComplete = new H4("Was dour completed today?");
                                    //dourComplete.addClassName("centered-content");
                                    dour.add(dourComplete);
                                    HorizontalLayout buttons = new HorizontalLayout();
                                   HorizontalLayout dours = new HorizontalLayout();
                                    Button complete = new Button ("Yes", e -> {
                                        listOfStudents.get(index).setTodayDourDoneOrNot(true);
                                        HorizontalLayout temp = new HorizontalLayout();
                                        IntegerField numQuarters = new IntegerField ("Enter number of quarters done");
                                        numQuarters.setWidth("200px");
                                        Button done = new Button ("Done", l -> {

                                                listOfStudents.get(index).setTodayQuartersDone(numQuarters.getValue());
                                                  int pastCurrentQuarter = listOfStudents.get(index).getCurrentQuarter();
                                                  int newCurrentQuarter = (numQuarters.getValue()+(listOfStudents.get(in-
dex).getCurrentQuarter()))%4;

                                                  listOfStudents.get(index).setCurrentQuarter(newCurrentQuarter);
                                                  if ((pastCurrentQuarter + numQuarters.getValue()) > 4) {
                                                            HorizontalLayout temp2 = new HorizontalLayout();
                                                    IntegerField newDour = new IntegerField ("Enter new current dour sapara");
                                                    newDour.setWidth("200px");
                                                    Button done2 = new Button ("Done", k -> {
                                                          H4 headline = new H4("Progress for today has been recorded.");
                                                          dour.add(headline);
                                                    });
                                                    temp2.add(newDour, done2);
                                                    temp2.setAlignItems(Alignment.BASELINE);
                                                    temp2.setSpacing(false);
                                                    dours.add(temp2);

                                                    listOfStudents.get(index).setTodayDourSaparaDone(listOfStudents.get(in-
dex).getDourCurrentSapara());


                                                    listOfStudents.get(index).setDourCurrentSapara(newDour.getValue());

                                                    listOfStudents.get(index).setTodayDourSaparaDoneOrNot(true);
                                                  } else {
                                                            H4 headline = new H4("Dour progress for today has been recorded.");
                                                          dour.add(headline);
                                                            listOfStudents.get(index).setTodayDourSaparaDoneOrNot(false);
                                                  }

                                        });
                                        temp.setAlignItems(Alignment.BASELINE);
                                        temp.add(numQuarters, done);
                                        temp.setSpacing(false);
                                        dours.add(temp);

                                    });

                                    Button incomplete = new Button("No", e-> {
                                        listOfStudents.get(index).setTodayDourDoneOrNot(false);
                                        H4 headline = new H4("Dour progress for today has been recorded.");
                                dour.add(headline);
                                    });
                                    buttons.add(complete, incomplete);
                                    dour.add(buttons);
                                    dour.add(dours);
                                    dour.setDefaultHorizontalComponentAlignment(Alignment.CENTER);



                                    //Record SAPARA
                                    VerticalLayout sapara = new VerticalLayout();
                                                H1 saparaTitle = new H1("Sabaq");
                                                   sapara.add(saparaTitle);
                                               Paragraph info3 = new Paragraph ("Sabaq Progress: " + listOfStudents.get(index).getFirst-
Name()+" is memorizing sapara " + listOfStudents.get(index).getCurrentSaparaMemorizing());

                                                //insertion sort
                                                //store what is to be sorted into a string and get rid of all spaces
                                                String toSort = listOfStudents.get(index).getSaparasDone();
```

```
                toSort = toSort.replaceAll(" ", "");
                //split the string by commas and make into string array
                String strArray[] = toSort.split(",");
              //convert string array into integer array of the same length as string array
                int arrayOfToSort[] = new int [strArray.length];
                for (int i = 0; i < strArray.length; i++) {
                                arrayOfToSort [i] = Integer.parseInt(strArray[i]);
                               }
                //start with 1 instead of 0 as element at 0 is already sorted
                    for (int i = 1; i < arrayOfToSort.length; i++)
                    {
                        int curNumber = arrayOfToSort[i];
                     // Set index to be place to the left
                        int curIndex = i-1;

                       //go through unsorted part of the array and find lowest value
                        while ( curIndex >= 0 && arrayOfToSort[curIndex] > curNumber)
                        {
                          // Shift the value at curIndex to the right one place
                           arrayOfToSort[curIndex+1] = arrayOfToSort[curIndex];
                           curIndex--;
                        }
                     // Put this number in the proper location
                        arrayOfToSort[curIndex + 1] = curNumber;
                    }
             //turn sorted array into a string separated by commas
            String sorted = "";
                 for (int k = 0; k < arrayOfToSort.length; k++) {
                    if (k == 0) {
            sorted = sorted + arrayOfToSort [k];
                     } else {
                     sorted = sorted + ", " + arrayOfToSort [k];
                     }
                   }
                 //store the sorted string
                ListOfStudents.get(index).setSaparasDone(sorted);
                Paragraph info4 = new Paragraph("Saparas Memorized: " + ListOfStudents.get(in-
dex).getSaparasDone());

                sapara.add(info3, info4);

                H4 saparaComplete = new H4("Was sabaq completed today?");
                sapara.add(saparaComplete);
                HorizontalLayout buttons1 = new HorizontalLayout();

                Button complete1 = new Button ("Yes", f -> {

                          ListOfStudents.get(index).setTodaySabaqDoneOrNot(true);

                          IntegerField numLines = new IntegerField ("Number of lines memorized");
                          numLines.setWidth("250px");

                          IntegerField numMistakes = new IntegerField ("Number of mistakes made");
                          numMistakes.setWidth("250px");


                          HorizontalLayout tempd = new HorizontalLayout(numLines, numMistakes);

                          sapara.add(tempd);
                          H4 saparaCompleted = new H4("Was a sapara memorized today?");
                                  sapara.add(saparaCompleted);

                                  HorizontalLayout tempButtons = new HorizontalLayout();
                                  Button memorized = new Button ("Yes", c -> {

                                     ListOfStudents.get(index).setTodayLinesMemorized(numLines.get-
Value());
                                     ListOfStudents.get(index).setTodayMistakesMade(numMistakes.get-
Value());

                                      ListOfStudents.get(index).setTodaySaparaFinished(true);
                                int pastSapara = ListOfStudents.get(index).getCurrentSaparaMemorizing();
                                ListOfStudents.get(index).setTodaySaparaDone(pastSapara);
                                ListOfStudents.get(index).addSaparasDone(pastSapara);
                                HorizontalLayout temporary = new HorizontalLayout();
                                IntegerField newSaparaa = new IntegerField("Enter the new sapara that they
are starting");
                                newSaparaa.setWidth("300px");
                                Button donee = new Button("Done", k -> {

                                       int newSapara = newSaparaa.getValue();
                                                    ListOfStudents.get(index).setCurrentSaparaMemoriz-
ing(newSapara);

                                                    if(ListOfStudents.get(index).getTotalSaparasDone() >=
30) {
                                                      ListOfStudents.get(index).setProgramChosen("ha-
fiz");
                                                      Notification.show("This student is now a hafiz.
There program has now sucessfully changed. To change it back, go to 'Change type of student option.'",
                                                              3000, Notification.Position.MIDDLE);
                                                    }

                                                    H4 headline = new H4("Sapara Progress for today has
been recorded.");
                                                sapara.add(headline);
                                });
                                temporary.add(newSaparaa, donee);
                                temporary.setAlignItems(Alignment.BASELINE);

                                sapara.add(temporary);
```

```
                                                        });

                                                        Button notMemorized = new Button("No", l -> {
                                                           listOfStudents.get(index).setTodayLinesMemorized(numLines.get-
Value());
                                                           listOfStudents.get(index).setTodayMistakesMade(numMistakes.get-
Value());
                                                           listOfStudents.get(index).setTodaySaparaFinished(false);
                                                           H4 headline = new H4("Sapara Progress for today has been recorded.");
                                                    sapara.add(headline);

                                                        });
                                                        tempButtons.add(memorized, notMemorized);
                                                        sapara.add(tempButtons);
                                                 });

                                                 Button incomplete1 = new Button ("No", e -> {
                                                          listOfStudents.get(index).setTodaySabaqDoneOrNot(false);
                                                          H4 headline = new H4("Sapara progress for today has been recorded.");
                                                sapara.add(headline);
                                                 });

                                                 buttons1.add(complete1, incomplete1);
                                                 sapara.add(buttons1);
                                                 sapara.setDefaultHorizontalComponentAlignment(Alignment.CENTER);
                                                //make the split layout


                                                        SplitLayout splitLayout = new SplitLayout(dour, sapara);

                                                        //splitLayout.setMaxHeight("280px");
                                                        add(splitLayout);


                        }
                                    // Footer
                                                Button done = new Button("Done", e -> {
                                                       closeFileOne(listOfStudents);
                                                       UI.getCurrent().navigate("menu");
                                                });
                                                done.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                                                done.getStyle().set("margin-right", "var(--lumo-space-s)");

                                                Button anotherStudent = new Button("Another Student", l ->{
                                                       closeFileOne(listOfStudents);
                                                       Dialog dialog = new Dialog();
                                                       dialog.getElement().setAttribute("aria-label", "Enter Student Information");

                                                       VerticalLayout dialogLayout = createDialogLayout(dialog);
                                                       dialog.add(dialogLayout);
                                                       dialog.open();
                                                       add(dialog);
                                                });
                                                anotherStudent.addThemeVariants(ButtonVariant.LUMO_TERTIARY);

                                                Footer footer = new Footer(done, anotherStudent);
                                                footer.getStyle().set("padding", "var(--lumo-space-wide-m)");
                                                footer.addClassName("centered-content");
                                                add(footer);

                                                setAlignItems(Alignment.STRETCH);
                                                //setHeight("400px");
                                                //setMaxWidth("100%");
                                                setPadding(false);
                                                setSpacing(false);
                                                //setWidth("360px");
                                               // getStyle().set("border", "1px solid var(--lumo-contrast-20pct)");

                        }




        }

        //access stored index of student in temp file
                  public static void index() {
                            index = -1;
                             try {
                                 fileScanner = new Scanner(new File("temp.txt"));
                                 index = Integer.parseInt(fileScanner.nextLine());

                                        fileScanner.close();
                                   } catch (FileNotFoundException e) {
                                     System.err.println("File not found! Choosing to quit now...");

                                     System.exit(0);
                                   }
                  }
                   public static void closeFileOne(ArrayList <Student> listOfStudents) {
                               PrintWriter pw = null;
                                     try {
                                        pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                                     } catch (FileNotFoundException e) {
                                       System.err.print("couldn't open file for writing!");
                                       System.exit(0);
                                     }
```

```
            for (int y = 0; y < listOfStudents.size(); y++) {

              if (y == 0) {
             pw.println(listOfStudents.get(y).getFirstName());
              } else {
                pw.println(listOfStudents.get(y).getFirstName());
              }
              pw.println(listOfStudents.get(y).getMiddleName());
             pw.println(listOfStudents.get(y).getLastName());
             pw.println(listOfStudents.get(y).getAddress());
             pw.println(listOfStudents.get(y).getDateOfBirth());
             pw.println(listOfStudents.get(y).getAge());
             pw.println(listOfStudents.get(y).getPostalCode());
             pw.println(listOfStudents.get(y).getLanguage());
             pw.println(listOfStudents.get(y).getCountryOfBirth());
pw.println(listOfStudents.get(y).getProgramChosen());
              pw.println(listOfStudents.get(y).getLastRecord());


          String holder = "";
         for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
              if (k == 0) {
 holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
             } else {
             holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
             }
             }
               pw.println(holder);

               holder = "";
    for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
              if (k == 0) {
               holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
             } else {
             holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
             }
             }
               pw.println(holder);

   pw.println(listOfStudents.get(y).getCurrentQuarter());

   holder = "";
              for (int k = 0; k < listOfStudents.get(y).getNumOfDourSaparasDoneMonth().length; k++) {
              if (k == 0) {
               holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
             } else {
             holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
             }
             }
               pw.println(holder);
     pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStudents.get(y).getDourNext-
Fill());

     DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDateTime firstNow = LocalDateTime.now();
       String alreadyDone = firstFormatter.format(firstNow);

     if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
         pw.println(false);
         pw.println(-1);
         pw.println(false);
         pw.println(-1);
         pw.println(false);
         pw.println(-1);
         pw.println(-1);
         pw.println(false);
         pw.println(-1);
     } else {
          pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
          pw.println(listOfStudents.get(y).getTodayQuartersDone());
          pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
          pw.println(listOfStudents.get(y).getTodayDourSaparaDone());

       if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
              pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
              pw.println(listOfStudents.get(y).getTodayLinesMemorized());
              pw.println(listOfStudents.get(y).getTodayMistakesMade());
              pw.println(listOfStudents.get(y).isTodaySaparaFinished());
              pw.println(listOfStudents.get(y).getTodaySaparaDone());
       } else {
         pw.println(false);
         pw.println(-1);
         pw.println(-1);
         pw.println(false);
         pw.println(-1);
       }
     }



     if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
           holder = "";
       for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
           if (k == 0) {
            holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
           } else {
           holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
           }
           }
       pw.println(holder);
```

```
                holder = "";
               for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
                     if (k == 0) {
                      holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
                    }
                    }
               pw.println(holder);

               holder = "";
               for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
                     if (k == 0) {
                      holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
                    }
                    }
               pw.println(holder);

               holder = "";
                 for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
                     if (k == 0) {
                      holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
                    }
                    }
                 pw.println(holder);

                 holder = "";
                 for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++) {
                     if (k == 0) {
                      holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
                    }
                    }
                 pw.println(holder);
               pw.println(listOfStudents.get(y).getTotalSaparasDone());
                    pw.println(listOfStudents.get(y).getSaparasDone());
          pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStudents.get(y).getSaparaNext-
Fill());
               } else {
                 pw.println(false);
                 pw.println(0);
                 pw.println(0);
                 pw.println(false);
                 pw.println(0);
                 pw.println(0);
                 pw.println(0);
                 pw.println(0);
                 pw.println(0);
               }


               //attendance
              //printing to file for attendance
               holder = "";
                     for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

                     if (k == 0) {
                      holder = "" + (listOfStudents.get(y).getAttendance().get(k));

                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));

                    }
                       }
                     pw.println(holder);

                     holder = "";
                    //printing to file for reason absent
                    for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

                     if (d == 0) {
                      holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

                    }
                      }
                    pw.println(holder);
                    //printing to file for covid screening
                    holder = "";
                    for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

                     if (r == 0) {
                      holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

                    } else {
                    holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

                    }
                   }
                     pw.println(holder);

                   //printing to file for reason covid screening was not done
                     holder = "";
```

```
                                    for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++) {

                                     if (p == 0) {
                                      holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

                                    } else {
                                    holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreening().get(p));

                                    }
                                  }
                                    pw.println(holder);

                                  //printing to file for dates
                                    holder = "";
                                  for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

                                     if (z == 0) {
                                      holder = ""+(listOfStudents.get(y).getDate().get(z));

                                    } else {
                                    holder = holder + ("," + listOfStudents.get(y).getDate().get(z));

                                    }
                                  }
                                  pw.println(holder);
                                  pw.println(listOfStudents.get(y).getGuardianOneFirstName());
                                  pw.println(listOfStudents.get(y).getGuardianOneLastName());
                                  pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
                                  pw.println(listOfStudents.get(y).getGuardianOneEmail());
                                  pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
                                  pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
                                  pw.println(listOfStudents.get(y).getGuardianTwoLastName());
                                  pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
                                  pw.println(listOfStudents.get(y).getGuardianTwoEmail());
                                  pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

                                  pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
                                  pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
                                  pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
                                  pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
                                  pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
                                  pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
                                  pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
                                  pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
                                  pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
                                  pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

                                  pw.println(listOfStudents.get(y).getHealthFactorOne());
                                  pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
                                  pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
                                  pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
                                  pw.println(listOfStudents.get(y).getHealthFactorTwo());
                                  pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
                                  pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
                                  pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
                                  pw.println(listOfStudents.get(y).getHealthFactorThree());
                                  pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
                                  pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
                                  pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                                      }
                              pw.close();


                                      }


                    public static ArrayList <Student> fileOneOpen() {
                           try {
                            fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
                          } catch (FileNotFoundException e) {
                            System.err.println("File not found! Choosing to quit now...");

                            System.exit(0);
                          }

                          //programChosen  - CHECK CONSTRUCTORS
                          //add health factors to printing out in emergency situation stuff

                         String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance,
tempReasonAttendance, tempCovid, tempReasonCovid;

                          Boolean[] dourDoneOrNot;
                          Boolean todayDourDoneOrNot;
                          int[] quarterNumDoneMonth;
                          int todayQuartersDone, currentQuarter;
                          Boolean[] numOfDourSaparasDoneMonth;
                          Boolean todayDourSaparaDoneOrNot;
                          int todayDourSaparaDone;
                          int dourCurrentSapara, dourNextFill;

                          String programChosen;
                          String lastRecord;

                          Boolean[] sabaqDoneOrNot;
                          Boolean todaySabaqDoneOrNot;
                          int[] linesMemorized;
                          int todayLinesMemorized;
                          int[] mistakesMade;
                          int todayMistakesMade;
                          Boolean[] numOfSaparasDoneMonth;
```

```
                    Boolean todaySaparaFinished;
                    int[] nameOfSaparasDoneMonth;
                    int totalSaparasDone;
                    int todaySaparaDone;
                    String saparasDone;
                    int currentSaparaMemorizing;
                    int saparaNextFill = 0;

                   int age;

                   String tempDate;
                   ArrayList<String> dates;


                   String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
                   String guardianOneEmail;
                   Boolean guardianOneCallAtWork;
                   String guardianTwoFirstName, guardianTwoLastName;
                   String guardianTwoPhoneNumber;
                   String guardianTwoEmail;
                   Boolean guardianTwoCallAtWork;

                   String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
                   String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
                   String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
                   String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

                   String healthFactorOne;
                   Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
                   String healthFactorTwo;
                   Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
                   String healthFactorThree;
                   Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedica-
tionsRequired;

                   Attendance attendanceOfStudent;
                   StudentProgress progressOfStudent;

                   while (fileScanner.hasNextLine()) {
                          dourDoneOrNot = new Boolean[30];
                               quarterNumDoneMonth = new int[30];
                              numOfDourSaparasDoneMonth = new Boolean[30];
                              sabaqDoneOrNot = new Boolean[30];
                               linesMemorized = new int[30];
                               mistakesMade = new int[30];
                               numOfSaparasDoneMonth = new Boolean[30];
                              nameOfSaparasDoneMonth = new int[30];

                     firstName = (fileScanner.nextLine()).toLowerCase();
                     middleName = (fileScanner.nextLine()).toLowerCase();
                     lastName = (fileScanner.nextLine()).toLowerCase();
                     address = (fileScanner.nextLine()).toLowerCase();
                     dateOfBirth = fileScanner.nextLine();
                     age = Integer.parseInt(fileScanner.nextLine());
                     postalCode = (fileScanner.nextLine()).toLowerCase();
                     language = (fileScanner.nextLine()).toLowerCase();
                     countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                     //progress of student
                     programChosen = (fileScanner.nextLine()).toLowerCase();
                     progressOfStudent = new StudentProgress();
                     progressOfStudent.setProgramChosen(programChosen);

                     lastRecord = (fileScanner.nextLine());
                     progressOfStudent.setLastRecord(lastRecord);


                     String tempDourDoneOrNot = fileScanner.nextLine();
                     String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                     for (int i = 0; i < strDourDoneOrNot.length; i++) {
                       dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                     }
                progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                     String tempQuarterNumDoneMonth = fileScanner.nextLine();
                     String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                     for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                       quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                     }
                progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                     currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuarter(cur-
rentQuarter);

                     String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                     String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                     for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                       numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                     }
                progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                     dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dour-
CurrentSapara);

                     dourNextFill = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenDourNextFill(dourNextFill);

                     DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                        LocalDateTime firstNow = LocalDateTime.now();
                       String alreadyDone = firstFormatter.format(firstNow);
```

```
            if (!(alreadyDone.equals(lastRecord))) {
              if (programChosen.equals("hafiz")) {
                Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                int holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());

                todayDourDoneOrNot = false;
                todayDourSaparaDoneOrNot = false;
                todayQuartersDone = 0;
                todayDourSaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
          progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
          progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
          progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

              } else {
                Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                int holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                todayDourDoneOrNot = false;
                 todayQuartersDone = 0;
                 todayDourSaparaDoneOrNot = false;
                todayDourSaparaDone = 0;
                 todaySabaqDoneOrNot = false;
                todayLinesMemorized = 0;
                todayMistakesMade = 0;
                todaySaparaFinished = false;
                todaySaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
          progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
          progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
          progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
          progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
          progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
          progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
          progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
          progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
            }


            } else {
              todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
              progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

              todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
              progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

              todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
             progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

             todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
             progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

              if (!(programChosen.equals("hafiz"))) {

            todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOr-
Not(todaySabaqDoneOrNot);

                todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
          progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
          progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
          progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
          progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
              } else {
                Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                int holder = Integer.parseInt(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
              }
            }



            if (!(programChosen.equals("hafiz"))) {
              String tempSabaqDoneOrNot = fileScanner.nextLine();
            String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
            for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
              sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
            }
          progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

            String tempLinesMemorized = fileScanner.nextLine();
```

```
      String strLinesMemorized[] = tempLinesMemorized.split(",");
      for (int i = 0; i < strLinesMemorized.length; i++) {
        linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
      }
      progressOfStudent.setOpenLinesMemorized(linesMemorized);

      String tempMistakesMade = fileScanner.nextLine();
      String strMistakesMade[] = tempMistakesMade.split(",");
      for (int i = 0; i < strMistakesMade.length; i++) {
        mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
      }
      progressOfStudent.setOpenMistakesMade(mistakesMade);

      String tempNumOfSaparasFinished = fileScanner.nextLine();
      String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
      for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
       numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

      String tempNameOfSaparasFinished = fileScanner.nextLine();
      String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
      for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
       nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

      totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

      saparasDone = fileScanner.nextLine();
      progressOfStudent.setOpenSaparasDone(saparasDone);

      currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

      saparaNextFill = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
      } else {
        String hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
      }


      //attendance

      tempAttendance = fileScanner.nextLine();
      String attendance[] = tempAttendance.split(",");
      attendanceOfStudent = new Attendance();
      for (int i = 0; i < attendance.length; i++ ) {
        attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
        }
      tempReasonAttendance = fileScanner.nextLine();
      String tempReason[] = tempReasonAttendance.split(",");
      for (int i = 0; i < tempReason.length; i++ ) {
        attendanceOfStudent.addReasonAbsent(tempReason[i]);
        }
      tempCovid = fileScanner.nextLine();
      String covid[] = tempCovid.split(",");
      for (int i = 0; i < covid.length; i++ ) {
        attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
        }
      tempReasonCovid = fileScanner.nextLine();
      String reasonCov[] = tempReasonCovid.split(",");
      for (int i = 0; i < reasonCov.length; i++ ) {
        attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
        }

        dates = new ArrayList<String>();

        tempDate = fileScanner.nextLine();
      String date[] = tempDate.split(",");
      for (int i = 0; i < date.length; i++ ) {
        dates.add(date[i]);
        }

        guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
        guardianOnePhoneNumber = fileScanner.nextLine();
        guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
        guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
        guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoPhoneNumber = fileScanner.nextLine();
        guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
        guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

        emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneHomeNumber = (fileScanner.nextLine());
        emergencyContactOneCellNumber = (fileScanner.nextLine());
        emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
```

```
                emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
                emergencyContactTwoHomeNumber = (fileScanner.nextLine());
                emergencyContactTwoCellNumber = (fileScanner.nextLine());

                healthFactorOne = (fileScanner.nextLine()).toLowerCase();
                healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

            Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, lan-
guage,countryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardia-
nOneEmail, guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardianTwoCallAtWork,
emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContac-
tOneCellNumber,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emer-
gencyContactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorOneMedica-
tionsRequired, healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, health-
FactorThree, healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
               listOfStudents.add(tempS);
            }
            fileScanner.close();

            return listOfStudents;
           }

               //dialog for entering another student's information
               private static VerticalLayout createDialogLayout(Dialog dialog) {
            H2 headline = new H2("Enter Student Information");
            headline.getStyle().set("margin", "var(--lumo-space-m) 0 0 0")
                    .set("font-size", "1.5em").set("font-weight", "bold");

            TextField firstNameField = new TextField("First Name");
            TextField lastNameField = new TextField("Last Name");
            VerticalLayout fieldLayout = new VerticalLayout(firstNameField,
                    lastNameField);
            fieldLayout.setSpacing(false);
            fieldLayout.setPadding(false);
            fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

            Button cancelButton = new Button("Cancel", e -> dialog.close());
            Button saveButton = new Button("Done", e -> {
               int index = -2;
                boolean found = false;
               for (int i = 0; i < listOfStudents.size(); i++) {
                    if (firstNameField.getValue().equals(listOfStudents.get(i).getFirstName()) && lastNameField.get-
Value().equals(listOfStudents.get(i).getLastName())) {
                      index = i;
                      found = true;
                      store(index);
                                  dialog.close();
                      UI.getCurrent().navigate("menuCProgressR");
                      UI.getCurrent().getPage().reload();
                      break;
                    }
                }

                if (found == false) {
                          Notification.show("Invalid name entered.",
                            3000, Notification.Position.MIDDLE);
                }


            });
            saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
            HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
                    saveButton);
            buttonLayout
                    .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

            VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
                    buttonLayout);
            dialogLayout.setPadding(false);
            dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
            dialogLayout.getStyle().set("width", "300px").set("max-width", "100%");

            return dialogLayout;
        }



            //store student index to temp file
            public static void store(int index) {
                        PrintWriter pw = null;
            try {
               pw = new PrintWriter(new File("../marchbreakia/temp.txt"));
               pw.println(index);
               pw.close();
            } catch (FileNotFoundException e) {
              System.err.print("couldn't open file for writing!");
              System.exit(0);
            }
               }
}
```

CLASS: MenuCProgressD.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.Key;
import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.html.H1;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H3;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.component.html.Section;
import com.vaadin.flow.component.orderedlayout.Scroller;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.Route;

@Route(value = "menuCProgressD", layout = Welcome.class)
public class MenuCProgressD extends VerticalLayout {

          static Scanner fileScanner;
          static ArrayList <Student> listOfStudents = new ArrayList <Student>();
          static int index = -1;

          public MenuCProgressD() {

          //read from files
          listOfStudents.removeAll(listOfStudents);
          listOfStudents = fileOneOpen();
          index();
          DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
    LocalDateTime firstNow = LocalDateTime.now();
   String alreadyDone = firstFormatter.format(firstNow);

          //check if today's progress has been recorded or not
    if (!(listOfStudents.get(index).getLastRecord().equals(alreadyDone))) {
             H1 done = new H1("Today's progress for " + listOfStudents.get(index).getFullName() + " is incomplete.");
      addClassName("centered-content");
      done.setWidth("500px");

      Button incomplete = new Button("Back", e -> {
             UI.getCurrent().navigate("menu");
      });

          incomplete.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
          incomplete.setMinWidth("250px");
        incomplete.addClickShortcut(Key.ENTER);

        add(done, incomplete);

        setSizeFull();
        setJustifyContentMode(JustifyContentMode.CENTER);
        setDefaultHorizontalComponentAlignment(Alignment.CENTER);
        getStyle().set("text-align", "center");
    } else {
          H2 intro = new H2 ("Daily Progress for " + listOfStudents.get(index).getFullName());

            H3 sabaqTitle = new H3("Sabaq Progress");
                        Section sabaq = new Section(sabaqTitle);

                      H3 dourTitle = new H3("Dour Progress");
                        Section dour = new Section(dourTitle);


                      if (!((listOfStudents.get(index).getProgramChosen()).equals("hafiz"))) {
    if (listOfStudents.get(index).getTodaySabaqDoneOrNot() == true) {
          Paragraph one = new Paragraph("Lines Memorized: " + listOfStudents.get(index).getTodayLinesMemorized());
          one.setWidthFull();
          sabaq.add(one);
          Paragraph two = new Paragraph("Mistakes Made: " + listOfStudents.get(index).getTodayMistakesMade());
          two.setWidthFull();
          sabaq.add(two);

      if (listOfStudents.get(index).isTodaySaparaFinished() == true) {
        Paragraph three = new Paragraph("Sapara Finished: " + listOfStudents.get(index).getTodaySaparaDone());
        three.setWidthFull();
        Paragraph four = new Paragraph("Total Number of Saparas Finished: " + listOfStudents.get(index).getTotalSaparasDone());
        four.setWidthFull();
        Paragraph five = new Paragraph ("Current Sapara Memorizing: " + listOfStudents.get(index).getCurrentSaparaMemorizing());
        five.setWidthFull();
        sabaq.add(three, four, five);
      } else {
        Paragraph six = new Paragraph ("Sapara Finished: None");
      six.setWidthFull();
      sabaq.add(six);
      }
    } else {
      Paragraph seven = new Paragraph("Sabaq was not done today.");
    seven.setWidthFull();
    sabaq.add(seven);
    }

                    }else {
```

```
                    Paragraph idek = new Paragraph ("Sabaq is not applicable for this student.");
                    idek.setWidthFull();
                    sabaq.add(idek);
            }

    if (listOfStudents.get(index).isTodayDourDoneOrNot() == true) {
      Paragraph eight = new Paragraph("Quarters Done Today: " + listOfStudents.get(index).getTodayQuartersDone());
      eight.setWidthFull();
      Paragraph nine = new Paragraph("Current Quarter: " + listOfStudents.get(index).getCurrentQuarter());
      nine.setWidthFull();
      dour.add(eight, nine);
      if (listOfStudents.get(index).isTodayDourSaparaDoneOrNot() == true) {
        Paragraph ten = new Paragraph ("The " + listOfStudents.get(index).getTodayDourSaparaDone() + " was finished today.");
        ten.setWidthFull();
        Paragraph eleven = new Paragraph("New Current Dour Sapara: " + listOfStudents.get(index).getDourCurrentSapara());
      eleven.setWidthFull();
      dour.add(ten, eleven);
      } else {
        Paragraph twelve = new Paragraph ("No new sapara was finished in dour today.");
      twelve.setWidthFull();
      dour.add(twelve);
      }

    } else {
      Paragraph thirteen = new Paragraph( "Dour was not done today.");
      thirteen.setWidthFull();
      dour.add(thirteen);
    }


           Button incomplete = new Button("Back", e -> {
            UI.getCurrent().navigate("menu");
     });

          incomplete.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
          incomplete.setMinWidth("250px");
       incomplete.addClickShortcut(Key.ENTER);

       setSizeFull();
       setJustifyContentMode(JustifyContentMode.CENTER);
       setDefaultHorizontalComponentAlignment(Alignment.CENTER);
       getStyle().set("text-align", "center");
       Scroller scroller = new Scroller(new Div(sabaq, dour));
               scroller.setScrollDirection(Scroller.ScrollDirection.VERTICAL);
               scroller.getStyle()
                       .set("border-bottom", "1px solid var(--lumo-contrast-20pct)")
                       .set("padding", "var(--lumo-space-m)");

          add(intro, scroller, incomplete);
    }
          }


           public static ArrayList <Student> fileOneOpen() {
                  try {
                    fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
                  } catch (FileNotFoundException e) {
                    System.err.println("File not found! Choosing to quit now...");

                    System.exit(0);
                  }

                  //programChosen  - CHECK CONSTRUCTORS
                  //add health factors to printing out in emergency situation stuff

                  String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tem-
pReasonAttendance, tempCovid, tempReasonCovid;

                   Boolean[] dourDoneOrNot;
                   Boolean todayDourDoneOrNot;
                   int[] quarterNumDoneMonth;
                   int todayQuartersDone, currentQuarter;
                   Boolean[] numOfDourSaparasDoneMonth;
                   Boolean todayDourSaparaDoneOrNot;
                   int todayDourSaparaDone;
                   int dourCurrentSapara, dourNextFill;

                   String programChosen;
                   String lastRecord;

                   Boolean[] sabaqDoneOrNot;
                   Boolean todaySabaqDoneOrNot;
                   int[] linesMemorized;
                   int todayLinesMemorized;
                   int[] mistakesMade;
                   int todayMistakesMade;
                   Boolean[] numOfSaparasDoneMonth;
                   Boolean todaySaparaFinished;
                   int[] nameOfSaparasDoneMonth;
                   int totalSaparasDone;
                   int todaySaparaDone;
                   String saparasDone;
                   int currentSaparaMemorizing;
                   int saparaNextFill = 0;

                  int age;

                  String tempDate;
                  ArrayList<String> dates;
```

```
                String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
                String guardianOneEmail;
                Boolean guardianOneCallAtWork;
                String guardianTwoFirstName, guardianTwoLastName;
                String guardianTwoPhoneNumber;
                String guardianTwoEmail;
                Boolean guardianTwoCallAtWork;

                String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
                String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
                String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
                String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

                String healthFactorOne;
                Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
                String healthFactorTwo;
                Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
                String healthFactorThree;
                Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

                Attendance attendanceOfStudent;
                StudentProgress progressOfStudent;

                while (fileScanner.hasNextLine()) {
                       dourDoneOrNot = new Boolean[30];
                            quarterNumDoneMonth = new int[30];
                           numOfDourSaparasDoneMonth = new Boolean[30];
                           sabaqDoneOrNot = new Boolean[30];
                            linesMemorized = new int[30];
                            mistakesMade = new int[30];
                            numOfSaparasDoneMonth = new Boolean[30];
                           nameOfSaparasDoneMonth = new int[30];

                  firstName = (fileScanner.nextLine()).toLowerCase();
                  middleName = (fileScanner.nextLine()).toLowerCase();
                  lastName = (fileScanner.nextLine()).toLowerCase();
                  address = (fileScanner.nextLine()).toLowerCase();
                  dateOfBirth = fileScanner.nextLine();
                  age = Integer.parseInt(fileScanner.nextLine());
                  postalCode = (fileScanner.nextLine()).toLowerCase();
                  language = (fileScanner.nextLine()).toLowerCase();
                  countryOfBirth = (fileScanner.nextLine()).toLowerCase();

                  //progress of student
                  programChosen = (fileScanner.nextLine()).toLowerCase();
                  progressOfStudent = new StudentProgress();
                  progressOfStudent.setProgramChosen(programChosen);

                  lastRecord = (fileScanner.nextLine());
                  progressOfStudent.setLastRecord(lastRecord);


                  String tempDourDoneOrNot = fileScanner.nextLine();
                  String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
                  for (int i = 0; i < strDourDoneOrNot.length; i++) {
                    dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
                  }
            progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

                  String tempQuarterNumDoneMonth = fileScanner.nextLine();
                  String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
                  for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                    quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
                  }
            progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

                  currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuarter(currentQuarter);

                  String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
                  String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
                  for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                    numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
                  }
            progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

                  dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCur-
rentSapara);

                  dourNextFill = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenDourNextFill(dourNextFill);

                  DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                     LocalDateTime firstNow = LocalDateTime.now();
                    String alreadyDone = firstFormatter.format(firstNow);

                  if (!(alreadyDone.equals(lastRecord))) {
                    if (programChosen.equals("hafiz")) {
                      Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      int holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                       temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());
                      temporary = Boolean.parseBoolean(fileScanner.nextLine());
                      holder = Integer.parseInt(fileScanner.nextLine());

                      todayDourDoneOrNot = false;
                      todayDourSaparaDoneOrNot = false;
                      todayQuartersDone = 0;
                      todayDourSaparaDone = 0;
```

```
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

              } else {
                Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                int holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                todayDourDoneOrNot = false;
                 todayQuartersDone = 0;
                 todayDourSaparaDoneOrNot = false;
                todayDourSaparaDone = 0;
                 todaySabaqDoneOrNot = false;
                todayLinesMemorized = 0;
                todayMistakesMade = 0;
                todaySaparaFinished = false;
                todaySaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
            progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
            progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
            progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
            progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
            progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                }


                } else {
                  todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                  progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                  todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                  progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                  todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                 progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

                 todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                 progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                  if (!(programChosen.equals("hafiz"))) {

                todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(to-
daySabaqDoneOrNot);

                  todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                  todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                  todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
            progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                  todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
            progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                } else {
                  Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                  int holder = Integer.parseInt(fileScanner.nextLine());
                  holder = Integer.parseInt(fileScanner.nextLine());
                  temporary = Boolean.parseBoolean(fileScanner.nextLine());
                  holder = Integer.parseInt(fileScanner.nextLine());
                }
              }



              if (!(programChosen.equals("hafiz"))) {
                String tempSabaqDoneOrNot = fileScanner.nextLine();
              String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
              for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
                sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
              }
            progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

              String tempLinesMemorized = fileScanner.nextLine();
              String strLinesMemorized[] = tempLinesMemorized.split(",");
              for (int i = 0; i < strLinesMemorized.length; i++) {
                linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
              }
              progressOfStudent.setOpenLinesMemorized(linesMemorized);

              String tempMistakesMade = fileScanner.nextLine();
              String strMistakesMade[] = tempMistakesMade.split(",");
              for (int i = 0; i < strMistakesMade.length; i++) {
                mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
              }
              progressOfStudent.setOpenMistakesMade(mistakesMade);

              String tempNumOfSaparasFinished = fileScanner.nextLine();
              String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
              for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
```

```
        numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

      String tempNameOfSaparasFinished = fileScanner.nextLine();
      String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
      for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
        nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
      }
      progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

      totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

      saparasDone = fileScanner.nextLine();
      progressOfStudent.setOpenSaparasDone(saparasDone);

      currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

      saparaNextFill = Integer.parseInt(fileScanner.nextLine());
      progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
      } else {
        String hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
        hold = fileScanner.nextLine();
      }


      //attendance

      tempAttendance = fileScanner.nextLine();
      String attendance[] = tempAttendance.split(",");
      attendanceOfStudent = new Attendance();
      for (int i = 0; i < attendance.length; i++ ) {
        attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
        }
      tempReasonAttendance = fileScanner.nextLine();
      String tempReason[] = tempReasonAttendance.split(",");
      for (int i = 0; i < tempReason.length; i++ ) {
        attendanceOfStudent.addReasonAbsent(tempReason[i]);
        }
      tempCovid = fileScanner.nextLine();
      String covid[] = tempCovid.split(",");
      for (int i = 0; i < covid.length; i++ ) {
        attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
        }
      tempReasonCovid = fileScanner.nextLine();
      String reasonCov[] = tempReasonCovid.split(",");
      for (int i = 0; i < reasonCov.length; i++ ) {
        attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
        }

        dates = new ArrayList<String>();

        tempDate = fileScanner.nextLine();
      String date[] = tempDate.split(",");
      for (int i = 0; i < date.length; i++ ) {
        dates.add(date[i]);
        }

        guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
        guardianOnePhoneNumber = fileScanner.nextLine();
        guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
        guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
        guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
        guardianTwoPhoneNumber = fileScanner.nextLine();
        guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
        guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

        emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
        emergencyContactOneHomeNumber = (fileScanner.nextLine());
        emergencyContactOneCellNumber = (fileScanner.nextLine());
        emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
        emergencyContactTwoHomeNumber = (fileScanner.nextLine());
        emergencyContactTwoCellNumber = (fileScanner.nextLine());

        healthFactorOne = (fileScanner.nextLine()).toLowerCase();
        healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
        healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorThree = (fileScanner.nextLine()).toLowerCase();
        healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
        healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
```

```
                    Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,coun-
tryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail,
guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardianTwoCallAtWork, emergencyCon-
tactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCell-
Number,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyCon-
tactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired,
healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree,
healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                      listOfStudents.add(tempS);
                    }
                    fileScanner.close();

                    return listOfStudents;
                   }

           //access stored index of student in temp file
                                public static void index() {
                                          index = -1;
                                           try {
                                               fileScanner = new Scanner(new File("temp.txt"));
                                               index = Integer.parseInt(fileScanner.nextLine());

                                                     fileScanner.close();
                                             } catch (FileNotFoundException e) {
                                               System.err.println("File not found! Choosing to quit now...");

                                               System.exit(0);
                                             }
                                }
}
```

CLASS: MenuCProgressM.java

```
package com.example.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.Key;
import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.html.Div;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.H3;
import com.vaadin.flow.component.html.Header;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.component.html.Section;
import com.vaadin.flow.component.orderedlayout.Scroller;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.router.Route;

@Route(value = "menuCProgressM", layout = Welcome.class)
public class MenuCProgressM extends VerticalLayout {

           static Scanner fileScanner;
           static ArrayList <Student> listOfStudents = new ArrayList <Student>();
           static int index = -1;

           public MenuCProgressM () {

                     //read from files
                     listOfStudents.removeAll(listOfStudents);
                     listOfStudents = fileOneOpen();
                     index();

                     //make the header
                      // Header
                 Header header = new Header();
                 header.getStyle()
                         .set("align-items", "center")
                         .set("border-bottom", "1px solid var(--lumo-contrast-20pct)")
                         .set("display", "flex")
                         .set("padding", "var(--lumo-space-m)");

                     H2 intro = new H2 ("Monthly Progress for " + listOfStudents.get(index).getFullName());
                     intro.getStyle().set("margin", "0");

                     header.add(intro);
                     add(header);

                if ((listOfStudents.get(index).getSabaqDoneOrNot()[29] == null)|| ( listOfStudents.get(index).getDourDoneOrNot() [29] ==
null)||(listOfStudents.get(index).getLastRecord().equals("11/11/2011") || (listOfStudents.get(index).getDourNextFill() < 29))) {
                    Paragraph send = new Paragraph("Sorry, not enough information to display monthly progress. Check back after 30 days.");
                    send.setWidthFull();
                    Scroller scroller = new Scroller(new Div(send));
                           scroller.setScrollDirection(Scroller.ScrollDirection.VERTICAL);
                           scroller.getStyle()
                                   .set("border-bottom", "1px solid var(--lumo-contrast-20pct)")
                                   .set("padding", "var(--lumo-space-m)");
                           add(scroller);

                                 Button incomplete = new Button("Back", e -> {
                                   UI.getCurrent().navigate("menu");
                            });

                                 incomplete.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
```

```
            incomplete.setMinWidth("250px");
          incomplete.addClickShortcut(Key.ENTER);

          setSizeFull();
          setJustifyContentMode(JustifyContentMode.CENTER);
          setDefaultHorizontalComponentAlignment(Alignment.CENTER);
          getStyle().set("text-align", "center");

            add(intro, scroller, incomplete);
} else {

      H3 sabaqTitle = new H3("Sabaq Progress");
        Section sabaq = new Section(sabaqTitle);

        H3 dourTitle = new H3("Dour Progress");
        Section dour = new Section(dourTitle);

  int numTimesSaparaNotDone;
  int averageLineMemorized;
  int averageMistakesMade;
  int numSaparasDone;
  String nameSaparasDone;
  int numTimesDourNotDone;
  int averageNumQuartersDone;
  int numDourSaparasDone;

  //use methods to calculate the monthly progress of student
  numTimesSaparaNotDone = timesNotDone(listOfStudents.get(index).getSabaqDoneOrNot());
  numTimesDourNotDone = timesNotDone(listOfStudents.get(index).getDourDoneOrNot());
  averageLineMemorized = findAverage(listOfStudents.get(index).getLinesMemorized());
  averageMistakesMade = findAverage(listOfStudents.get(index).getMistakesMade());
  averageNumQuartersDone = findAverage(listOfStudents.get(index).getQuarterNumDoneMonth());
  numSaparasDone = timesDone(listOfStudents.get(index).getNumOfSaparasDoneMonth());
  numDourSaparasDone = timesDone(listOfStudents.get(index).getNumOfDourSaparasDoneMonth());
  nameSaparasDone = findName(listOfStudents.get(index).getNameOfSaparasDoneMonth());

  //display the calculated progress in paragraphs
  if (!(listOfStudents.get(index).getProgramChosen().equals("hafiz"))) {
    Paragraph one = new Paragraph ("Total Number Of Times Sapara Not Done: " + numTimesSaparaNotDone);
    one.setWidthFull();
    Paragraph two = new Paragraph("Average Number Of Lines Memorized Per Day: " + averageLineMemorized);
    two.setWidthFull();
    Paragraph three = new Paragraph ("Average Mistakes Made Per Day: " + averageMistakesMade);
    three.setWidthFull();
    Paragraph four = new Paragraph ("Total Number Of Saparas Done: " + numSaparasDone);
    four.setWidthFull();

    //SEQUENTIAL SORT
    //turn string that is to be sorted into an integer array called temp
    String temp2 = nameSaparasDone;
    temp2 = temp2.replaceAll(" ", "");
    String strTemp[] = temp2.split(",");
    int temp[] = new int [strTemp.length];
    for (int i = 0; i < strTemp.length; i++) {
                      temp [i] = Integer.parseInt(strTemp[i]);
                    }
    //traverse through the temp array until it is sorted
    for (int index = 0; index < temp.length-1; index++) {
     //pick the current index.
      int minIndex = index;
    //find minimum in the rest of the array
      for (int i = index; i < temp.length; i++) {
        if (temp [i] < temp[minIndex]) {
          minIndex = i;
        }
      }
    //swap to put the minimum in current position.
      int tempValue = temp [index];
      temp [index] = temp [minIndex];
      temp [minIndex] = tempValue;
    //value at current index is sorted
     //repeated for the rest of the array
    }
    //make sorted array into a string and print
String print = "";
     for (int k = 0; k < temp.length; k++) {
           if (k == 0) {
print = print + temp [k];
          } else {
          print = print + ", " + temp [k];
          }
       }
    nameSaparasDone = print;
    Paragraph five = new Paragraph ("Saparas That Were Done: " + nameSaparasDone);
    five.setWidthFull();
    sabaq.add(one, two, three, four, five);
  } else {
    Paragraph one = new Paragraph("Sabaq not applicable to this student.");
    one.setWidthFull();
    sabaq.add(one);
  }

  Paragraph onee = new Paragraph("Total Number Of Times Dour Not Done: " + numTimesDourNotDone);
  onee.setWidthFull();
  Paragraph twoo = new Paragraph("Average Number of Dour Quarters Done Per Day: " + averageNumQuartersDone);
  twoo.setWidthFull();
  Paragraph three = new Paragraph("Number Of Saparas Done In Dour: " + numDourSaparasDone);
  three.setWidthFull();
  dour.add(onee,twoo,three);

  Scroller scroller = new Scroller(new Div(sabaq, dour));
```

```
                scroller.setScrollDirection(Scroller.ScrollDirection.VERTICAL);
                scroller.getStyle()
                        .set("border-bottom", "1px solid var(--lumo-contrast-20pct)")
                        .set("padding", "var(--lumo-space-m)");

                      Button incomplete = new Button("Back", e -> {
                        UI.getCurrent().navigate("menu");
                 });

                      incomplete.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
                      incomplete.setMinWidth("250px");
                   incomplete.addClickShortcut(Key.ENTER);

                   setSizeFull();
                   setJustifyContentMode(JustifyContentMode.CENTER);
                   setDefaultHorizontalComponentAlignment(Alignment.CENTER);
                   getStyle().set("text-align", "center");

                       add(intro, scroller, incomplete);
            }



        }


         public static ArrayList <Student> fileOneOpen() {
                try {
                 fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
                } catch (FileNotFoundException e) {
                 System.err.println("File not found! Choosing to quit now...");

                 System.exit(0);
                }

                //programChosen  - CHECK CONSTRUCTORS
                //add health factors to printing out in emergency situation stuff

                String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tem-
pReasonAttendance, tempCovid, tempReasonCovid;

                 Boolean[] dourDoneOrNot;
                 Boolean todayDourDoneOrNot;
                 int[] quarterNumDoneMonth;
                 int todayQuartersDone, currentQuarter;
                 Boolean[] numOfDourSaparasDoneMonth;
                 Boolean todayDourSaparaDoneOrNot;
                 int todayDourSaparaDone;
                 int dourCurrentSapara, dourNextFill;

                 String programChosen;
                 String lastRecord;

                 Boolean[] sabaqDoneOrNot;
                 Boolean todaySabaqDoneOrNot;
                 int[] linesMemorized;
                 int todayLinesMemorized;
                 int[] mistakesMade;
                 int todayMistakesMade;
                 Boolean[] numOfSaparasDoneMonth;
                 Boolean todaySaparaFinished;
                 int[] nameOfSaparasDoneMonth;
                 int totalSaparasDone;
                 int todaySaparaDone;
                 String saparasDone;
                 int currentSaparaMemorizing;
                 int saparaNextFill = 0;

                int age;

                String tempDate;
                ArrayList<String> dates;


                String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
                String guardianOneEmail;
                Boolean guardianOneCallAtWork;
                String guardianTwoFirstName, guardianTwoLastName;
                String guardianTwoPhoneNumber;
                String guardianTwoEmail;
                Boolean guardianTwoCallAtWork;

                String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
                String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
                String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
                String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

                String healthFactorOne;
                Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
                String healthFactorTwo;
                Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
                String healthFactorThree;
                Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

                Attendance attendanceOfStudent;
                StudentProgress progressOfStudent;

                while (fileScanner.hasNextLine()) {
                       dourDoneOrNot = new Boolean[30];
                            quarterNumDoneMonth = new int[30];
```

```
                    numOfDourSaparasDoneMonth = new Boolean[30];
                    sabaqDoneOrNot = new Boolean[30];
                     linesMemorized = new int[30];
                     mistakesMade = new int[30];
                     numOfSaparasDoneMonth = new Boolean[30];
                    nameOfSaparasDoneMonth = new int[30];

               firstName = (fileScanner.nextLine()).toLowerCase();
               middleName = (fileScanner.nextLine()).toLowerCase();
               lastName = (fileScanner.nextLine()).toLowerCase();
               address = (fileScanner.nextLine()).toLowerCase();
               dateOfBirth = fileScanner.nextLine();
               age = Integer.parseInt(fileScanner.nextLine());
               postalCode = (fileScanner.nextLine()).toLowerCase();
               language = (fileScanner.nextLine()).toLowerCase();
               countryOfBirth = (fileScanner.nextLine()).toLowerCase();

               //progress of student
               programChosen = (fileScanner.nextLine()).toLowerCase();
               progressOfStudent = new StudentProgress();
               progressOfStudent.setProgramChosen(programChosen);

               lastRecord = (fileScanner.nextLine());
               progressOfStudent.setLastRecord(lastRecord);


               String tempDourDoneOrNot = fileScanner.nextLine();
               String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
               for (int i = 0; i < strDourDoneOrNot.length; i++) {
                 dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
               }
          progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

               String tempQuarterNumDoneMonth = fileScanner.nextLine();
               String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
               for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
                 quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
               }
          progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

               currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuarter(currentQuarter);

               String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
               String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
               for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
                 numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
               }
          progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

               dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCur-
rentSapara);

               dourNextFill = Integer.parseInt(fileScanner.nextLine());
          progressOfStudent.setOpenDourNextFill(dourNextFill);

               DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
                  LocalDateTime firstNow = LocalDateTime.now();
                 String alreadyDone = firstFormatter.format(firstNow);

               if (!(alreadyDone.equals(lastRecord))) {
                 if (programChosen.equals("hafiz")) {
                   Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   int holder = Integer.parseInt(fileScanner.nextLine());
                   temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());
                   temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());

                   todayDourDoneOrNot = false;
                   todayDourSaparaDoneOrNot = false;
                   todayQuartersDone = 0;
                   todayDourSaparaDone = 0;
                   progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
          progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
          progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
          progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                 } else {
                   Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   int holder = Integer.parseInt(fileScanner.nextLine());
                   temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());
                    temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());
                   temporary = Boolean.parseBoolean(fileScanner.nextLine());
                   holder = Integer.parseInt(fileScanner.nextLine());
                   todayDourDoneOrNot = false;
                    todayQuartersDone = 0;
                    todayDourSaparaDoneOrNot = false;
                   todayDourSaparaDone = 0;
                    todaySabaqDoneOrNot = false;
                   todayLinesMemorized = 0;
                   todayMistakesMade = 0;
                   todaySaparaFinished = false;
                   todaySaparaDone = 0;
                   progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
          progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
```

```
                progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
                progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
                progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
                progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
                progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
                progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
                progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                     }


                     } else {
                       todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                       progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                       todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                       progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                       todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                      progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

                      todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                      progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                       if (!(programChosen.equals("hafiz"))) {

                     todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(to-
daySabaqDoneOrNot);

                       todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                       todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                       todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
                progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                       todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                     } else {
                       Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                       int holder = Integer.parseInt(fileScanner.nextLine());
                       holder = Integer.parseInt(fileScanner.nextLine());
                       temporary = Boolean.parseBoolean(fileScanner.nextLine());
                       holder = Integer.parseInt(fileScanner.nextLine());
                     }
                    }



                    if (!(programChosen.equals("hafiz"))) {
                      String tempSabaqDoneOrNot = fileScanner.nextLine();
                    String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
                    for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
                      sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
                    }
                progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

                    String tempLinesMemorized = fileScanner.nextLine();
                    String strLinesMemorized[] = tempLinesMemorized.split(",");
                    for (int i = 0; i < strLinesMemorized.length; i++) {
                      linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
                    }
                    progressOfStudent.setOpenLinesMemorized(linesMemorized);

                    String tempMistakesMade = fileScanner.nextLine();
                    String strMistakesMade[] = tempMistakesMade.split(",");
                    for (int i = 0; i < strMistakesMade.length; i++) {
                      mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
                    }
                    progressOfStudent.setOpenMistakesMade(mistakesMade);

                    String tempNumOfSaparasFinished = fileScanner.nextLine();
                    String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
                    for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
                     numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
                    }
                    progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

                    String tempNameOfSaparasFinished = fileScanner.nextLine();
                    String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
                    for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
                     nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
                    }
                    progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

                    totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
                    progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

                    saparasDone = fileScanner.nextLine();
                    progressOfStudent.setOpenSaparasDone(saparasDone);

                    currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

                    saparaNextFill = Integer.parseInt(fileScanner.nextLine());
                    progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
                    } else {
                      String hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
                      hold = fileScanner.nextLine();
```

```
                    hold = fileScanner.nextLine();
                    hold = fileScanner.nextLine();
                    hold = fileScanner.nextLine();
                    hold = fileScanner.nextLine();
                    hold = fileScanner.nextLine();
                    hold = fileScanner.nextLine();
                  }


                  //attendance

                  tempAttendance = fileScanner.nextLine();
                  String attendance[] = tempAttendance.split(",");
                  attendanceOfStudent = new Attendance();
                  for (int i = 0; i < attendance.length; i++ ) {
                    attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
                    }
                  tempReasonAttendance = fileScanner.nextLine();
                  String tempReason[] = tempReasonAttendance.split(",");
                  for (int i = 0; i < tempReason.length; i++ ) {
                    attendanceOfStudent.addReasonAbsent(tempReason[i]);
                    }
                  tempCovid = fileScanner.nextLine();
                  String covid[] = tempCovid.split(",");
                  for (int i = 0; i < covid.length; i++ ) {
                    attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
                    }
                  tempReasonCovid = fileScanner.nextLine();
                  String reasonCov[] = tempReasonCovid.split(",");
                  for (int i = 0; i < reasonCov.length; i++ ) {
                    attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
                    }

                    dates = new ArrayList<String>();

                    tempDate = fileScanner.nextLine();
                  String date[] = tempDate.split(",");
                  for (int i = 0; i < date.length; i++ ) {
                    dates.add(date[i]);
                    }

                    guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
                    guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
                    guardianOnePhoneNumber = fileScanner.nextLine();
                    guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
                    guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
                    guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                    guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
                    guardianTwoPhoneNumber = fileScanner.nextLine();
                    guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
                    guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

                    emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactOneHomeNumber = (fileScanner.nextLine());
                    emergencyContactOneCellNumber = (fileScanner.nextLine());
                    emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactTwoHomeNumber = (fileScanner.nextLine());
                    emergencyContactTwoCellNumber = (fileScanner.nextLine());

                    healthFactorOne = (fileScanner.nextLine()).toLowerCase();
                    healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                    healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                    healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

                Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,coun-
tryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail,
guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardianTwoCallAtWork, emergencyCon-
tactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCell-
Number,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyCon-
tactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired,
healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree,
healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                  listOfStudents.add(tempS);
                }
                fileScanner.close();

                return listOfStudents;
              }

        //access stored index of student in temp file
                        public static void index() {
                                  index = -1;
                                   try {
                                       fileScanner = new Scanner(new File("temp.txt"));
                                       index = Integer.parseInt(fileScanner.nextLine());

                                              fileScanner.close();
                                     } catch (FileNotFoundException e) {
                                       System.err.println("File not found! Choosing to quit now...");
```

```
                        System.exit(0);
                    }
        }

         //METHODS

        //find the number of times sabag/dour was not done
    public static int timesNotDone(Boolean[] array) {

        //counter
        int times = 0;
   for (int i = 0; i < array.length; i++) {

          //if boolean at index i of array is false, this means it was incomplete
     if (array [i] == false) {
     times++;
     }
   }
   return times;
}

//find the number of times sabag/dour was done
public static int timesDone(Boolean[] array) {

        //counter
   int times = 0;
   for (int i = 0; i < array.length; i++) {

        //if boolean at index i of array is true, this means it was complete
     if (array [i] == true) {
     times++;
     }
   }
   return times;
}

//find average method
public static int findAverage(int [] array) {
   int average = 0;
   int total = 0;
   int counter = 0;
   //calculate total
   for (int i = 0; i < array.length; i++) {
     total = total + array [i];
     counter ++;
   }
   //divide total by counter to find average
   average = total/counter;

   return average;
}



//find total method of integers in an array
public static int findTotal(int [] array) {
   int total = 0;
   for (int i = 0; i < array.length; i++) {
     total = total + array [i];
   }
   return total;
}



public static String findName(int [] array) {
   String name = "";
   int count = 0;
   for (int i = 0; i < array.length; i++) {
     if ((array [i] != 0) && (count == 0)) {
       name = "" + array [i];
       count++;
     } else if ((array [i] != 0)) {
        name = name + ", " + array [i];
     }
   }
   return name;

}
}
```

CLASS: MenuDOtherN.java

```
package com.example.test;

import com.vaadin.flow.router.Route;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Scanner;

import com.vaadin.flow.component.UI;
import com.vaadin.flow.component.button.Button;
import com.vaadin.flow.component.button.ButtonVariant;
import com.vaadin.flow.component.checkbox.Checkbox;
```

```java
import com.vaadin.flow.component.datepicker.DatePicker;
import com.vaadin.flow.component.formlayout.FormLayout;
import com.vaadin.flow.component.formlayout.FormLayout.ResponsiveStep;
import com.vaadin.flow.component.html.Footer;
import com.vaadin.flow.component.html.H2;
import com.vaadin.flow.component.html.Paragraph;
import com.vaadin.flow.component.orderedlayout.VerticalLayout;
import com.vaadin.flow.component.textfield.IntegerField;
import com.vaadin.flow.component.textfield.TextField;
import com.vaadin.flow.data.binder.Binder;

@Route(value = "menuDOtherN", layout = Welcome.class)
public class MenuDOtherN extends VerticalLayout {

	static Scanner fileScanner;
	static ArrayList <Student> listOfStudents = new ArrayList <Student>();

	public MenuDOtherN () {
		listOfStudents.removeAll(listOfStudents);
		listOfStudents = fileOneOpen();
		//introduce
		VerticalLayout tempp = new VerticalLayout();
		H2 intro = new H2 ("Set Up A New Student");
		intro.setMinWidth("700px");
	    intro.setSizeFull();
	    intro.getStyle().set("text-align", "center");

	    Paragraph intro2 = new Paragraph ("Add 'n/a' where field is not applicable");
		intro2.setMinWidth("700px");
	    intro2.setSizeFull();
	    intro2.getStyle().set("text-align", "center");

	    tempp.add(intro, intro2);

	    Binder<Student> personBinder = new Binder<>(Student.class);
	    Student temp = new Student();
	    personBinder.setBean(temp);
	    FormLayout formLayout = createFormLayout();

	    TextField fName = new TextField("First Name");
	    personBinder.forField(fName).bind(
	     Student::getFirstName,
	     Student::setFirstName);
	    fName.setRequired(true);
	    fName.setRequiredIndicatorVisible(true);
	    fName.setErrorMessage("This field is required");

	    TextField middleName = new TextField("Middle Name");
	    personBinder.forField(middleName).bind(
	                    Student::getMiddleName,
	                    Student::setMiddleName);
	    middleName.setRequired(true);
	    middleName.setRequiredIndicatorVisible(true);
	    middleName.setErrorMessage("This field is required");

	    TextField lastName = new TextField("Last Name");
	    personBinder.forField(lastName).bind(
	     Student::getLastName,
	     Student::setLastName
	                    );
	    lastName.setRequired(true);
	    lastName.setRequiredIndicatorVisible(true);
	    lastName.setErrorMessage("This field is required");

	    TextField address = new TextField("Address");
	    personBinder.forField(address).bind(
	                    Student::getAddress,
	                    Student::setAddress);
	    address.setRequired(true);
	    address.setRequiredIndicatorVisible(true);
	    address.setErrorMessage("This field is required");

	    TextField postalCode = new TextField("Postal Code");
	    personBinder.forField(postalCode).bind(
	                    Student::getPostalCode,
	                    Student::setPostalCode);
	    postalCode.setRequired(true);
	    postalCode.setRequiredIndicatorVisible(true);
	    postalCode.setErrorMessage("This field is required");

	    DatePicker datePicker = new DatePicker("Date Of Birth");
	    personBinder.forField(datePicker).bind(
	                    Student::getDateOfBirthLocalDate,
	                    Student::setDateOfBirthLocalDate
	                    );
	    address.setRequired(true);
	    address.setRequiredIndicatorVisible(true);
	    address.setErrorMessage("This field is required");

	    IntegerField age = new IntegerField("Age");
	    personBinder.forField(age).bind(
	                    Student::getAge,
	                    Student::setAge
	                    );
	    age.setRequiredIndicatorVisible(true);
	    age.setErrorMessage("This field is required");

	    TextField language = new TextField("Language");
	    personBinder.forField(language).bind(
	                    Student::getLanguage,
	                    Student::setLanguage);
```

```
        language.setRequired(true);
        language.setRequiredIndicatorVisible(true);
        language.setErrorMessage("This field is required");

        TextField country = new TextField("Country of Birth");
        personBinder.forField(country).bind(
                        Student::getCountryOfBirth,
                        Student::setCountryOfBirth);
        country.setRequired(true);
        country.setRequiredIndicatorVisible(true);
        country.setErrorMessage("This field is required");

        formLayout.add(fName, middleName, lastName);
        formLayout.add(address, postalCode);
        formLayout.add(datePicker, age, language, country);

        TextField gFirstName = new TextField("Guardian One First Name");
        personBinder.forField(gFirstName).bind(
         Student::getGuardianOneFirstName,
         Student::setGuardianOneFirstName);
        gFirstName.setRequired(true);
        gFirstName.setRequiredIndicatorVisible(true);
        gFirstName.setErrorMessage("This field is required");

        TextField gLastName = new TextField("Guardian One Last Name");
        personBinder.forField(gLastName).bind(
         Student::getGuardianOneLastName,
         Student::setGuardianOneLastName);
        gLastName.setRequired(true);
        gLastName.setRequiredIndicatorVisible(true);
        gLastName.setErrorMessage("This field is required");

        TextField gPhoneNum = new TextField("Guardian One Phone Number");
        personBinder.forField(gPhoneNum).bind(
         Student::getGuardianOnePhoneNumber,
         Student::setGuardianOnePhoneNumber);
        gPhoneNum.setRequired(true);
        gPhoneNum.setRequiredIndicatorVisible(true);
        gPhoneNum.setErrorMessage("This field is required");

        TextField gEmail = new TextField("Guardian One Email");
        personBinder.forField(gEmail).bind(
         Student::getGuardianOneEmail,
         Student::setGuardianOneEmail);
        gEmail.setRequired(true);
        gEmail.setRequiredIndicatorVisible(true);
        gEmail.setErrorMessage("This field is required");

Checkbox callAtWork1 = new Checkbox();
callAtWork1.setLabel("Call Guardian One at Work");
personBinder.forField(callAtWork1).bind(
        Student::isGuardianOneCallAtWork,
        Student::setGuardianOneCallAtWork);

TextField gFirstName2 = new TextField("Guardian Two First Name");
       personBinder.forField(gFirstName2).bind(
        Student::getGuardianTwoFirstName,
        Student::setGuardianTwoFirstName);

       TextField gLastName2 = new TextField("Guardian Two Last Name");
       personBinder.forField(gLastName2).bind(
        Student::getGuardianTwoLastName,
        Student::setGuardianTwoLastName);

       TextField gPhoneNum2 = new TextField("Guardian Two Phone Number");
       personBinder.forField(gPhoneNum2).bind(
        Student::getGuardianTwoPhoneNumber,
        Student::setGuardianTwoPhoneNumber);

       TextField gEmail2 = new TextField("Guardian Two Email");
       personBinder.forField(gEmail2).bind(
        Student::getGuardianTwoEmail,
        Student::setGuardianTwoEmail);

Checkbox callAtWork2 = new Checkbox();
callAtWork2.setLabel("Call Guardian Two at Work");
personBinder.forField(callAtWork2).bind(
        Student::isGuardianTwoCallAtWork,
        Student::setGuardianTwoCallAtWork);

formLayout.add(gFirstName, gLastName, gPhoneNum);
formLayout.add(gEmail, 3);
formLayout.add(callAtWork1, 3);
formLayout.add(gFirstName2, gLastName2, gPhoneNum2);
formLayout.add(gEmail2, 3);
formLayout.add(callAtWork2, 3);

TextField eContactOneFirstName = new TextField("Contact One First Name");
       personBinder.forField(eContactOneFirstName).bind(
        Student::getEmergencyContactOneFirstName,
        Student::setEmergencyContactOneFirstName);

       TextField eContactOneLastName = new TextField("Contact One Last Name");
       personBinder.forField(eContactOneLastName).bind(
        Student::getEmergencyContactOneLastName,
        Student::setEmergencyContactOneLastName);

       TextField eContactOneRelationship = new TextField("Contact One Relationship");
       personBinder.forField(eContactOneRelationship).bind(
        Student::getEmergencyContactOneRelationship,
        Student::setEmergencyContactOneRelationship);
```

```
        TextField eContactOneHomeNumber = new TextField("Contact One Home Number");
        personBinder.forField(eContactOneHomeNumber).bind(
          Student::getEmergencyContactOneHomeNumber,
          Student::setEmergencyContactOneHomeNumber);

        TextField eContactOneCellNumber = new TextField("Contact One Cell Number");
        personBinder.forField(eContactOneCellNumber).bind(
          Student::getEmergencyContactOneCellNumber,
          Student::setEmergencyContactOneCellNumber);

        TextField eContactTwoFirstName = new TextField("Contact Two First Name");
        personBinder.forField(eContactTwoFirstName).bind(
          Student::getEmergencyContactTwoFirstName,
          Student::setEmergencyContactTwoFirstName);

        TextField eContactTwoLastName = new TextField("Contact Two Last Name");
        personBinder.forField(eContactTwoLastName).bind(
          Student::getEmergencyContactTwoLastName,
          Student::setEmergencyContactTwoLastName);

        TextField eContactTwoRelationship = new TextField("Contact Two Relationship");
        personBinder.forField(eContactTwoRelationship).bind(
          Student::getEmergencyContactTwoRelationship,
          Student::setEmergencyContactTwoRelationship);

        TextField eContactTwoHomeNumber = new TextField("Contact Two Home Number");
        personBinder.forField(eContactTwoHomeNumber).bind(
          Student::getEmergencyContactTwoHomeNumber,
          Student::setEmergencyContactTwoHomeNumber);

        TextField eContactTwoCellNumber = new TextField("Contact Two Cell Number");
        personBinder.forField(eContactTwoCellNumber).bind(
          Student::getEmergencyContactTwoCellNumber,
          Student::setEmergencyContactTwoCellNumber);

        formLayout.add(eContactOneFirstName, eContactOneLastName, eContactOneRelationship, eContactOneHomeNumber);
        formLayout.add(eContactOneCellNumber, 2);
        formLayout.add(eContactTwoFirstName, eContactTwoLastName, eContactTwoRelationship, eContactTwoHomeNumber);
        formLayout.add(eContactTwoCellNumber, 2);

        TextField healthFactorOneName = new TextField("Health Factor One");
        personBinder.forField(healthFactorOneName).bind(
          Student::getHealthFactorOne,
          Student::setHealthFactorOne);

        Checkbox lifeThreatening1 = new Checkbox();
        lifeThreatening1.setLabel("Life Threatening");
personBinder.forField(lifeThreatening1).bind(
          Student::isHealthFactorOneLifeThreatening,
          Student::setHealthFactorOneLifeThreatening);

Checkbox planOfCareRequired1 = new Checkbox();
planOfCareRequired1.setLabel("Plan Of Care Required");
personBinder.forField(planOfCareRequired1).bind(
          Student::isHealthFactorOnePlanOfCareRequired,
          Student::setHealthFactorOnePlanOfCareRequired);

Checkbox medicationsRequired1 = new Checkbox();
medicationsRequired1.setLabel("Medications Required");
personBinder.forField(medicationsRequired1).bind(
          Student::isHealthFactorOneMedicationsRequired,
          Student::setHealthFactorOneMedicationsRequired);

TextField healthFactorTwoName = new TextField("Health Factor Two");
        personBinder.forField(healthFactorTwoName).bind(
          Student::getHealthFactorTwo,
          Student::setHealthFactorTwo);

        Checkbox lifeThreatening2 = new Checkbox();
        lifeThreatening2.setLabel("Life Threatening");
personBinder.forField(lifeThreatening2).bind(
          Student::isHealthFactorTwoLifeThreatening,
          Student::setHealthFactorTwoLifeThreatening);

Checkbox planOfCareRequired2 = new Checkbox();
planOfCareRequired2.setLabel("Plan Of Care Required");
personBinder.forField(planOfCareRequired2).bind(
          Student::isHealthFactorTwoPlanOfCareRequired,
          Student::setHealthFactorTwoPlanOfCareRequired);

Checkbox medicationsRequired2 = new Checkbox();
medicationsRequired2.setLabel("Medications Required");
personBinder.forField(medicationsRequired2).bind(
          Student::isHealthFactorTwoMedicationsRequired,
          Student::setHealthFactorTwoMedicationsRequired);

TextField healthFactorThreeName = new TextField("Health Factor Three");
        personBinder.forField(healthFactorThreeName).bind(
          Student::getHealthFactorThree,
          Student::setHealthFactorThree);

        Checkbox lifeThreatening3 = new Checkbox();
        lifeThreatening3.setLabel("Life Threatening");
personBinder.forField(lifeThreatening3).bind(
          Student::isHealthFactorThreeLifeThreatening,
          Student::setHealthFactorThreeLifeThreatening);

Checkbox planOfCareRequired3 = new Checkbox();
planOfCareRequired3.setLabel("Plan Of Care Required");
personBinder.forField(planOfCareRequired3).bind(
```

```
        Student::isHealthFactorThreePlanOfCareRequired,
        Student::setHealthFactorThreePlanOfCareRequired);

Checkbox medicationsRequired3 = new Checkbox();
medicationsRequired3.setLabel("Medications Required");
personBinder.forField(medicationsRequired3).bind(
        Student::isHealthFactorThreeMedicationsRequired,
        Student::setHealthFactorThreeMedicationsRequired);

TextField saparasDone = new TextField("Enter Sapras Memorized (seperated by a comma)");
personBinder.bind(saparasDone,
             Student -> Student.getSaparasDone(),
             (Student, title) -> {
                        Student.setSaparasDone(saparasDone.getValue());
             });

IntegerField currentSaparaMemorizing = new IntegerField("Current Sapara Memorizing");
personBinder.bind(currentSaparaMemorizing,
             Student -> Student.getCurrentSaparaMemorizing(),
             (Student, title) -> {
                        Student.setCurrentSaparaMemorizing(currentSaparaMemorizing.getValue());
             });

IntegerField saparasMemorizedT = new IntegerField("Total Saparas Memorized");
       personBinder.bind(saparasMemorizedT,
                        Student -> Student.getTotalSaparasDone(),
                         (Student, title) -> {
                         Student.setTotalSaparasDone(title);
                         if (saparasMemorizedT.getValue() <= 5) {
                             Student.setProgramChosen("beginner");
                           } else if (saparasMemorizedT.getValue() > 5 && saparasMemorizedT.getValue() <= 12) {
                                    Student.setProgramChosen("intermediate");
                           } else if ((saparasMemorizedT.getValue() > 12) && (saparasMemorizedT.getValue() != 30)) {
                                    Student.setProgramChosen("advanced");;
                           } else {
                                    Student.setProgramChosen("hafiz");
                                    currentSaparaMemorizing.setEnabled(false);
                           }

                         if ((saparasMemorizedT.getValue() == 30 )) {
                             saparasDone.setEnabled(false);
                               Student.setSaparasDone("all");
                               currentSaparaMemorizing.setEnabled(false);
                               Student.setCurrentSaparaMemorizing(0);
                             } else if ((saparasMemorizedT.getValue() == 0)) {
                                  saparasDone.setEnabled(false);
                                         Student.setSaparasDone("0");
                             }

                     });

       IntegerField curQuarter = new IntegerField("Dour Current Quarter");
       personBinder.forField(curQuarter).bind(
        Student::getCurrentQuarter,
        Student::setCurrentQuarter);

IntegerField curSapara = new IntegerField ("Dour Current Sapara");
personBinder.forField(curSapara).bind(
        Student::getDourCurrentSapara,
        Student::setDourCurrentSapara);

formLayout.add(saparasMemorizedT, 2);
formLayout.add(currentSaparaMemorizing, saparasDone, curQuarter, curSapara);

       formLayout.add(healthFactorOneName, 3);
       formLayout.add(lifeThreatening1, planOfCareRequired1, medicationsRequired1);

       formLayout.add(healthFactorTwoName, 3);
       formLayout.add(lifeThreatening2, planOfCareRequired2, medicationsRequired2);

       formLayout.add(healthFactorThreeName, 3);
       formLayout.add(lifeThreatening3, planOfCareRequired3, medicationsRequired3);

    // Footer
       Button done = new Button("Done", e -> {
                Boolean[] dourDoneOrNot = new Boolean[30];
                temp.setDourDoneOrNot(dourDoneOrNot);
                      temp.setTodayDourDoneOrNot(false);
                      int[] quarterNumDoneMonth = new int[30];
                      temp.setQuarterNumDoneMonth(quarterNumDoneMonth);
                      int todayQuartersDone, currentQuarter;
                      temp.setTodayQuartersDone(-1);
                      Boolean[] numOfDourSaparasDoneMonth = new Boolean[30];
                      temp.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);
                      Boolean todayDourSaparaDoneOrNot;
                      temp.setTodayDourSaparaDone(-1);
                      temp.setOpenDourNextFill(0);
                      temp.setLastRecord("11/11/2011");

                      Boolean[] sabaqDoneOrNot = new Boolean[30];
                      temp.setSabaqDoneOrNot(sabaqDoneOrNot);
                      temp.setOpenTodaySabaqDoneOrNot(false);
                      int[] linesMemorized = new int[30];
                      temp.setLinesMemorized(linesMemorized);
                      int todayLinesMemorized;
                      temp.setTodayLinesMemorized(0);
                      int[] mistakesMade = new int[30];
                      temp.setMistakesMade(mistakesMade);
                      int todayMistakesMade;
                      temp.setTodayMistakesMade(0);
                      Boolean[] numOfSaparasDoneMonth = new Boolean[30];
```

```
                        temp.setNumOfSaparasDoneMonth(numOfSaparasDoneMonth);
                        Boolean todaySaparaFinished;
                        temp.setTodaySaparaFinished(false);
                        int[] nameOfSaparasDoneMonth = new int[30];
                        temp.setNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);
                        temp.setTodaySaparaDone(-1);
                        temp.setOpenSaparaNextFill(0);


                       ArrayList<String> dates = new ArrayList<>();
                       temp.setDate(dates);
                         //Student tempS = new Student (temp.getFirstName(), temp.getMiddleName(), temp.getLastName(), temp.getAddress(),
temp.getDateOfBirth(), age, postalCode, language,countryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardia-
nOneLastName, guardianOnePhoneNumber,guardianOneEmail, guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber,
guardianTwoEmail, guardianTwoCallAtWork, emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergency-
ContactOneHomeNumber, emergencyContactOneCellNumber,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship,
emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCare-
Required, healthFactorOneMedicationsRequired, healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFac-
torTwoMedicationsRequired, healthFactorThree, healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  temp.getHeaealthFactor-
ThreeMedicationsRequired);


                listOfStudents.add(temp);
                closeFileOne(listOfStudents);
                UI.getCurrent().navigate("menu");
            });
            done.addThemeVariants(ButtonVariant.LUMO_PRIMARY, ButtonVariant.LUMO_CONTRAST);
            done.getStyle().set("margin-right", "var(--lumo-space-s)");

            Button anotherStudent = new Button("Cancel", 1 ->{
                listOfStudents.removeAll(listOfStudents);
                UI.getCurrent().navigate("menu");
            });
            anotherStudent.addThemeVariants(ButtonVariant.LUMO_TERTIARY, ButtonVariant.LUMO_CONTRAST);

            Footer footer = new Footer(done, anotherStudent);
            footer.getStyle().set("padding", "var(--lumo-space-wide-m)");

        footer.getStyle().set("text-align", "center");

            setAlignItems(Alignment.STRETCH);
            setPadding(false);
            setSpacing(false);
            getStyle().set("border", "1px solid var(--lumo-contrast-20pct)");

        tempp.add(formLayout, footer);
        tempp.setPadding(true);
        add(tempp);
           }

            public static void closeFileOne(ArrayList <Student> listOfStudents) {
                        PrintWriter pw = null;
                              try {
                                 pw = new PrintWriter(new File("../marchbreakia/student.txt"));
                              } catch (FileNotFoundException e) {
                                System.err.print("couldn't open file for writing!");
                                System.exit(0);
                              }



                              for (int y = 0; y < listOfStudents.size(); y++) {

                                if (y == 0) {
                               pw.println(listOfStudents.get(y).getFirstName());
                                } else {
                                  pw.println(listOfStudents.get(y).getFirstName());
                                }
                                pw.println(listOfStudents.get(y).getMiddleName());
                               pw.println(listOfStudents.get(y).getLastName());
                               pw.println(listOfStudents.get(y).getAddress());
                               pw.println(listOfStudents.get(y).getDateOfBirth());
                               pw.println(listOfStudents.get(y).getAge());
                               pw.println(listOfStudents.get(y).getPostalCode());
                               pw.println(listOfStudents.get(y).getLanguage());
                               pw.println(listOfStudents.get(y).getCountryOfBirth());
                        pw.println(listOfStudents.get(y).getProgramChosen());
                                pw.println(listOfStudents.get(y).getLastRecord());


                             String holder = "";
                            for (int k = 0; k < listOfStudents.get(y).getDourDoneOrNot().length; k++) {
                                 if (k == 0) {
                         holder = "" + listOfStudents.get(y).getDourDoneOrNot()[0];
                                } else {
                                holder = holder + "," + listOfStudents.get(y).getDourDoneOrNot()[k];
                                }
                                }
                                  pw.println(holder);

                                  holder = "";
                             for (int k = 0; k < listOfStudents.get(y).getQuarterNumDoneMonth().length; k++) {
                                 if (k == 0) {
                                  holder = "" + listOfStudents.get(y).getQuarterNumDoneMonth()[0];
                                } else {
                                holder = holder + "," + listOfStudents.get(y).getQuarterNumDoneMonth()[k];
                                }
                                }
                                  pw.println(holder);

                           pw.println(listOfStudents.get(y).getCurrentQuarter());
```

```
holder = "";
        for (int k = 0; k < listOfStudents.get(y).getNumOfDourSaparasDoneMonth().length; k++) {
       if (k == 0) {
        holder = "" + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[0];
      } else {
      holder = holder + "," + listOfStudents.get(y).getNumOfDourSaparasDoneMonth()[k];
      }
      }
        pw.println(holder);
  pw.println(listOfStudents.get(y).getDourCurrentSapara()); pw.println(listOfStudents.get(y).getDourNextFill());

  DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
     LocalDateTime firstNow = LocalDateTime.now();
    String alreadyDone = firstFormatter.format(firstNow);

  if (!(alreadyDone.equals(listOfStudents.get(y).getLastRecord()))) {
      pw.println(false);
      pw.println(-1);
      pw.println(false);
      pw.println(-1);
      pw.println(false);
      pw.println(-1);
      pw.println(-1);
      pw.println(false);
      pw.println(-1);
  } else {
       pw.println(listOfStudents.get(y).isTodayDourDoneOrNot());
       pw.println(listOfStudents.get(y).getTodayQuartersDone());
       pw.println(listOfStudents.get(y).isTodayDourSaparaDoneOrNot());
       pw.println(listOfStudents.get(y).getTodayDourSaparaDone());

    if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
          pw.println(listOfStudents.get(y).getTodaySabaqDoneOrNot());
          pw.println(listOfStudents.get(y).getTodayLinesMemorized());
          pw.println(listOfStudents.get(y).getTodayMistakesMade());
          pw.println(listOfStudents.get(y).isTodaySaparaFinished());
          pw.println(listOfStudents.get(y).getTodaySaparaDone());
    } else {
      pw.println(false);
      pw.println(-1);
      pw.println(-1);
      pw.println(false);
      pw.println(-1);
    }
  }



  if (!(listOfStudents.get(y).getProgramChosen().equals("hafiz"))) {
        holder = "";
    for (int k = 0; k < listOfStudents.get(y).getSabaqDoneOrNot().length; k++) {
        if (k == 0) {
         holder = "" + (listOfStudents.get(y).getSabaqDoneOrNot()[0]);
       } else {
       holder = holder + ("," + listOfStudents.get(y).getSabaqDoneOrNot()[k]);
       }
       }
    pw.println(holder);

    holder = "";
  for (int k = 0; k < listOfStudents.get(y).getLinesMemorized().length; k++) {
        if (k == 0) {
         holder = "" + (listOfStudents.get(y).getLinesMemorized()[0]);
       } else {
       holder = holder + ("," + listOfStudents.get(y).getLinesMemorized()[k]);
       }
       }
  pw.println(holder);

  holder = "";
  for (int k = 0; k < listOfStudents.get(y).getMistakesMade().length; k++) {
        if (k == 0) {
         holder = "" + (listOfStudents.get(y).getMistakesMade()[0]);
       } else {
       holder = holder + ("," + listOfStudents.get(y).getMistakesMade()[k]);
       }
       }
  pw.println(holder);

  holder = "";
    for (int k = 0; k < listOfStudents.get(y).getNumOfSaparasDoneMonth().length; k++) {
        if (k == 0) {
         holder = "" + (listOfStudents.get(y).getNumOfSaparasDoneMonth()[0]);
       } else {
       holder = holder + ("," + listOfStudents.get(y).getNumOfSaparasDoneMonth()[k]);
       }
       }
    pw.println(holder);

    holder = "";
    for (int k = 0; k < listOfStudents.get(y).getNameOfSaparasDoneMonth().length; k++) {
        if (k == 0) {
         holder = "" + (listOfStudents.get(y).getNameOfSaparasDoneMonth()[0]);
       } else {
       holder = holder + ("," + listOfStudents.get(y).getNameOfSaparasDoneMonth()[k]);
       }
       }
    pw.println(holder);
  pw.println(listOfStudents.get(y).getTotalSaparasDone());
       pw.println(listOfStudents.get(y).getSaparasDone());
```

```
pw.println(listOfStudents.get(y).getCurrentSaparaMemorizing()); pw.println(listOfStudents.get(y).getSaparaNextFill());
     } else {
       pw.println(false);
       pw.println(0);
       pw.println(0);
       pw.println(false);
       pw.println(0);
       pw.println(0);
       pw.println(0);
       pw.println(0);
       pw.println(0);
     }


     //attendance
    //printing to file for attendance
     holder = "";
           for (int k = 0; k < listOfStudents.get(y).getAttendance().size(); k++) {

           if (k == 0) {
            holder = "" + (listOfStudents.get(y).getAttendance().get(k));

          } else {
          holder = holder + ("," + listOfStudents.get(y).getAttendance().get(k));

          }
             }
           pw.println(holder);

           holder = "";
          //printing to file for reason absent
          for (int d = 0; d < listOfStudents.get(y).getReasonAbsent().size(); d++) {

           if (d == 0) {
            holder = "" + (listOfStudents.get(y).getReasonAbsent().get(d));

          } else {
          holder = holder + ("," + listOfStudents.get(y).getReasonAbsent().get(d));

          }
            }
          pw.println(holder);
          //printing to file for covid screening
          holder = "";
          for (int r = 0; r < listOfStudents.get(y).getCovidScreening().size(); r++) {

           if (r == 0) {
            holder = "" + (listOfStudents.get(y).getCovidScreening().get(r));

          } else {
          holder = holder + ("," + listOfStudents.get(y).getCovidScreening().get(r));

          }
        }
          pw.println(holder);

        //printing to file for reason covid screening was not done
          holder = "";
          for (int p = 0; p < listOfStudents.get(y).getReasonCovidScreening().size(); p++) {

           if (p == 0) {
            holder = "" + (listOfStudents.get(y).getReasonCovidScreening().get(p));

          } else {
          holder = holder + ("," + listOfStudents.get(y).getReasonCovidScreening().get(p));

          }
        }
          pw.println(holder);

        //printing to file for dates
          holder = "";
        for (int z = 0; z < listOfStudents.get(y).getDate().size(); z++) {

           if (z == 0) {
            holder = ""+(listOfStudents.get(y).getDate().get(z));

          } else {
          holder = holder + ("," + listOfStudents.get(y).getDate().get(z));

          }
        }
        pw.println(holder);
        pw.println(listOfStudents.get(y).getGuardianOneFirstName());
        pw.println(listOfStudents.get(y).getGuardianOneLastName());
        pw.println(listOfStudents.get(y).getGuardianOnePhoneNumber());
        pw.println(listOfStudents.get(y).getGuardianOneEmail());
        pw.println(listOfStudents.get(y).isGuardianOneCallAtWork());
        pw.println(listOfStudents.get(y).getGuardianTwoFirstName());
        pw.println(listOfStudents.get(y).getGuardianTwoLastName());
        pw.println(listOfStudents.get(y).getGuardianTwoPhoneNumber());
        pw.println(listOfStudents.get(y).getGuardianTwoEmail());
        pw.println(listOfStudents.get(y).isGuardianTwoCallAtWork());

        pw.println(listOfStudents.get(y).getEmergencyContactOneFirstName());
        pw.println(listOfStudents.get(y).getEmergencyContactOneLastName());
        pw.println(listOfStudents.get(y).getEmergencyContactOneRelationship());
        pw.println(listOfStudents.get(y).getEmergencyContactOneHomeNumber());
        pw.println(listOfStudents.get(y).getEmergencyContactOneCellNumber());
        pw.println(listOfStudents.get(y).getEmergencyContactTwoFirstName());
        pw.println(listOfStudents.get(y).getEmergencyContactTwoLastName());
```

```
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoRelationship());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoHomeNumber());
                        pw.println(listOfStudents.get(y).getEmergencyContactTwoCellNumber());

                        pw.println(listOfStudents.get(y).getHealthFactorOne());
                        pw.println(listOfStudents.get(y).isHealthFactorOneLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorOnePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorOneMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorTwo());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoPlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorTwoMedicationsRequired());
                        pw.println(listOfStudents.get(y).getHealthFactorThree());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeLifeThreatening());
                        pw.println(listOfStudents.get(y).isHealthFactorThreePlanOfCareRequired());
                        pw.println(listOfStudents.get(y).isHealthFactorThreeMedicationsRequired());

                            }
                    pw.close();


                            }


        public static ArrayList <Student> fileOneOpen() {
               try {
                fileScanner = new Scanner(new File("../marchbreakia/student.txt"));
              } catch (FileNotFoundException e) {
                System.err.println("File not found! Choosing to quit now...");

                System.exit(0);
              }

              //programChosen  - CHECK CONSTRUCTORS
              //add health factors to printing out in emergency situation stuff

             String firstName, middleName, lastName, address, dateOfBirth, postalCode, language, countryOfBirth, tempAttendance, tem-
pReasonAttendance, tempCovid, tempReasonCovid;

              Boolean[] dourDoneOrNot;
              Boolean todayDourDoneOrNot;
              int[] quarterNumDoneMonth;
              int todayQuartersDone, currentQuarter;
              Boolean[] numOfDourSaparasDoneMonth;
              Boolean todayDourSaparaDoneOrNot;
              int todayDourSaparaDone;
              int dourCurrentSapara, dourNextFill;

              String programChosen;
              String lastRecord;

              Boolean[] sabaqDoneOrNot;
              Boolean todaySabaqDoneOrNot;
              int[] linesMemorized;
              int todayLinesMemorized;
              int[] mistakesMade;
              int todayMistakesMade;
              Boolean[] numOfSaparasDoneMonth;
              Boolean todaySaparaFinished;
              int[] nameOfSaparasDoneMonth;
              int totalSaparasDone;
              int todaySaparaDone;
              String saparasDone;
              int currentSaparaMemorizing;
              int saparaNextFill = 0;

             int age;

             String tempDate;
             ArrayList<String> dates;


             String guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber;
             String guardianOneEmail;
             Boolean guardianOneCallAtWork;
             String guardianTwoFirstName, guardianTwoLastName;
             String guardianTwoPhoneNumber;
             String guardianTwoEmail;
             Boolean guardianTwoCallAtWork;

             String emergencyContactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship;
             String emergencyContactOneHomeNumber, emergencyContactOneCellNumber;
             String emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship;
             String emergencyContactTwoHomeNumber, emergencyContactTwoCellNumber;

             String healthFactorOne;
             Boolean healthFactorOneLifeThreatening,healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired;
             String healthFactorTwo;
             Boolean healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired;
             String healthFactorThree;
             Boolean healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired, healthFactorThreeMedicationsRequired;

             Attendance attendanceOfStudent;
             StudentProgress progressOfStudent;

             while (fileScanner.hasNextLine()) {
                    dourDoneOrNot = new Boolean[30];
                         quarterNumDoneMonth = new int[30];
                        numOfDourSaparasDoneMonth = new Boolean[30];
                        sabaqDoneOrNot = new Boolean[30];
                         linesMemorized = new int[30];
```

```
                    mistakesMade = new int[30];
                    numOfSaparasDoneMonth = new Boolean[30];
                   nameOfSaparasDoneMonth = new int[30];

            firstName = (fileScanner.nextLine()).toLowerCase();
            middleName = (fileScanner.nextLine()).toLowerCase();
            lastName = (fileScanner.nextLine()).toLowerCase();
            address = (fileScanner.nextLine()).toLowerCase();
            dateOfBirth = fileScanner.nextLine();
            age = Integer.parseInt(fileScanner.nextLine());
            postalCode = (fileScanner.nextLine()).toLowerCase();
            language = (fileScanner.nextLine()).toLowerCase();
            countryOfBirth = (fileScanner.nextLine()).toLowerCase();

            //progress of student
            programChosen = (fileScanner.nextLine()).toLowerCase();
            progressOfStudent = new StudentProgress();
            progressOfStudent.setProgramChosen(programChosen);

            lastRecord = (fileScanner.nextLine());
            progressOfStudent.setLastRecord(lastRecord);


            String tempDourDoneOrNot = fileScanner.nextLine();
            String strDourDoneOrNot[] = tempDourDoneOrNot.split(",");
            for (int i = 0; i < strDourDoneOrNot.length; i++) {
              dourDoneOrNot [i] = Boolean.parseBoolean(strDourDoneOrNot[i]);
            }
        progressOfStudent.setOpenDourDoneOrNot(dourDoneOrNot);

            String tempQuarterNumDoneMonth = fileScanner.nextLine();
            String strQuarterNumDoneMonth[] = tempQuarterNumDoneMonth.split(",");
            for (int i = 0; i < strQuarterNumDoneMonth.length; i++) {
              quarterNumDoneMonth [i] = Integer.parseInt(strQuarterNumDoneMonth[i]);
            }
        progressOfStudent.setQuarterNumDoneMonth(quarterNumDoneMonth);

            currentQuarter = Integer.parseInt(fileScanner.nextLine());     progressOfStudent.setOpenCurrentQuarter(currentQuarter);

            String tempNumOfDourSaparasDoneMonth = fileScanner.nextLine();
            String strNumOfDourSaparasDoneMonth[] = tempNumOfDourSaparasDoneMonth.split(",");
            for (int i = 0; i < strNumOfDourSaparasDoneMonth.length; i++) {
              numOfDourSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfDourSaparasDoneMonth[i]);
            }
        progressOfStudent.setNumOfDourSaparasDoneMonth(numOfDourSaparasDoneMonth);

            dourCurrentSapara = Integer.parseInt(fileScanner.nextLine());  progressOfStudent.setOpenDourCurrentSapara(dourCur-
rentSapara);

            dourNextFill = Integer.parseInt(fileScanner.nextLine());
        progressOfStudent.setOpenDourNextFill(dourNextFill);

            DateTimeFormatter firstFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
               LocalDateTime firstNow = LocalDateTime.now();
              String alreadyDone = firstFormatter.format(firstNow);

            if (!(alreadyDone.equals(lastRecord))) {
              if (programChosen.equals("hafiz")) {
                Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                int holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());

                todayDourDoneOrNot = false;
                todayDourSaparaDoneOrNot = false;
                todayQuartersDone = 0;
                todayDourSaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
        progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
        progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
        progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

               } else {
                Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                int holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                 temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                temporary = Boolean.parseBoolean(fileScanner.nextLine());
                holder = Integer.parseInt(fileScanner.nextLine());
                todayDourDoneOrNot = false;
                 todayQuartersDone = 0;
                 todayDourSaparaDoneOrNot = false;
                todayDourSaparaDone = 0;
                 todaySabaqDoneOrNot = false;
                todayLinesMemorized = 0;
                todayMistakesMade = 0;
                todaySaparaFinished = false;
                todaySaparaDone = 0;
                progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);
        progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);
        progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);
        progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);
        progressOfStudent.setOpenTodaySabaqDoneOrNot(todaySabaqDoneOrNot);
```

```
                progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);
                progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);
                progressOfStudent.setOpenTodaySaparaFinished(todaySaparaFinished);
                progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                     }


                     } else {
                       todayDourDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                       progressOfStudent.setOpenTodayDourDoneOrNot(todayDourDoneOrNot);

                       todayQuartersDone = Integer.parseInt(fileScanner.nextLine());
                       progressOfStudent.setOpenTodayQuartersDone(todayQuartersDone);

                       todayDourSaparaDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());
                      progressOfStudent.setOpenTodayDourSaparaDoneOrNot(todayDourSaparaDoneOrNot);

                      todayDourSaparaDone = Integer.parseInt(fileScanner.nextLine());
                      progressOfStudent.setOpenTodayDourSaparaDone(todayDourSaparaDone);

                       if (!(programChosen.equals("hafiz"))) {

                     todaySabaqDoneOrNot = Boolean.parseBoolean(fileScanner.nextLine());   progressOfStudent.setOpenTodaySabaqDoneOrNot(to-
daySabaqDoneOrNot);

                       todayLinesMemorized = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodayLinesMemorized(todayLinesMemorized);

                       todayMistakesMade = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodayMistakesMade(todayMistakesMade);

                       todaySaparaFinished = Boolean.parseBoolean(fileScanner.nextLine());
                progressOfStudent.setTodaySaparaFinished(todaySaparaFinished);

                       todaySaparaDone = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenTodaySaparaDone(todaySaparaDone);
                      } else {
                        Boolean temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        int holder = Integer.parseInt(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                        temporary = Boolean.parseBoolean(fileScanner.nextLine());
                        holder = Integer.parseInt(fileScanner.nextLine());
                      }
                     }



                     if (!(programChosen.equals("hafiz"))) {
                       String tempSabaqDoneOrNot = fileScanner.nextLine();
                     String strSabaqDoneOrNot[] = tempSabaqDoneOrNot.split(",");
                     for (int i = 0; i < strSabaqDoneOrNot.length; i++) {
                       sabaqDoneOrNot [i] = Boolean.parseBoolean(strSabaqDoneOrNot[i]);
                     }
                progressOfStudent.setOpenSabaqDoneOrNot(sabaqDoneOrNot);

                     String tempLinesMemorized = fileScanner.nextLine();
                     String strLinesMemorized[] = tempLinesMemorized.split(",");
                     for (int i = 0; i < strLinesMemorized.length; i++) {
                       linesMemorized [i] = Integer.parseInt(strLinesMemorized[i]);
                     }
                     progressOfStudent.setOpenLinesMemorized(linesMemorized);

                     String tempMistakesMade = fileScanner.nextLine();
                     String strMistakesMade[] = tempMistakesMade.split(",");
                     for (int i = 0; i < strMistakesMade.length; i++) {
                       mistakesMade [i] = Integer.parseInt(strMistakesMade[i]);
                     }
                     progressOfStudent.setOpenMistakesMade(mistakesMade);

                     String tempNumOfSaparasFinished = fileScanner.nextLine();
                     String strNumOfSaparasFinished [] = tempNumOfSaparasFinished.split(",");
                     for (int i = 0; i < strNumOfSaparasFinished.length; i++) {
                      numOfSaparasDoneMonth [i] = Boolean.parseBoolean(strNumOfSaparasFinished[i]);
                     }
                     progressOfStudent.setOpenNumOfSaparasDoneMonth(numOfSaparasDoneMonth);

                     String tempNameOfSaparasFinished = fileScanner.nextLine();
                     String strNameOfSaparasFinished [] = tempNameOfSaparasFinished.split(",");
                     for (int i = 0; i < strNameOfSaparasFinished.length; i++) {
                      nameOfSaparasDoneMonth [i] = Integer.parseInt(strNameOfSaparasFinished[i]);
                     }
                     progressOfStudent.setOpenNameOfSaparasDoneMonth(nameOfSaparasDoneMonth);

                     totalSaparasDone = Integer.parseInt(fileScanner.nextLine());
                     progressOfStudent.setOpenTotalSaparasDone(totalSaparasDone);

                     saparasDone = fileScanner.nextLine();
                     progressOfStudent.setOpenSaparasDone(saparasDone);

                     currentSaparaMemorizing = Integer.parseInt(fileScanner.nextLine());
                progressOfStudent.setOpenCurrentSaparaMemorizing(currentSaparaMemorizing);

                     saparaNextFill = Integer.parseInt(fileScanner.nextLine());
                     progressOfStudent.setOpenSaparaNextFill(saparaNextFill);
                     } else {
                       String hold = fileScanner.nextLine();
                       hold = fileScanner.nextLine();
                       hold = fileScanner.nextLine();
                       hold = fileScanner.nextLine();
                       hold = fileScanner.nextLine();
                       hold = fileScanner.nextLine();
```

```
                    hold = fileScanner.nextLine();
                    hold = fileScanner.nextLine();
                    hold = fileScanner.nextLine();
                  }


                  //attendance

                  tempAttendance = fileScanner.nextLine();
                  String attendance[] = tempAttendance.split(",");
                  attendanceOfStudent = new Attendance();
                  for (int i = 0; i < attendance.length; i++ ) {
                    attendanceOfStudent.addAttendance(Boolean.parseBoolean(attendance[i]));
                    }
                  tempReasonAttendance = fileScanner.nextLine();
                  String tempReason[] = tempReasonAttendance.split(",");
                  for (int i = 0; i < tempReason.length; i++ ) {
                    attendanceOfStudent.addReasonAbsent(tempReason[i]);
                    }
                  tempCovid = fileScanner.nextLine();
                  String covid[] = tempCovid.split(",");
                  for (int i = 0; i < covid.length; i++ ) {
                    attendanceOfStudent.addCovidScreening(Boolean.parseBoolean(covid[i]));
                    }
                  tempReasonCovid = fileScanner.nextLine();
                  String reasonCov[] = tempReasonCovid.split(",");
                  for (int i = 0; i < reasonCov.length; i++ ) {
                    attendanceOfStudent.addReasonCovidScreening(reasonCov[i]);
                    }

                    dates = new ArrayList<String>();

                    tempDate = fileScanner.nextLine();
                  String date[] = tempDate.split(",");
                  for (int i = 0; i < date.length; i++ ) {
                    dates.add(date[i]);
                    }

                    guardianOneFirstName = (fileScanner.nextLine()).toLowerCase();
                    guardianOneLastName = (fileScanner.nextLine()).toLowerCase();
                    guardianOnePhoneNumber = fileScanner.nextLine();
                    guardianOneEmail = (fileScanner.nextLine()).toLowerCase();
                    guardianOneCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());
                    guardianTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                    guardianTwoLastName = (fileScanner.nextLine()).toLowerCase();
                    guardianTwoPhoneNumber = fileScanner.nextLine();
                    guardianTwoEmail = (fileScanner.nextLine()).toLowerCase();
                    guardianTwoCallAtWork = Boolean.parseBoolean(fileScanner.nextLine());

                    emergencyContactOneFirstName = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactOneLastName = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactOneRelationship = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactOneHomeNumber = (fileScanner.nextLine());
                    emergencyContactOneCellNumber = (fileScanner.nextLine());
                    emergencyContactTwoFirstName = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactTwoLastName = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactTwoRelationship = (fileScanner.nextLine()).toLowerCase();
                    emergencyContactTwoHomeNumber = (fileScanner.nextLine());
                    emergencyContactTwoCellNumber = (fileScanner.nextLine());

                    healthFactorOne = (fileScanner.nextLine()).toLowerCase();
                    healthFactorOneLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorOnePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorOneMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorTwo = (fileScanner.nextLine()).toLowerCase();
                    healthFactorTwoLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorTwoPlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorTwoMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThree = (fileScanner.nextLine()).toLowerCase();
                    healthFactorThreeLifeThreatening = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThreePlanOfCareRequired = Boolean.parseBoolean(fileScanner.nextLine());
                    healthFactorThreeMedicationsRequired = Boolean.parseBoolean(fileScanner.nextLine());

                  Student tempS = new Student (firstName, middleName, lastName, address, dateOfBirth, age, postalCode, language,coun-
tryOfBirth, attendanceOfStudent, progressOfStudent, dates, guardianOneFirstName, guardianOneLastName, guardianOnePhoneNumber,guardianOneEmail,
guardianOneCallAtWork, guardianTwoFirstName, guardianTwoLastName, guardianTwoPhoneNumber, guardianTwoEmail, guardianTwoCallAtWork, emergencyCon-
tactOneFirstName, emergencyContactOneLastName, emergencyContactOneRelationship, emergencyContactOneHomeNumber, emergencyContactOneCell-
Number,emergencyContactTwoFirstName, emergencyContactTwoLastName, emergencyContactTwoRelationship, emergencyContactTwoHomeNumber, emergencyCon-
tactTwoCellNumber, healthFactorOne, healthFactorOneLifeThreatening, healthFactorOnePlanOfCareRequired, healthFactorOneMedicationsRequired,
healthFactorTwo, healthFactorTwoLifeThreatening, healthFactorTwoPlanOfCareRequired, healthFactorTwoMedicationsRequired, healthFactorThree,
healthFactorThreeLifeThreatening, healthFactorThreePlanOfCareRequired,  healthFactorThreeMedicationsRequired);
                    listOfStudents.add(tempS);
                }
                fileScanner.close();

                return listOfStudents;
              }


            //make new form layout
            private FormLayout createFormLayout() {
                 FormLayout billingAddressFormLayout = new FormLayout();
                 billingAddressFormLayout.setResponsiveSteps(
                             new ResponsiveStep("0", 1),
                         new ResponsiveStep("320px", 2),
                         new ResponsiveStep("500px", 3)
                 );
                 return billingAddressFormLayout;
             }

}
```

CLASS: Teacher.java

```java
package com.example.test;

public class Teacher {

		//instance variables
		private String firstName;
		private String lastName;
		private String password;

		//constructor
		public Teacher (String firstName, String lastName, String password) {
		  this.firstName = firstName;
		  this.password = password;
		  this.lastName = lastName;
		}

		//getter methods
		public String getFirstName() {
		  return firstName;
		}

		public String getLastName() {
		  return lastName;
		}

		public String getPassword() {
		  return password;
		}

		//setter methods
		public void setFirstName(String name) {
		  firstName = name;
		}

		public void setLastName(String name) {
		  lastName = name;
		}

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

CLASS: StudentProgress.java

```java
package com.example.test;

public class StudentProgress {

		//instance variables
		//program
		private String programChosen;

		//progress
		private String progressOfStudentDaily;
		private String progressOfStudentMonthly;

		private String lastRecord = "";

		//sapara
		private Boolean[] sabaqDoneOrNot = new Boolean[30];
		private Boolean todaySabaqDoneOrNot;
		private int[] linesMemorized = new int[30];
		private int todayLinesMemorized;
		private int[] mistakesMade = new int[30];
		private int todayMistakesMade;
		private Boolean[] numOfSaparasDoneMonth = new Boolean[30];
		private Boolean todaySaparaFinished;
		//if yes then add one to totalsaparadone
		private int[] nameOfSaparasDoneMonth = new int[30];
		private int totalSaparasDone;
		private int todaySaparaDone;
		//and add one to saparas done
		//COME BACK IF MORE THAN 30 THEN URGE QARI TO CHANGE STUDENT TO HAFIZ PROGRAM
		private String saparasDone;
		//run through array and see if new sapara entered already exists
		private int currentSaparaMemorizing;
		private int saparaNextFill = 0;

		//dour
		private Boolean[] dourDoneOrNot = new Boolean[30];
		private Boolean todayDourDoneOrNot;
		private int[] quarterNumDoneMonth = new int[30];
		private int todayQuartersDone;
		private int currentQuarter;
		//if quarter done plus current quarter is more than 4, then ask for new current sapara and add current sapara to dour saparas done.
Assign the % of the total to current quarter.
		private Boolean[] numOfDourSaparasDoneMonth = new Boolean[30];
		private Boolean todayDourSaparaDoneOrNot;
		private int todayDourSaparaDone;
		//this is just the last current sapara

		//private String dourSaparaDone;
		private int dourCurrentSapara;
		private int dourNextFill = 0;
```

```
            //constructors
            public StudentProgress(String programChosen, Boolean[] sabaqDoneOrNot, int[] linesMemorized, int[] mistakesMade, Boolean[] numOf-
SaparasDoneMonth, int[] nameOfSaparasDoneMonth, int totalSaparasDone, String saparasDone, int currentSaparaMemorizing, Boolean[] dourDoneOrNot,
int[] quarterNumDoneMonth, int currentQuarter, Boolean[] numOfDourSaparasDoneMonth, int dourCurrentSapara, String teacherComment) {

              this.programChosen = programChosen;
              this.sabaqDoneOrNot = sabaqDoneOrNot;
              this.linesMemorized = linesMemorized;
              this.mistakesMade = mistakesMade;
              this.numOfSaparasDoneMonth = numOfSaparasDoneMonth;
              this.nameOfSaparasDoneMonth =  nameOfSaparasDoneMonth;
              this.totalSaparasDone = totalSaparasDone;
              this.saparasDone = saparasDone;
              this.currentSaparaMemorizing = currentSaparaMemorizing;
              this.dourDoneOrNot= dourDoneOrNot;
              this.quarterNumDoneMonth = quarterNumDoneMonth;
              this.currentQuarter = currentQuarter;
              this.numOfDourSaparasDoneMonth = numOfDourSaparasDoneMonth;
              this.dourCurrentSapara = dourCurrentSapara;

            }

            public StudentProgress (String programChosen,
                                    String lastRecord, Boolean[] sabaqDoneOrNot, Boolean todaySabaqDoneOrNot, int[] linesMemorized,
                                    int todayLinesMemorized, int[] mistakesMade, int todayMistakesMade, Boolean[] numOfSaparasDoneMonth,
                                    Boolean todaySaparaFinished, int[] nameOfSaparasDoneMonth, int totalSaparasDone, int todaySaparaDone,
                                    String saparasDone, int currentSaparaMemorizing, int saparaNextFill, Boolean[] dourDoneOrNot,
                                    Boolean todayDourDoneOrNot, int[] quarterNumDoneMonth, int todayQuartersDone, int currentQuarter,
                                    Boolean[] numOfDourSaparasDoneMonth, Boolean todayDourSaparaDoneOrNot, int todayDourSaparaDone,
                                    int dourCurrentSapara, int dourNextFill, String teacherComment) {
                 super();
                            this.programChosen = programChosen;
                            this.lastRecord = lastRecord;
                            this.sabaqDoneOrNot = sabaqDoneOrNot;
                            this.todaySabaqDoneOrNot = todaySabaqDoneOrNot;
                            this.linesMemorized = linesMemorized;
                            this.todayLinesMemorized = todayLinesMemorized;
                            this.mistakesMade = mistakesMade;
                            this.todayMistakesMade = todayMistakesMade;
                            this.numOfSaparasDoneMonth = numOfSaparasDoneMonth;
                            this.todaySaparaFinished = todaySaparaFinished;
                            this.nameOfSaparasDoneMonth = nameOfSaparasDoneMonth;
                            this.totalSaparasDone = totalSaparasDone;
                            this.todaySaparaDone = todaySaparaDone;
                            this.saparasDone = saparasDone;
                            this.currentSaparaMemorizing = currentSaparaMemorizing;
                            this.saparaNextFill = saparaNextFill;
                            this.dourDoneOrNot = dourDoneOrNot;
                            this.todayDourDoneOrNot = todayDourDoneOrNot;
                            this.quarterNumDoneMonth = quarterNumDoneMonth;
                            this.todayQuartersDone = todayQuartersDone;
                            this.currentQuarter = currentQuarter;
                            this.numOfDourSaparasDoneMonth = numOfDourSaparasDoneMonth;
                            this.todayDourSaparaDoneOrNot = todayDourSaparaDoneOrNot;
                            this.todayDourSaparaDone = todayDourSaparaDone;
                            this.dourCurrentSapara = dourCurrentSapara;
                            this.dourNextFill = dourNextFill;
                   }

            public StudentProgress() {

            }

            public StudentProgress(String programChosen, String saparasDone, int totalSaparasDone, int currentSaparaMemorizing, int dourCur-
rentQuarter, int dourCurrentSapara) {
              this.programChosen = programChosen;
              this.saparasDone = saparasDone;
              this.totalSaparasDone = totalSaparasDone;
              this.currentSaparaMemorizing = currentSaparaMemorizing;
              this.currentQuarter = dourCurrentQuarter;
              this.dourCurrentSapara = dourCurrentSapara;
            }




            //getters and setters

            //PROGRAM CHOSEN
            public String getProgramChosen() {
              return programChosen;
            }
            public void setProgramChosen(String programChosen) {
              this.programChosen = programChosen;
            }


              //SABAQ

            public Boolean[] getSabaqDoneOrNot() {
                            return sabaqDoneOrNot;
                        }

                        public void setSabaqDoneOrNot(Boolean[] sabaqDoneOrNot) {
                            this.sabaqDoneOrNot = sabaqDoneOrNot;
                        }


                        public Boolean getTodaySabaqDoneOrNot() {
                            return todaySabaqDoneOrNot;
```

```
		}

		public void setTodaySabaqDoneOrNot(Boolean todaySabaqDoneOrNot) {
			this.todaySabaqDoneOrNot = todaySabaqDoneOrNot;
saparaNextFill++;
sabaqDoneOrNot[saparaNextFill%30] = todaySabaqDoneOrNot;
}

		public int[] getLinesMemorized() {
			return linesMemorized;
		}

		public void setLinesMemorized(int[] linesMemorized) {
			this.linesMemorized = linesMemorized;
		}

		public int getTodayLinesMemorized() {
			return todayLinesMemorized;
		}

		public void setTodayLinesMemorized(int todayLinesMemorized) {
			this.todayLinesMemorized = todayLinesMemorized;
linesMemorized[saparaNextFill%30] = todayLinesMemorized;
		}

		public int[] getMistakesMade() {
			return mistakesMade;
		}

		public void setMistakesMade(int[] mistakesMade) {
			this.mistakesMade = mistakesMade;
		}

		public int getTodayMistakesMade() {
			return todayMistakesMade;
		}

		public void setTodayMistakesMade(int todayMistakesMade) {
			this.todayMistakesMade = todayMistakesMade;
mistakesMade [saparaNextFill%30] = todayMistakesMade;
		}

		public Boolean[] getNumOfSaparasDoneMonth() {
			return numOfSaparasDoneMonth;
		}

		public void setNumOfSaparasDoneMonth(Boolean[] numOfSaparasDoneMonth) {
			this.numOfSaparasDoneMonth = numOfSaparasDoneMonth;
		}

		public Boolean isTodaySaparaFinished() {
			return todaySaparaFinished;
		}

		public void setTodaySaparaFinished(Boolean todaySaparaFinished) {
			this.todaySaparaFinished = todaySaparaFinished;
numOfSaparasDoneMonth [saparaNextFill%30] = todaySaparaFinished;
totalSaparasDone++;
if (todaySaparaFinished == false) {
  setTodaySaparaDone(0);
}
		}

		public int[] getNameOfSaparasDoneMonth() {
			return nameOfSaparasDoneMonth;
		}

		public void setNameOfSaparasDoneMonth(int [] nameOfSaparasDoneMonth) {
			this.nameOfSaparasDoneMonth = nameOfSaparasDoneMonth;
		}

		public int getTotalSaparasDone() {
			return totalSaparasDone;
		}

		public void setTotalSaparasDone(int totalSaparasDone) {
			this.totalSaparasDone = totalSaparasDone;
		}

		public int getTodaySaparaDone() {
			return todaySaparaDone;
		}

		public void setTodaySaparaDone(int todaySaparaDone) {
			this.todaySaparaDone = todaySaparaDone;
if (todaySaparaDone != 0) {
totalSaparasDone++;
}
nameOfSaparasDoneMonth [saparaNextFill%30] = todaySaparaDone;
		}

		public String getSaparasDone() {
			return saparasDone;
		}

		public void addSaparasDone(int saparasDoneNow) {
if (saparasDone.equals(null)) {
  saparasDone = "" + saparasDoneNow;
} else {
			saparasDone = saparasDone + ", " + saparasDoneNow;
}
```

```
                }

                public int getCurrentSaparaMemorizing() {
                        return currentSaparaMemorizing;
                }

                public void setCurrentSaparaMemorizing(int currentSaparaMemorizing) {
                        this.currentSaparaMemorizing = currentSaparaMemorizing;
                }

//DOUR
                public Boolean[] getDourDoneOrNot() {
                        return dourDoneOrNot;
                }

                public void setDourDoneOrNot(Boolean[] dourDoneOrNot) {
                        this.dourDoneOrNot = dourDoneOrNot;
                }

                public Boolean isTodayDourDoneOrNot() {
                        return todayDourDoneOrNot;
                }

                public void setTodayDourDoneOrNot(Boolean todayDourDoneOrNot) {
                        this.todayDourDoneOrNot = todayDourDoneOrNot;
  dourNextFill++;
  dourDoneOrNot[dourNextFill%30] = todayDourDoneOrNot;
                }

                public int[] getQuarterNumDoneMonth() {
                        return quarterNumDoneMonth;
                }

                public void setQuarterNumDoneMonth(int[] quarterNumDoneMonth) {
                        this.quarterNumDoneMonth = quarterNumDoneMonth;
                }

                public int getTodayQuartersDone() {
                        return todayQuartersDone;
                }

                public void setTodayQuartersDone(int todayQuartersDone) {
                        this.todayQuartersDone = todayQuartersDone;
  quarterNumDoneMonth [dourNextFill%30] = todayQuartersDone;
                }

                public int getCurrentQuarter() {
                        return currentQuarter;
                }

                public void setCurrentQuarter(int currentQuarter) {
                        this.currentQuarter = currentQuarter;
                }

                public Boolean[] getNumOfDourSaparasDoneMonth() {
                        return numOfDourSaparasDoneMonth;
                }

                public void setNumOfDourSaparasDoneMonth(Boolean[] numOfDourSaparasDoneMonth) {
                        this.numOfDourSaparasDoneMonth = numOfDourSaparasDoneMonth;
                }

                public Boolean isTodayDourSaparaDoneOrNot() {
                        return todayDourSaparaDoneOrNot;
                }

                public void setTodayDourSaparaDoneOrNot(Boolean todayDourSaparaDoneOrNot) {
                        this.todayDourSaparaDoneOrNot = todayDourSaparaDoneOrNot;
  numOfDourSaparasDoneMonth [dourNextFill%30] = todayDourSaparaDoneOrNot;
  if (todayDourSaparaDoneOrNot == false) {
    setTodayDourSaparaDone(0);
  }

                }

                public int getTodayDourSaparaDone() {
                        return todayDourSaparaDone;
                }

                public void setTodayDourSaparaDone(int todayDourSaparaDone) {
                        this.todayDourSaparaDone = todayDourSaparaDone;
                }

                public int getDourCurrentSapara() {
                        return dourCurrentSapara;
                }

                public void setDourCurrentSapara(int dourCurrentSapara) {
                        this.dourCurrentSapara = dourCurrentSapara;
                }

 //OTHER


//SETTERS FOR FILE OPEN
public void setOpenProgramChosen(String programChosen) {
                this.programChosen = programChosen;
        }

        public void setOpenProgressOfStudentDaily(String progressOfStudentDaily) {
                this.progressOfStudentDaily = progressOfStudentDaily;
```

```
}

public void setOpenProgressOfStudentMonthly(String progressOfStudentMonthly) {
        this.progressOfStudentMonthly = progressOfStudentMonthly;
}

public void setOpenSabaqDoneOrNot(Boolean[] sabaqDoneOrNot) {
        this.sabaqDoneOrNot = sabaqDoneOrNot;
}

public void setOpenTodaySabaqDoneOrNot(Boolean todaySabaqDoneOrNot) {
        this.todaySabaqDoneOrNot = todaySabaqDoneOrNot;
}

public void setOpenLinesMemorized(int[] linesMemorized) {
        this.linesMemorized = linesMemorized;
}

public void setOpenTodayLinesMemorized(int todayLinesMemorized) {
        this.todayLinesMemorized = todayLinesMemorized;
}

public void setOpenMistakesMade(int[] mistakesMade) {
        this.mistakesMade = mistakesMade;
}

public void setOpenTodayMistakesMade(int todayMistakesMade) {
        this.todayMistakesMade = todayMistakesMade;
}

public void setOpenNumOfSaparasDoneMonth(Boolean[] numOfSaparasDoneMonth) {
        this.numOfSaparasDoneMonth = numOfSaparasDoneMonth;
}

public void setOpenTodaySaparaFinished(Boolean todaySaparaFinished) {
        this.todaySaparaFinished = todaySaparaFinished;
}

public void setOpenNameOfSaparasDoneMonth(int[] nameOfSaparasDoneMonth) {
        this.nameOfSaparasDoneMonth = nameOfSaparasDoneMonth;
}

public void setOpenTotalSaparasDone(int totalSaparasDone) {
        this.totalSaparasDone = totalSaparasDone;
}

public void setOpenTodaySaparaDone(int todaySaparaDone) {
        this.todaySaparaDone = todaySaparaDone;
}

public void setOpenSaparasDone(String saparasDone) {
        this.saparasDone = saparasDone;
}

public void setOpenCurrentSaparaMemorizing(int currentSaparaMemorizing) {
        this.currentSaparaMemorizing = currentSaparaMemorizing;
}

public void setOpenSaparaNextFill(int saparaNextFill) {
        this.saparaNextFill = saparaNextFill;
}

public void setOpenDourDoneOrNot(Boolean[] dourDoneOrNot) {
        this.dourDoneOrNot = dourDoneOrNot;
}

public void setOpenTodayDourDoneOrNot(Boolean todayDourDoneOrNot) {
        this.todayDourDoneOrNot = todayDourDoneOrNot;
}

public void setOpenQuarterNumDoneMonth(int[] quarterNumDoneMonth) {
        this.quarterNumDoneMonth = quarterNumDoneMonth;
}

public void setOpenTodayQuartersDone(int todayQuartersDone) {
        this.todayQuartersDone = todayQuartersDone;
}

public void setOpenCurrentQuarter(int currentQuarter) {
        this.currentQuarter = currentQuarter;
}

public void setOpenNumOfDourSaparasDoneMonth(Boolean[] numOfDourSaparasDoneMonth) {
        this.numOfDourSaparasDoneMonth = numOfDourSaparasDoneMonth;
}

public void setOpenTodayDourSaparaDoneOrNot(Boolean todayDourSaparaDoneOrNot) {
        this.todayDourSaparaDoneOrNot = todayDourSaparaDoneOrNot;
}

public void setOpenTodayDourSaparaDone(int todayDourSaparaDone) {
        this.todayDourSaparaDone = todayDourSaparaDone;
}

public void setOpenDourCurrentSapara(int dourCurrentSapara) {
        this.dourCurrentSapara = dourCurrentSapara;
}

public void setOpenDourNextFill(int dourNextFill) {
        this.dourNextFill = dourNextFill;
}
```

```
    //METHODS

    public static int timesNotDone(Boolean[] array) {
      int times = 0;
      for (int i = 0; i < array.length; i++) {
        if (array [i] == false) {
        times++;
        }
      }
      return times;
    }

    public static int timesDone(Boolean[] array) {
      int times = 0;
      for (int i = 0; i < array.length; i++) {
        if (array [i] == true) {
        times++;
        }
      }
      return times;
    }

    public static int findAverage(int [] array) {
      int average = 0;
      int total = 0;
      int counter = 0;
      for (int i = 0; i < array.length; i++) {
        total = total + array [i];
        counter ++;
      }
      average = total/counter;

      return average;
    }



    public static int findTotal(int [] array) {
      int total = 0;
      for (int i = 0; i < array.length; i++) {
        total = total + array [i];
      }
      return total;
    }

    public static String findName(int [] array) {
      String name = "";
      for (int i = 0; i < array.length; i++) {
        if (array [i] != 0) {
          name = name + array [i] + ", ";
        }
      }
      return name;

    }

    public String getLastRecord() {
                return lastRecord;
          }

          public void setLastRecord(String lastRecord) {
                this.lastRecord = lastRecord;
          }

  public int getSaparaNextFill() {
    return saparaNextFill;
  }

  public int getDourNextFill() {
    return dourNextFill;
  }


}
```