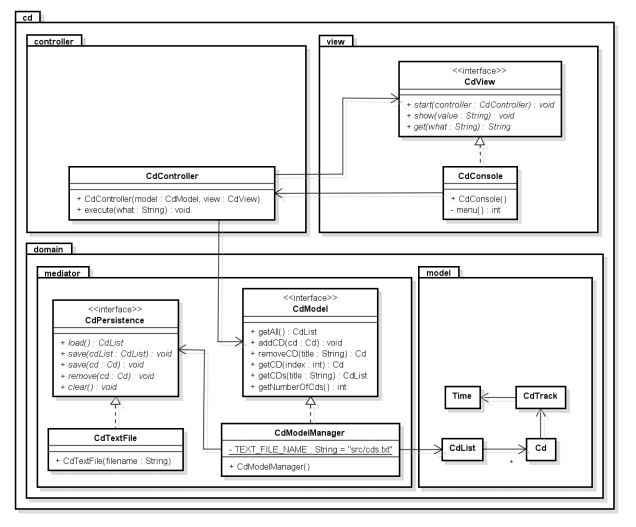
Exercise X8 – Model-View-Controller:

A) Find (and insert into a document) a general diagram of a Model-View-Controller design pattern.

B) Describe the overall purpose for the Model-View-Controller design pattern.

Model–view–controller (MVC) is a software architectural pattern for implementing systems with user interfaces. MVC is splitting application into three main parts. The purpose of that is archive situation when every component is doing one thing and make more code more readable and easier to work on.

C) Describe in text each part of the Model-View-Controller. Hereby purpose and responsibility for each part of the pattern.

**Model** - representation of problem or logic of application.

**View** - describing how to show some parts of model in user interface.

**Controller** - taking input form a user and managing updating model and refreshing view.

D) Describe how to use a Model-View-Controller design pattern.

To use MVC programmer should split system into three parts. place interface in view, application logic and date managing in model and reactions on user input into controller and only allow them to communicate. None of these parts should include things that other parts are responsible for, to avoid spaghetti code. It is highly recommended to use packages.

E) Give overall a few remarks to implementation, what to remember.

It is important to remember which part is responsible for and following a rule that every component is making one and only one thing. Programmer have to keep in mind that main reason of using mvc is to make work easier and follow the “*divide and conquer”* rule.

F) Insert code examples (your own code) for each part of the Model-View-Controller pattern. Not full code only code fractions directly related to the pattern.

Show List method form Controller class:

public void showList(ProxyTripList trips) {

System.out.println("method called");

if (tripList!=null) {

System.out.println(trips.getSize());

System.out.println("wennt inside if");

trips.sort();

ObservableList<Trip> data = FXCollections.observableArrayList();

for (int i = 0; i < trips.getSize(); i++) {

data.add(trips.get(i));

}

System.out.println(data);

System.out.println(data.size());

tripList.setItems(data);

}

}

Load buses method from DateHandler class:

public ObservableList<Bus> loadBuses(String busType, LocalDate startDatePicker, LocalDate endDatePicker,

String fieldStartTime, String fieldEndTime) {

BusList buses;

if (busType.equals("Mini Bus"))

buses = new BusList(busList.searchByType("server.domain.model.MiniBus"));

else if (busType.equals("Party Bus"))

buses = new BusList(busList.searchByType("server.domain.model.PartyBus"));

else if (busType.equals("Luxury Bus"))

buses = new BusList(busList.searchByType("server.domain.model.LuxuryBus"));

else

buses = new BusList(busList.searchByType("server.domain.model.ClassicBus"));

if (startDatePicker != null && endDatePicker != null && validateTimeField(fieldStartTime)

&& validateTimeField(fieldEndTime)) {

String[] lineToken = fieldStartTime.split(":");

int hours = Integer.parseInt(lineToken[0]);

int minutes = Integer.parseInt(lineToken[1]);

Date dateStart = new Date(startDatePicker.getYear() - 1900, startDatePicker.getMonthValue(),

startDatePicker.getDayOfMonth(), hours, minutes);

lineToken = fieldEndTime.split(":");

hours = Integer.parseInt(lineToken[0]);

minutes = Integer.parseInt(lineToken[1]);

Date dateEnd = new Date(endDatePicker.getYear() - 1900, endDatePicker.getMonthValue(),

endDatePicker.getDayOfMonth(), hours, minutes);

buses = buses.getAvailable(dateStart, dateEnd);

}

ObservableList<Bus> items = FXCollections.observableArrayList();

items.addAll(buses.getArrayBuses());

return items;

}

Main scene from server view:

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<?import javafx.scene.text.Font?>

<AnchorPane xmlns:fx="http://javafx.com/fxml/1" id="AnchorPane" maxHeight="-Infinity" maxWidth="-Infinity"

minHeight="-Infinity"

minWidth="-Infinity" prefHeight="600.0" prefWidth="1010.0"

xmlns="http://javafx.com/javafx/8.0.91" fx:controller="server.controller.Controller">

<BorderPane onMouseClicked="#changeView" prefHeight="600.0" prefWidth="1010.0">

<bottom>

<Label alignment="CENTER\_RIGHT" prefWidth="1000.0" text="Copyright Â© 2016 Group1. All rights reserved.">

<BorderPane.margin>

<Insets right="10.0"/>

</BorderPane.margin>

</Label>

</bottom>

<center>

<VBox prefHeight="360.0" prefWidth="1000.0">

<padding>

<Insets bottom="20.0" left="10.0" right="10.0"/>

</padding>

<Label alignment="CENTER" prefHeight="140.0" prefWidth="1010.0" text="VIA Bus" textAlignment="CENTER">

<font>

<Font size="100.0"/>

</font>

<VBox.margin>

<Insets bottom="5.0"/>

</VBox.margin>

</Label>

<Button fx:id="createTour" mnemonicParsing="false" onMouseClicked="#changeView" prefHeight="50.0"

prefWidth="1010.0" text="Create tour">

<font>

<Font size="20.0" fx:id="x1"/>

</font>

<VBox.margin>

<Insets bottom="20.0" top="20.0" fx:id="x2"/>

</VBox.margin>

</Button>

<Button fx:id="mkReservation" font="$x1" mnemonicParsing="false" onMouseClicked="#changeView"

prefHeight="50.0" prefWidth="1010.0" text="Make reservation" VBox.margin="$x2"/>

<Button fx:id="findTrip" font="$x1" mnemonicParsing="false" onMouseClicked="#changeView"

prefHeight="50.0" prefWidth="1010.0" text="Find Reservation / Tour" VBox.margin="$x2"/>

<Label alignment="CENTER" prefHeight="17.0" prefWidth="1010.0" text="Overview">

<font>

<Font size="17.0"/>

</font>

</Label>

<ListView fx:id="tripList" prefHeight="200.0" prefWidth="200.0"/>

</VBox>

</center>

<top>

<MenuBar fx:id="menu">

<Menu mnemonicParsing="false" text="Home">

<MenuItem fx:id="homeHome" mnemonicParsing="false" onAction="#changeViewMenu" text="Home"/>

<MenuItem fx:id="homeTour" mnemonicParsing="false" onAction="#changeViewMenu" text="Create tour"/>

<MenuItem fx:id="homeReserve" mnemonicParsing="false" onAction="#changeViewMenu"

text="Make reservation"/>

<MenuItem fx:id="homeSearch" mnemonicParsing="false" onAction="#changeViewMenu" text="Search"/>

</Menu>

<Menu mnemonicParsing="false" text="Bus List">

<MenuItem fx:id="homeBus" mnemonicParsing="false" onAction="#changeViewMenu" text="Bus List"/>

<MenuItem fx:id="homeBusAdd" mnemonicParsing="false" onAction="#changeViewMenu" text="Add bus"/>

</Menu>

<Menu mnemonicParsing="false" text="Chauffeur list">

<MenuItem fx:id="homeDriver" mnemonicParsing="false" onAction="#changeViewMenu"

text="Chauffeur list"/>

<MenuItem fx:id="homeDriverAdd" mnemonicParsing="false" onAction="#changeViewMenu"

text="Add chauffeur"/>

</Menu>

</MenuBar>

</top>

</BorderPane>

</AnchorPane>

G) Create and insert into the document a UML class diagram of the Model-View-Controller design pattern you implemented.

