**Lab 03: AppointmentApp**

*In this lab, you will:*

* *Work with and complete an ‘Appointment’ application.*
* *Further to lab 2, you will work with more examples of file-based data persistence.*
* Add and cancel appointment, as the program user, and store them back to a text file.
* *Extend a class by implementing a subclass of Appointment (PromoAppointment)*

*… which will result in the following outcomes:*

* *Further practice on the use of Readers and Writers.*
* Practice the use of java time classes (LocalTime).
* Knowledge of the use of ArrayLists as field variables.
* Knowledge of subclassing and adapting a controller type to determine at run time the kind of Appointment object created.

Table of Contents

[1.1 Preliminary 3](#_Toc503272551)

[1.2 Exercise 1 3](#_Toc503272552)

[1.3 Exercise 2 4](#_Toc503272553)

[1.4 Exercise 3 5](#_Toc503272554)

Figure 1: AppointmentAppStarter 3

## Preliminary

* Download the lab 3 zip file <**Lab3\_AppointmentAppStarter.zip**> from GCULearn and unzip it.
* Open the **Lab3\_AppointmentAppStarter** project in NetBeans:

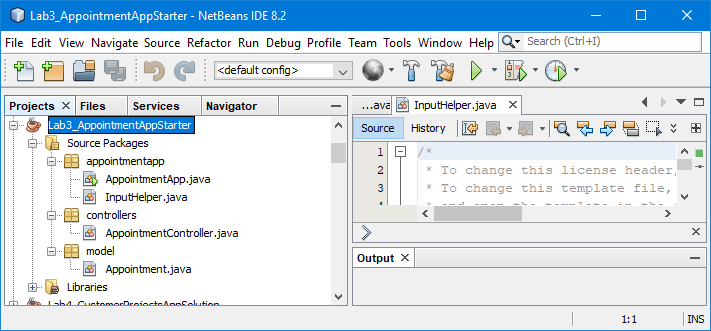


Figure : AppointmentAppStarter

## Exercise 1

1. Open the **Appointment** class, in the model folder, and note the attributes: **customerName**, **appointmentTime** and **appointmentLength**. Constructor, getter and setter methods are defined as well as the overridden method **toString()**.

Note, times are specified as **LocalTime** objects.

1. Now open the *appointments20161001.txt* file and note the format:

**"Lynn Kelly","10:00",30**

**"Ciara O'Donnell","11:15",45**

**"Louise Kelly","13:15",60**

This delimited text file is used to persist appointment details. Note that each appointment is represented on a single line. Open the AppointmentCntroller class: a constructor method requests a date (and file type) from the user and passes the filename into a load method. Examine the **loadAppointmentsFromTextFile()** private method:

**private void loadAppointmentsFromTextFile(String fileName) {**

**char DELIMITER=',';**

**try (BufferedReader br =**

**new BufferedReader(new FileReader(fileName))) {**

**String[] temp;**

**String line = br.readLine();**

**temp=line.split(Character.toString(DELIMITER));**

**customerName = stripQuotes(temp[0]);**

**localTimeStr = stripQuotes(temp[1]);**

**appointmentTime = LocalTime.parse(localTimeStr);**

**appointmentLength = Integer.parseInt(temp[2]);**

A stream is opened to the text file and the **readLine()** method used to read a line from the text file into a **String** variable called **line**. This string is then split up an array of **String**s – **temp** - using the **split()** method. Each element of the array holds a piece of text which has been delimited using, in this case, a comma – CSV file. Each element is accessed in turn, and, where necessary, converted into a variable of an appropriate type, e.g. **int**, **LocalTime** etc. which are then used to create an **Appointment** object.

**Appointment appointment = new Appointment(customerName,**

**appointmentTime,**

**appointmentLength);**

The appointment is then added to the appointments **ArrayList** object.

1. Run the **AppointmentApp** class and test that the appointments can be successfully loaded and displayed. Add appointments and cancel appointments and store them back to a text file. Ensure you can reload the file correctly.

## Exercise 2

We are now going to extend the app to include persistence using object streams.

1. Open the **AppointmentController** class and implement the **loadAppointmentsFromObjectFile()** method using a similar approach to Lab 2.
2. Implement the **storeAppointmentsToObjectFile()** method using a similar approach to Lab 2.
3. Run the **AppointmentApp** class and test that a new appointments file can be created and stored as an object file and then successfully loaded and displayed.

## Exercise 3

We are now going to implement Promo Appointments which are specialized versions of **Appointment** which add a string attribute **promoCode** and an integer **discountPercentage**.

1. Implement the **PromoAppointment** subclass making the necessary changes to the **Appointment** class.
2. Adjust the controller class method **loadAppointmentsFromTextFile()** to be able to determine when a **PromoAppointment** object should be created rather than an **Appointment** object.
3. Adjust the controller method **addAppointment()** to ask the user if a **PromoAppointment** is required, and, if so, ask for the promo code and discount percentage, create a **PromoAppointment** object and add it to the appointments attribute.
4. Run the **AppointmentApp** class and test you can load and store appointments, including promo appointments, from both text files and object files; and, add and cancel appointments of both types.