**Lab 02 CarInsuranceApp**

*In this lab, you will:*

* *Work with and complete a ‘starter’ car insurance application.*
* *See examples of file-based data persistence*
* *Work with example code that uses java classes to read and clean data. These simple introductory examples, however, speak to important skills in the context of modern programming.*
* *Complete the partial implementation of the application, by extending java objects and working with another controller class in the context of data reading and writing vis a vis the general notion of programming basic persistence of object states.*

*… which will result in the following outcomes:*

* *Appreciation of simple, structured data and how it can be read and cleaned from a java program using native java objects.*
* Appreciation of how different data types can be created by parsing various data String formats.
* Knowledge of how to storage formatted data using different approaches based on object readers and streams.

Table of Contents

[1.1 Preliminary 3](#_Toc503268345)

[1.2 Exercise 1 4](#_Toc503268346)

[1.3 Exercise 2 5](#_Toc503268347)

[1.4 Exercise 3 6](#_Toc503268348)

[1.5 Exercise 4 7](#_Toc503268349)

Figure 1: CarInsuranceStarter 3

Figure 2: Project files and folders in windows explorer 3

## Preliminary

* Download the lab 2 zip <**Lab2\_CarInsuranceAppStarter.zip**> file from GCULearn.
* unzip and open the **Lab2\_CarInsuranceAppStarter** project in NetBeans:

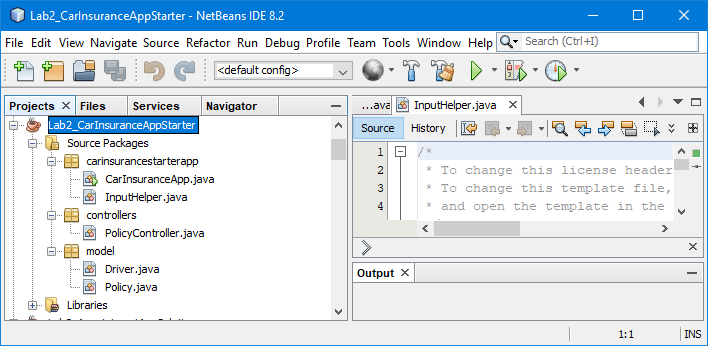


Figure : CarInsuranceAppStarter

* Browse, using windows explorer, if you are using windows (equivalent like Finder on MacOS) to the project folder
* Note the presence of the text files. These will be used to implement a strategy of object persistence, which is a key general theme of this lab.
* You can also view the files and folder structure in the Files view within Netbeans.

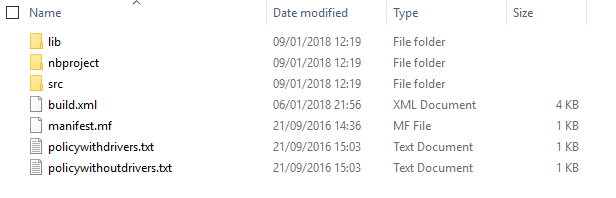


Figure : Project files and folders in windows explorer

## Exercise 1

1. Open the **Policy** class, in the model folder, and note the attributes: **policyId**, **policyType**, **policyOwner, carReg, carDescription** and **policyStartDate**. Constructor, getter and setter methods are defined as well as the overridden method **toString()**.

Note, dates are specified as **Calendar** objects and a private method **getFormattedStartDate()**is used to produce a specified string format for output:

**private String getFormattedStartDate()**

**{**

**SimpleDateFormat formatter =**

**new SimpleDateFormat("yyyy-MM-dd");**

**String dateString =**

**formatter.format(this.policyStartDate.getTime());**

**return dateString;**

**}**

This is then used in the **toString()** method:

@Override

**public String toString() {**

**return "\nId: " + Integer.toString(this.policyId) +**

**" Owner: " + this.policyOwner +**

**" Reg: " + this.carReg +**

**" Description: " + this.carDescription +**

**" Start: " + getFormattedStartDate() + "\n";**

**}**

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

**1,"Fully Comprehensive","Genesis","NC70 GEN","Green Jaguar XE","2016-09-01"**

This delimited text file is used to persist policy details. Open the controller class: a constructor method requests a filename (and file type) from the user and passes the filename into a load method. Examine the **loadPolicyFromTextFile()** private method:

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

**char DELIMITER=',';**

**try (BufferedReader br =**

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

**String[] temp;**

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

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

**int policyID = Integer.parseInt(temp[0]);**

**String policyType = stripQuotes(temp[1]);**

**String policyOwner = stripQuotes(temp[2]);**

**String carReg = stripQuotes(temp[3]);**

**String carDescription = stripQuotes(temp[4]);**

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**, **Calendar** etc. which are then used to create a **Policy** object.

A private method **stripQuotes()** is used to strip quotes from the start and end of **String**s.

Note a wee bit of work has to be undertaken to convert a date represented as a **String** into a **Calendar** object:

**DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");**

**String policyStartDateStr = stripQuotes(temp[5]);**

**Date policyStartDate;**

**policyStartDate = dateFormat.parse(policyStartDateStr);**

**policyStart = Calendar.getInstance();**

**policyStart.setTime(policyStartDate);**

**…**

**policy = new Policy(policyID, policyType, policyOwner, carReg,**

**carDescription, policyStart);**

1. Run the **CarInsuranceApp** class and test that the policy can be successfully loaded and displayed. Remember that you may have to adjust the configuration of the project to identify this as the location of the **main()** method.

## Exercise 2

We are now going to add a list of named drivers to the policy.

1. Open the **Policy** class and uncomment code which refers to the attribute **namedDrivers**. Extend the **toString()** method to display named drivers.
2. Open the *policywithdrivers.txt* file and note the format including named drivers:

**1,”Fully Comprehensive”,”Genesis”,”NC70 GEN”,”Green Jaguar XE”,”2016-09-01”,3**

**“Tony”,”Banks”,”1950-03-27”**

**“Phil”,”Collins”,”1951-01-30”**

**“Mike”,”Rutherford”,”1950:10:02”**

The additional value at the end of the first line specifies the number of named drivers i.e. the number of following lines which represent named driver information.

1. Open the **PolicyController** class and extend the private method **loadPolicyFromTextFile()** as follows:

**noDrivers = Integer.parseInt(temp[6]);**

**ArrayList<Driver> namedDrivers = new ArrayList<>();**

**Driver newDriver;**

**for (int i=0; i<noDrivers; i++) {**

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

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

**String firstName = stripQuotes(temp[0]);**

**String surname = stripQuotes(temp[1]);**

**dateFormat = new SimpleDateFormat("yyyy-MM-dd");**

**String dateOfBirthStr = stripQuotes(temp[2]);**

**Date dateOfBirthDate;**

**try {**

**dateOfBirthDate = dateFormat.parse(dateOfBirthStr);**

**dateOfBirth = Calendar.getInstance();**

**dateOfBirth.setTime(dateOfBirthDate);**

**} catch (ParseException ex) {**

**Logger.getLogger(**

**PolicyController.class.getName()).log(Level.SEVERE,**

**null, ex);**

**}**

**newDriver = new Driver(firstName, surname, dateOfBirth);**

**namedDrivers.add(newDriver);**

**}**

The code will be located between the try-catch for **policyStartDate** and the creation of the **policy** object – note you will need to replace the call to this constructor with one that includes the **namedDrivers** as an argument.

1. Uncomment the **addDriver()** and **deleteDriver()** methods.
2. Run the **CarInsuranceApp** class and test that the policy can be successfully loaded and drivers can be added and removed.

## Exercise 3

1. Open the **PolicyController** class and add the private method **storePolicyToTextFile()** as below:

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

**char DELIMITER=',';**

**try (PrintWriter output = new PrintWriter(fileName)) {**

**output.println(policy.toString(DELIMITER));**

**output.close();**

**} catch (FileNotFoundException ex) {**

**Logger.getLogger(**

**PolicyController.class.getName()).log(Level.SEVERE,**

**null, ex);**

**}**

**}**

Add a call to it when the Finish option is chosen from the menu:

**case 'F':**

**InputHelper inputHelper = new InputHelper();**

**char c = inputHelper.readCharacter(**

**"Store to a Text File (T) or Object File (O)?");**

**String fileName =**

**inputHelper.readString("Enter File name");**

**if (c == 'T') {**

**storePolicyToTextFile(fileName);**

**}**

**else if (c == 'O') {**

**// storePolicyToObjectFile(fileName);**

**}**

**finished = true;**

1. Run the **CarInsuranceApp** class and test that the policy can be successfully stored and re-loaded.

## Exercise 4

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

1. Open the **PolicyController** class and uncomment the **loadPolicyFromObjectFile()** method.
2. Uncomment the **storePolicyToObjectFile()** method.
3. Run the **CarInsuranceApp** class and test that a new policy can be created and stored as an object file and then successfully loaded and displayed.

**Note:** You should ensure that you understand the role of each class and the operation of each method.