WESTERN SYDNEY UNIVERSITY

SCHOOL OF COMPUTER, DATA & MATHEMATICAL SCIENCES
300581: Programming Techniques

Spring 2021: Programming Project

Due by 7:00pm on Friday 15th October 2021 **Assessment Weight: 40%**

A. Requirements

a) ALL instructions given in this document MUST be followed in order to be **eligible** for full marks for the assignment. This document has five (5) pages.
b) This assignment is **NOT** a group assignment; collusion, plagiarism, cheating of any kind is not acceptable. As part of your submission you MUST certify that all work submitted is your own. If you cannot honestly certify that the work is your own then do not submit the assignment. Breaches of the Misconduct Rule will be dealt with according to the university Rule (see the learning guide for more information).
c) All assignment submissions will be checked for academic misconduct by the use of the MOSS program from Stanford University. Details on MOSS can be obtained from the MOSS web site http://theory.stanford.edu/~aiken/moss/

For the problem definition described in section B you must

d) include your student id at the end of all filenames for all java code files. Two classes have been identified in section B as being required as part of your solution. Do not modify the names of these classes except for adding your student id to the end of the filename. Other Java files will be needed as part of your solution. All Java code files that are used in this assignment MUST have your student id appended to the filename. For example, Driver_#########.java;
e) include your authorship details at the top of **each** file in code comments (see item 3.1 in Section C of this document for details);
f) adhere to the coding standard as identified in the Google Java Style Guide (see Section C of this document for details);
g) ensure that standard console Input/Output are used in all code segments, **do not** use Swing;
h) ensure that your java code is appropriately modularised for the given problem definition. That is, you need to write appropriate classes and methods to solve the problem;
i) **reference** all sources that you used for inspiration of your solution as per Section D of this document;
j) Ensure that your java code compiles and runs in Eclipse installed in the SCDMS labs.

B. Project Details

B(i) - Background information and description

Callum's Air Conditioning Service (CACS) is a Sydney-based company that carries out repairs, services, and installations of residential air conditioning systems. The owner, Callum Murray, requires a Java program to assist his business keep accurate records of repair, service, and installation bookings. Each booking that is made by a customer is for one or more repairs, services, or the installation of an air conditioning system at their place of residence. Each booking is carried out on a designated date. Payment for the booking is required to be received from the customer on the date of the booking. Currently, the maximum number of bookings that can be made for any particular date is five (5); CACS currently own five (5) vans and employ ten (10) air conditioning technicians. CACS does not accept bookings for either Saturday or Sunday.

The specific program requirements are detailed in section *B(ii)* of this document. To perform the tasks identified in section *B(ii)* your program will need to read data from, and write data to, a small number of text files that are detailed in section *B(iii)*. The program will need to be Object-Oriented, utilising a small number of specified classes that are described in section *B(iv)*.

B(ii) - Program Requirements/Functionality

The Java program must be object-oriented utilising the classes described in section *B(iv)* **as a minimum**; other classes may also be needed to solve the problem requirements. Upon program execution, read the data from the customers and services files into **appropriate arrays or arraylists of objects** (see section B(iv) for descriptions of the relevant classes). The daily bookings file(s) will be read into memory only when needed by the relevant menu choice. Your program should not assume that the files exist on the file system. If any of the files don't exist, determine from the user if the files are in a different location or if they have different names, or if they wish to terminate the program; either load the files or terminate the program accordingly. After processing the relevant files, the program will display the main menu, which contains the following menu items:

1. Display Bookings
2. Make a Booking
3. Exit Program

Within the Java program implement the required functionality for each of these menu items as described below:

1. Display Bookings – The purpose of this menu option is to display all of today's bookings. When this menu option is selected the program should determine what the current date is then list on screen all bookings that are booked for the current date. When listing the bookings for the current date the following summary data needs to be displayed for each booking:

Date, Booking ID, Customer ID, Customer Surname, Service Code(s)

After displaying the list of bookings for the current date display the submenu:

1. Display Booking Detail
2. Change Display Day
3. Set Fee
4. Exit Sub-menu

Implement Java code for these sub-menu items as follows:

1. Display Booking Detail – when this menu item is selected the program should display the details for a chosen Booking ID. For this chosen booking, display

Date, Booking ID, Customer ID, Customer Surname, full Customer Address details, Service Code(s), Service Name(s), Service Price(s), and Total Fee

Ensure that this information is displayed in a sensible and easy to read format with appropriate subheadings and labels as needed.

2. Change Display Day **–** The program should allow the user to change the list of displayed bookings to a different date. Doing so will display the summary list of bookings for the newly chosen date.

3. Set Fee - when this menu item is selected the program should enable the user to set the fee for a chosen booking. The fee for the booking should be set to that entered by the user only if it is valid. In this context, valid means that the value entered must be a positive numeric value that is no more than 10% less than the **calculated** total fee (based upon the services for the booking) (ie, the **maximum** discount that can be applied is 10% of the calculated total fee). If the fee is not valid do not change its current value.

For example, if a booking has several services listed which, when calculated from the service information, totalled $500 then the minimum value which would be allowed for the total fee is 500 - 10/100 * 500 = 500 – 50 = 450

If the change to the fee for the booking is successful write the booking to the relevant booking file.

4. Exit Sub-menu – when this menu item is selected program control should return to the main menu.

2. Make Booking – The purpose of this menu item is to be able to make a booking for service, repair, or installation. To do this display the following sub-menu:

1. By Phone Number
2. By Surname
3. Exit Sub-menu

Implement Java code for these sub-menu items as follows:

1. By Phone Number – when this menu option is selected the program should perform a search for the phone number of the customer. For a matching phone number, the user should then be prompted for the date that the customer wishes to make a booking for. **Note**: before allowing the booking, the program must check if there are any bookings available for the chosen date. If the chosen date is available, then

determine from the user which service(s) are required and then make the booking by generating a new Booking for it and storing the Booking in the appropriate array/arraylist. The new booking should also be written to the relevant booking file. **Note**: the case of adding a new customer is not covered in this project so if no matches for the phone number are found simply display an appropriate message for the user and return program control to the sub-menu.

2. By Surname **-** when this menu option is selected the program should perform a search for the customer surname. The program should display any matches found such that the correct customer can be identified (note: there may be more than one match when searching by surname). If the customer can be identified from the search results, then the user should indicate the date that the customer wishes to make a booking for. **Note**: before allowing the booking the program must check if there are any bookings available for the chosen date. If the chosen date is available, then determine from the user which service(s) are required; then make the booking by generating a new Booking and storing the Booking in the appropriate array/arraylist. The new booking should also be written to the relevant booking file. **Note**: the case of adding a new customer is not covered in this project so if no matches for the customer surname are found simply display an appropriate message for the user and return program control to the sub-menu.

3. Exit sub-menu – when this menu item is selected program control should return to the main menu.

3. Exit Program – the program should terminate when this menu item is selected. The program should warn the user if any changes have been made to any bookings and allow them to save the files if they choose to before exiting the program.

B(iii) - Text files to be processed

The data that is to be manipulated by your Java program is contained in the text files **Customers.txt, Services.txt**, and **bookings-yyyy-mm-dd.txt.** Examples of these files are found in the zip file for this project. The data within these text files will need to be read into memory by your program so that it may be manipulated to solve many aspects of the required functionality of this programming project. The text files have been created to conform to a particular format. The format for each file is described below:

File: Customers.txt

This file contains a full record of all customers that at one time or another have had their air conditioning serviced or installed by CACS. Each line within the file represents a customer, and has the following format:

Customer ID, Surname, First name, Phone, Address, Suburb, Postcode

where each data item is separated by a comma (,).

A brief explanation of each of these data items:

Customer ID:	a unique numeric identifier for a customer
Surname:	customer surname
First name:	customer first name
Phone:	customer phone number
Address:	customer residential street address
Suburb:	customer residential suburb
Postcode:	customer residential postal code; numeric

File: Services.txt

This file contains a full record of the different services that CACS can provide for air conditioning systems. In this context, service means either repair, service, or installation. Each line within this file represents an individual service, and has the following format:

Service Code, Name, Description, Price

where each data item is separated by a comma (,).

A brief explanation of each of these data items:

Service Code:	a unique numeric identifier for the service
Name:	name of the service
Description:	description of the service
Price:	normal price for service or repair (before any discounts or surcharges)

File: bookings-yyyy-mm-dd.txt
There are multiple Bookings files. Each file represents the bookings that have been booked in for a specific date. The date of booking is incorporated in the filename for easy identification and processing by the program. For example, bookings for the date 21/09/2021 are in the file bookings-2021-09-21.txt. Each line within each file represents a booking, and has the following format:

Booking ID, Customer ID, Date, Total Fee, Service Code(s)

where each data item is separated by a comma (,).

A brief explanation of each of these data items:

Booking ID:	a unique numeric identifier for the booking
Customer ID:	the customer ID for this booking
Date:	date when CACS carries out the booking; in the format dd/mm/yyyy
Total Fee:	total price for the Booking that has been quoted to the customer
Service Code(s):	the service code(s) to be carried out at the residence for the booking

Note: for the purpose of marking the assignment the number of lines of data and the data values in the text files will be replaced with different data by the marker. This is to ensure that your solution has not relied upon specific data values or the number of lines in the text files to work. You should therefore test your program with different data files before submission.

B(iv) - Required Classes

To write your solution for this assignment it is a **requirement** that you write appropriate code for the following java Classes.

- Customer
- Service
- Booking

These classes are described in general terms as follows:

- **Customer class**: the Customer class represents an ***individual*** CACS customer. The Customer class needs to store data for the customer ID (integer), customer surname (string), customer first name (string), customer residential street address (string), the customer residential suburb (string), and the customer residential postcode (string), customer phone number (string). As well as the normal methods that should be created for a class (eg, constructors, mutators, and accessors) you will need to decide upon other ***appropriate*** methods for this class based upon the general requirements of the assignment specification.

- **Service class**: the Service class represents an ***individual*** service that CACS can perform. The Service class needs to store data for the service code (integer), service name (string), service description (string), and the price of the service (float). As well as the normal methods that should be created for a class (eg, constructors, mutators, and accessors) you will need to decide upon other ***appropriate*** methods for this class based upon the general requirements of the assignment specification.

- **Booking class**: the Booking class represents an ***individual*** booking. The Booking class needs to store data for the Booking identification number (integer), the customer ID (integer), a list of service Codes for the services that are to be carried out (integer array or arraylist), the total price that was quoted for the booking (float), the date that the booking will be carried out (Date or string).

These classes **must** be incorporated into your solution. It is likely that you **may also need** to write other classes depending upon your solution method.

C. Google Java Style Guide

The submission in this assignment must adhere to the following listed coding standards as defined in the Google Java Style Guide that is found at https://google.github.io/styleguide/javaguide.html

Style Guide Item Number	Changes to	Modification
2.1 to 2.3	**2.1 modified**	2.1 - File Name: The source file name consists of the case-sensitive name of the top-level class it contains as identified in the question, plus an underscore, plus the student ID, plus the .java extension Example: Driver_12345678.java
3.1 to 3.4.1	**3.1 modified**	3.1 – License Info is replaced by **Authorship** information All java source files must contain the authorship information as follows: **Student ID:** **Name:** **Campus:** **Tutor Name:** **Class Day:** **Class Time:**
4.1 to 4.7, 4.8.2.1 to 4.8.2.3, 4.8.4 to 4.8.4.3, 4.8.6	**NIL**	
5, 5.1 to 5.3	**NIL**	

D. Referencing

Referencing must follow the guidelines given in Section 2.5.1 of the unit Learning Guide. An example implementation of this referencing style can be found in the FAQ in the Programming Techniques vUWS site.

E. Project Submission Procedure

To submit your **project** you must do the following by the due date and time specified on page 1 of this document:

1. Create a zip file which contains
    a. your **complete Java project** including the Java source code file(s).

    Note: The zip file must be named according to the naming convention

    *studentid*_*StudentName*_300581_Project_*TutorsName*.zip

    where *studentid* is your student id, and *StudentName* is your full name, *TutorsName* is the name of your tutor.
2. Upload the above zip file in vUWS in the **Project Submission** link provided.

F. Marking Criteria and Standards

The marking criteria and standards for the assignment are published in **section 2.5.2 in the Learning Guide** and will be used to assess your assignment submission according to the specific weightings identified in the table below

Code Functionality/Correctness:	55%	Code Documentation:	5%
Class Construction	20%	Identifier Use:	5%
Algorithm Selection:	10%	Code Readability:	5%