WESTERN SYDNEY UNIVERSITY

SCHOOL OF COMPUTING, ENGINEERING & MATHEMATICS
300581: Programming Techniques

Spring 2020: Programming Project

Due by 7:00pm on Friday 23rd October 2020 **Assessment Weight: 50%**

A. Requirements

a) This assessment has a threshold requirement of 40%. That is, you must achieve a mark of at least 40% (20 out of 50) to be eligible for a passing grade in this unit.
b) ALL instructions given in this document MUST be followed in order to be **eligible** for full marks for the assignment.
c) 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).
d) All assignment submissions will be checked for academic misconduct by the use of the MOSS program from Stanford University (http://theory.stanford.edu/~aiken/moss/).

For the problem definition described in section B you must

e) Include your student ID at the end of all filenames for all java code file. Four 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, Client_########.java.
f) 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).
g) Adhere to the coding standard as identified in the Google Java Style Guide (see Section C of this document for details).
h) Ensure that standard Input/Output are used in all code segments, **do not** use Swing.
i) 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.
j) **Reference** all sources that you used for inspiration of your solution as per Section D of this document;
k) Ensure that your java code compiles and runs in Eclipse 2020-06

B. Project Details

B(i) - Background information and description

In real estate, a “property manager” is responsible for renting and maintaining properties owned by clients. For providing this service, property managers charge the client a fee. For this assignment, we assume the fee is calculated as a percentage of the rental income.

The property manager needs to record income (i.e. rent collected) and expenses (i.e. maintenance costs and monthly fees) for each property. If a client owns more than one property, their entire collection of properties is known as a “portfolio”. The property manager also needs to be able to generate portfolio reports for clients, which provide a summary of income, fees, and expenses for all the properties in that client’s portfolio.

In this assignment, your task is to create an object-oriented, menu-driven Java program that implements a limited set of functionalities (i.e., the program will not be a completely real-world implementation) that a property manager can use to record rental income and expenses for each client’s property, generate portfolio reports, and retrieve and save data in secondary storage.

In general, your program will need to read data from the keyboard and from certain text files in secondary storage, store the data in appropriate data structures using objects, sort and search the data, and write output data to both the screen and to secondary storage. The specific functional requirements are described in section B(ii) of this document. The text files that are to be used for this assignment are described in section B(iii). The classes that **must** be used **as a minimum** are described in section B(iv).

B(ii) - Program Requirements/Functionality

The Java program **must**

a) be object-oriented utilising the classes described in section B (iv) **as a minimum**. Other classes may also be needed to solve the program requirements;

b) be menu-driven. The main menu must have the following menu items:

1. Record Rent Collection.
2. Record Expense.
3. Generate Portfolio Report.
4. Save.
5. Exit Program.

c) be able to process the defined text files. The text files and their formats are described in section B (iii).

Program Start Up

When the java program starts it must perform the following file related operations:

d) Read the data from the **clients.txt** file into computer memory into an appropriate array/arraylist of objects (see section B (iii) for a description of the clients.txt file and section B (iv) for a description of the Clients class). If the clients.txt file does not exist, then the user should be informed of this and given the opportunity to provide an alternative filename that contains the client data;

e) Read the data from the **properties.txt** file into computer memory into an appropriate array/arraylist of objects (see section B (iii) for a description of the properties.txt file and section B (iv) for a description of the Properties class). If the properties.txt file does not exist, then the user should be informed of this and given the opportunity to provide an alternative filename that contains the property data;

f) Read the data from the **expenses.txt** file into computer memory into an appropriate array/arraylist of objects (see section B (iii) for a description of the expenses.txt file and section B (iv) for a description of the Expenses class). If the expenses.txt file does not exist, then the user should be informed of this and given the opportunity to provide an alternative filename that contains the expenses data;

g) Read the data from the **rents.txt** file into computer memory into an appropriate array/arraylist of objects (see section B (iii) for a description of the rents.txt file and section B (iv) for a description of the Rents class). If the rents.txt file does not exist, then the user should be informed of this and given the opportunity to provide an alternative filename that contains the rents data.

After processing these four files the program should display the main menu identified in section B(ii)(b) above.

Main Menu Item Functionality

The required functionality for each menu item is described as follows:

1. Record Rent Collection - when this menu option is selected the following actions should be performed by the program:

- Allow the user to find the property for which the rent was collected by entering the property's address.
  - The user should not have to type the entire address. For example, if the user typed "Smith St", the search would return all and any properties whose address included "Smith St". If a search returns multiple results (for example, there is more than one property at Smith St), the program must list on the screen all matching properties so that the user may choose a property from the list.
- Display the chosen property's address, weekly rent charged, and property owner's full name.
- The program should ask the user how many weeks of rent was collected for the chosen property.

- Calculate the amount of rent collected based upon the weekly rent rate and the number of weeks entered by the user.
- Record the new rent collection using today's date in the appropriate array/arraylist.
- Generate an on-screen summary of the rent collection transaction. The summary must show the property address, the monetary amount, the number of weeks rent collected, the property owner's name, and the current date. Make sure that this is displayed in a clear and logical format on the screen.

Ensure that the program enforces **all sensible** validation conditions. For example, the number of weeks rent entered by the user must be greater than zero.

2. Record Expense - when this menu option is selected the following actions should be performed by the program:

- Allow the user to find the property for which the expense was incurred by entering the property's address. As per Record Rent Collection, allow the user to search for the property using a partial address.
- Display the property's address, weekly rent charged, and property owner's full name.
- Allow the user to record an expense, which must include the current date, a monetary amount greater than 0, and a description of the expense. Add the new expense to the appropriate array/arraylist.
- Generate an on-screen summary of the expense. The summary must show the property address, the monetary amount, the description of the expense, the property owner's name, and the current date. Make sure that this is displayed in a clear and logical format on the screen.

Ensure that the program enforces all sensible logical validation conditions when doing so, such as ensuring the user enters a description for the expense, and the cost of the expense is greater than zero.

3. Portfolio Report - when this menu option is selected the following actions should be performed:

- Ask the user if the report is for all clients, a specific client, or for all properties in a specific postcode.
- Generate the Portfolio Report(s) for the user's choice:
  - If the report is for a specific client, the user should be able to search for the client by their name. For example, if the user typed "Chris", the search would return any and all clients whose name contained "Chris" and produce a portfolio report for that/those client/s.
  - If the report is for all clients, produce the portfolio report for all clients outputting the results sorted in ascending order of client last-name.
  - If the report is for a specified postcode, obtain the postcode from the user, then produce the report for all properties that are located in the entered postcode.
- The Portfolio Report displays an on-screen summary of the rent, expenses and fees for all properties owned by the client.
- The Portfolio Report should have a similar layout to the following example:

```
PORTFOLIO REPORT
Client: Steve Austin, 3 Doris St North Sydney NSW 2060
Report Generated: Tue Sep 22 14:11:18 AEST 2020

-------------------------------------------------------------------------------------------------
|                            Property |       Rent |   Expenses | Fee Rate |    Fees |       Net |
-------------------------------------------------------------------------------------------------
|   567 Elizabeth St Redfern NSW 2016 |       0.00 |     199.90 |     0.05 |    0.00 |   -199.90 |
|      168 Lawson St Redfern NSW 2016 |    3200.00 |     300.00 |     0.08 |  256.00 |   2644.00 |
-------------------------------------------------------------------------------------------------
|                              TOTAL |    3200.00 |     499.90 |          |  256.00 |   2444.10 |
-------------------------------------------------------------------------------------------------
```

Notes:

- If no rent or expense records can be found for the client portfolio, inform the user that no rental or expense records can be found for the client.
- The "Rent" column should display the total amount of rent collected for each property the client owns.

- The “Expenses” column should display the total amount of expenses incurred for each property the client owns.
- The “Fee Rate” column should display the correct rate for each property.
- The “Fees” column should be calculated by multiplying the rent collected by the fee rate.
- The “Net” column is calculated by subtracting expenses and fees from the rent collected.
- The last row should display totals for all numeric columns except “Fee Rate”.
- Your table columns should be neatly aligned. Currency values should be printed to two decimal places. There are multiple ways to do this, such as using “System.out.printf” and “System.out.format”.

Ensure that the program enforces all sensible logical validation conditions when doing so, such as ensuring the user enters a postcode that is 4 digits.

4. Save – All rent and expense information entered by the user via the menu items “Record Rent Collection” and “Record Expense” must be saved to the files rents.txt and expenses.txt respectively. Pre-existing data in rents.txt and expenses.txt must be preserved. Clients.txt and properties.txt should not be changed.

5. Exit Program – the program must terminate when this menu item is selected. **The program should not terminate until this option is chosen**. *If the rent or expense data has changed since the last save, then **do not immediately exit the program***. Instead, warn the user that they have unsaved changes, and give them the option to either (1) return to the main menu or (2) exit the program.

B(iii) - Text files to be processed

The data that is to be manipulated by your Java program for this assignment is contained in the text files **clients.txt, properties.txt, expenses.txt** and **rents.txt.** Examples of these text files are found in the zip file for the assignment. 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 the assignment. The text files have been created to conform to a particular format. The format for each file is described below:

File: clients.txt

This file contains a list of all clients managed by the property manager.
Each line within the file represents an individual client, and has the following format:

Client ID, Client Name, Street Address, Suburb, State, postcode

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

A brief explanation of each of these data items:

Client ID:	a unique numeric identifier for a client
Client Name:	the client’s name in the format: first-name last-name
Street Address:	the address of the client to which correspondence should be mailed.
Suburb:	the suburb for the client
State:	the state for the client
Post code:	the postcode for the client

File: properties.txt

This file contains a list of all properties managed by the property manager.
Each line within the file represents an individual property, and has the following format:

Property ID, Street Address, Suburb, State, postcode, Weekly Rent, Management Fee, Client ID

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

A brief explanation of each of these data items:

Property ID:	a unique numeric identifier for a property
Street Address:	the address of the property

Suburb: the suburb for the property
State: the state for the property
Post code: the postcode for the property
Weekly Rent: the rental amount charged for the property per week in dollars
Management Fee: the percentage of the weekly rent claimed as a fee for managing this property.
Client ID: the client ID of the property's owner.

File: expenses.txt
This file contains a list of all expenses for all properties.
Each line within the file represents an individual expense, and has the following format:

Property ID, Expense Description, Expense Amount, Date

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

A brief explanation of each of these data items:
Property ID: the unique numeric identifier of the property for which the expense was incurred
Expense Description: a description of the expense, e.g. "fix leaking tap"
Expense Amount: the cost of the expense
Date: the date on which the expense was incurred in yyyy-mm-dd format

File: rents.txt
This file contains a list of all rental income for all properties.
Each line within the file represents an individual rent collection event, and has the following format:

Property ID, Rent Amount, Date

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

A brief explanation of each of these data items:
Property ID: the unique numeric identifier of the property for which the rent was collected
Rent Amount: the monetary amount of rent collected
Date: the date on which the rent was collected in yyyy-mm-dd format

Notes:
- When reading the text files into memory, the data should be read into **appropriate** arrays/arraylists of objects. The classes for these objects are briefly outlined in section B(iv).
- 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 of varying length and varying data before submission.

B(iv) - Required Classes
To write your solution for this assignment it is a **requirement** that you write appropriate code for **at least** the following java Classes:
a) Client
b) Property
c) Expense
d) Rent

These classes are described in general terms as follows:
a) **Client class**: The Client class represents *an individual* client (customer). The Client class needs data fields for the **client ID**, **first name**, **surname**, and **address (the address consists of street, suburb,**

state, post code). Implement appropriate constructors, accessors, and mutators where necessary and other appropriate methods for this class based upon the general requirements of the assignment specification – that is, you will need to identify if the Client class is required to perform any other actions and implement the identified methods in the class.

b) **Property class**: The Property class represents a single property that *an individual* Client owns. The Property class needs data fields for the **property ID**, **property address (the address consists of street, suburb, state, post code)**, the **weekly rent**, the **management fee**, and the **client ID** of the property's owner. Implement appropriate constructors, accessors, and mutators where necessary and other appropriate methods for this class based upon the general requirements of the assignment specification – that is, you will need to identify if the property class is required to perform any other actions and implement the identified methods in the class.

c) **Expense class**: The Expense class represents *an individual* Expense incurred for a property. The Expense class needs data fields for the **property ID**, the **expense description**, the **expense amount**, and the **date** on which the expense was incurred. Implement appropriate constructors, accessors, and mutators where necessary and other appropriate methods for this class based upon the general requirements of the assignment specification – that is, you will need to identify if the Expense class is required to perform any other actions and implement the identified methods in the class.

d) **Rent class**: The Rent class represents *an individual* rent collection for a property. The Rent class needs data fields for the **property ID**, the **rent amount**, and the **date** on which the rent was collected. Implement appropriate constructors, accessors, and mutators where necessary and other appropriate methods for this class based upon the general requirements of the assignment specification – that is, you will need to identify if the rent class is required to perform any other actions and implement the identified methods in the class.

Apart from the above four classes it is possible that you **will 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/styleguid/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: Expense_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 on page 8 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 the **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. All of your project **Java source code files** (the **.java** files are located in the **`src`** folder of your project)

    Do not include folders in the zip file. We just want the `.java` files for the project placed in the root of the zip file.

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

    *studentid_StudentName*_300581_Project.zip

    where *studentid* is your student ID, and *StudentName* is your full name.

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 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:	50%	Code Documentation:	5%
Class Construction	20%	Identifier Use:	5%
Algorithm Selection:	15%	Code Readability:	5%