ITECH1000 / ITECH5000 Programming 1 Semester 1 2017

Assignment 1

Overview

This is an individual assignment that requires you to design, develop and test a small procedural Java program.

Timelines and Expectations

Percentage Value of Task: 20%

Due: 16:00 Friday Week 7 (refer to course description for your campus for the exact date)

Minimum time expectation: 10 hours

Learning Outcomes Assessed

The following course learning outcomes are assessed by completing this assessment:

- Identify and use the correct syntax of a common programming language
- Recall and use typical programming constructs to design and implement simple software solutions
- Reproduce and adapt commonly used basic algorithms
- Utilise pseudocode and/or algorithms as a major program design technique
- Write and implement a solution algorithm using basic programming constructs
- Demonstrate debugging and testing skills whilst writing code
- Develop self-reliance and judgement in adapting algorithms to diverse contexts
- Design and write program solutions to identified problems using accepted design constructs

Assessment Details

Your task is to design, develop and test a small application which will allow a mobile phone user to compare the cost of their phone usage under plans from two different phone providers.

Stage 1: Design

This stage requires you to prepare documentation that describes the function of the program and how it is to be tested. There is no coding or code testing involved in this stage. A document template has been provided for your use.

Requirements:

1) Read through *Stage 2: Program Development* to obtain details of the requirements of this program.
2) Write an algorithm that describes how the program will operate.
    a. All program requirements – base, standard and advanced – must be included, even if you do not end up including all these requirements in your program code.
    b. The algorithm must be structured logically so that the program would function correctly.
3) Prepare and document test cases that can be used to check that the program works correctly, once it has been coded. You do NOT need to actually run the test cases in this stage; this will occur in *Stage 3: Testing*.
    a. All program requirements – base, standard and advanced, must be included, even if you do not end up including all these requirements in your program code.
    b. Make sure the test cases include checking of data entered by the user to make sure that only valid data is accepted. If the user enters invalid data, the user should be informed of this and given another chance to enter the data. *NB: As we have not covered exception handling, you may assume that the user will always enter a value of the correct type (i.e. an integer when an integer is requested).*
    c. Test cases should be documented using a template like the one below. You may include extra information if you wish. At this stage, the Actual Result column will be left blank.

Test Case	Expected Result	Actual Result
The user enters a negative number when prompted to enter the number of MB of data used: `Enter the amount of data in MB: -5`	The user should be informed of the error and prompted to re-enter a positive number: `Value must be positive. Please try again:`	

Stage 2: Program Development

Using the Design Documentation to assist you, develop a Java program that allows the user to enter details of their phone usage and then compare the bill which would result from this usage under different billing plans. These requirements have been broken into three groups:

- *Base Functionality* includes the minimal level of requirements to achieve the essential components of this assignment. This group of requirements focuses on getting the code to work and on using the programming constructs we cover in class. You can expect to use constants, variables, loops, conditional statements and arithmetic operators for these requirements and you should look for opportunities to use these wherever you can. You will not receive full marks for implementing a requirement, even if it works, if you have not used the appropriate programming construct to do so.

  At this level, you can decide if it is easier for you to code everything within a single method, or to modularize it straight away.

- *Standard Functionality* ensures the code is modularized, and that method calls are used to ensure the program flows correctly. It allows data to pass from one method to another as parameters. It also includes implementing one additional billing plan which is slightly more complex than the two plans used in the Base Functionality.

- *Advanced Functionality* provides a challenge task, and is best left until all the other requirements have been addressed. It requires looking at a Java API to find out how to use a class we have not covered in the course, and using this to apply a discounting to phone calls made on a weekend.

All three groups require that you follow coding conventions, such as proper layout of code, using naming conventions and writing meaningful comments throughout your program.

Base Functionality:

1. Display a welcome message when the program starts
   - The welcome message should have a row of equals signs (=) at the top and the bottom, just long enough to extend over the text. *Hint: Use a For loop for this.*
   - The first line of the message should read “FEDERATION UNIVERSITY PHONE BILL COMPARISON SYSTEM” and be approximately centered in the row of equals signs by printing white space first. *Hint: Can you modify the For loop from the previous step to print the white spaces?*
   - The second line of the message should be blank.
   - The third line should read “Developed by” followed by your name and a comma, then “ student ID ”, then your student id, then finally “ for ITECH1000/5000 Sem 1 2017”.
   - The fourth line should be blank, and the fifth should be another row of ====

```
================================================================================
            FEDERATION UNIVERSITY PHONE BILL COMPARISON SYSTEM

Developed by Peter Vamplew, student ID 19051251 for ITECH1000 Semester 1 2017


================================================================================
```

2. Provide a menu from which the user can select to Enter Usage Details, Display Cost Under Plan A, Display Cost Under Plan B, Clear Usage, or Exit System. This menu should be repeated each time after the user has chosen and completed an option until the user chooses to Exit. The user selects an option by entering the number next to it. If an invalid number is selected, the user is advised to make another selection.

```
MAIN MENU
Please select an option from the menu:
1. Enter Usage Details
2. Display Cost Under Plan A
3. Display Cost Under Plan B
4. Clear Usage Details
5. Exit System
6
Value must be between 1 and 5. Please try again:
0
Value must be between 1 and 5. Please try again:
1
```

3. When the user selects the Enter Usage Details option, provide another menu from which the user can select Phone Call, SMS, Data Usage, or Return to Main Menu. The user selects an option by entering the number next to it. If an invalid number is selected, the user is told to make another selection.

```
ENTER USAGE DETAILS MENU
Please select an option from the menu:
1. Phone Call
2. SMS
3. Data Usage
4. Return to main menu
```

   a. Should the user select Phone Call, they are prompted to enter the length of the call in seconds. The value entered must be positive – if not, the user should be prompted to re-enter a new value. After entering a valid call length, the user is returned to the Enter Usage Details Menu so that they may choose to enter additional usage details.

```
1
Enter call length in seconds:
0
Value must be positive. Please try again:
27
ENTER USAGE DETAILS MENU
Please select an option from the menu:
1. Phone Call
2. SMS
3. Data Usage
4. Return to main menu
```

   b. Should the user select SMS, the program should simply increment the count of the number of SMS messages. No further information is required so the program should

simply display the total number of SMS messages recorded so far, and then return to the Enter Usage Details Menu.

```
2
Total number of SMS so far = 1
```

c. Should the user select Data Usage, they should be prompted to enter the amount of data used in MB (it will be assumed that this is an integer value). The value entered must be positive – if not, the user should be prompted to re-enter a new value. After entering a valid value, the user is returned to the Enter Usage Details Menu so that they may choose to enter additional usage details.

```
3
Enter the amount of data in MB:
-5
Value must be positive. Please try again:
120
```

4. When the user selects the Display Cost Under Plan A option, the program should display a summary of the usage details which have been entered, and their cost under Plan A, along with the total cost, formatted as shown in the screenshot below. The cost structure for Plan A is listed in the following table. Once the bill summary has been displayed, the program should return to the Main Menu.

Usage Item	**Item Cost – Plan A**
Per phone call (flag fall charge)	$0.23
Per second of total time over all phone calls	$0.02
Per SMS	$0.12
Per MB of data usage	$0.03

```
Cost under Plan A
=======================================
Number of calls = 3             $0.69
Total call time (secs) = 202    $4.4
Number of SMS = 3               $0.36
Data usage (MB) = 120           $3.60
=======================================
TOTAL COST                      $8.69
```

To assist in formatting the display of the costs in this step, you may copy the following method into your code and make use of it.

```
// Print the specified value in cents in dollar and cents format
private static void displayAsDollarsAndCents(int cents)
{
     System.out.print("$" + (cents/100) + "." + (cents%100));
}
```

5. When the user selects the Display Cost Under Plan B option, the program should do the same as in Step 4, but using Plan B's cost structure instead, which is listed in the following table, and then return to the Main Menu.

Usage Item	Item Cost – Plan B
Per phone call (flag fall charge)	$0.17
Per second of total time over all phone calls	$0.03
Per SMS	$0.15
Per MB of data usage	$0.02

```
Cost under Plan B
=========================================
Number of calls = 3              $0.51
Total call time (secs) = 202     $6.6
Number of SMS = 3                $0.45
Data usage (MB) = 120            $2.40
=========================================
TOTAL COST                       $9.42
```

6. When the user selects Clear Usage Details the value of all variables related to the usage (number of calls, total length of calls, number of SMS, total data usage) should all be reset to 0. A message reporting this should be displayed as shown below (including the rows of equals signs), and the program should return to the Main Menu.

```
=====================================================================
ALL USAGE DETAILS HAVE BEEN RESET TO 0
=====================================================================
```

7. When the user selects Exit System, quit the program with a message to the user.

```
================================================================
Thank you for using this software. We hope you found it useful.
================================================================
```

Standard Functionality:

8. Add an additional option to the Main Menu to Display Cost Under Plan C. When this option is selected, the program should display the cost under this Plan in the same way as previously done for Plans A and B, but using Plan C's pricing structure as shown below. Note that Plan C charges data at a different rate based on whether the user has exceed the cap of 100MB or not. Usage up to and including the first 100MB is charged at the lower rate; any further usage above 100MB is charged at a higher rate.

Usage Item	Item Cost – Plan C
Per phone call (flag fall charge)	$0.11
Per second of total time over all phone calls	$0.03
Per SMS	$0.09
Per MB of data usage up to 100MB	$0.02
Per MB of data usage over 100MB	$0.10

9. Modularize the code, correctly using method calls and passing data between methods as parameters.

Advanced Functionality:

10. All the companies have introduced a new policy that phone calls are not timed if they are made on weekends (only the flagfall charge is applied). Modify the code from Stage 3, so that when the user selects the Enter Usage Details option, they are first prompted to enter a date (as a day, then a month, then a year). The program should check whether this data is on a weekend, and if so there is no need to ask the user to enter the length of the call when the Phone Call option is selected (note that the flagfall charge is still incurred).

    a. Use the java.util.GregorianCalendar class to store the data entered by the user.

    b. If you have a GregorianCalendar object called g, you can obtain a number representing the day of the week by calling

    ```
    g.get(GregorianCalendar.DAY_OF_WEEK);
    ```

    This returns a value from 1 to 7 representing the day of the week corresponding to the current data stored in the GregorianCalendar object. 1 is Sunday and 7 is Saturday.

Stage 3: Testing

Using a copy of the test cases developed in *Stage 1: Design*, test the program you have developed in *Stage 2: Program Development*. Document your results, including both failed and successful tests.

Note: Please do not leave out any failed tests. If your testing highlights that your program has not worked correctly, then the failed tests help to demonstrate that you have been testing your program properly.

To show that you have tested your program, include small (but readable) screen captures in your Actual Results as well as any explanatory comments. Microsoft Windows includes a Snipping Tool that is useful for taking captures of the relevant parts of the screen.

You may test your program in phases if you wish. *Stage 2: Program Development* provides three separate groups of functionality in the requirements, working from the minimal level of requirements through to more advanced concepts. Each of these groups can be tested individually.

Base Functionality:

This phase requires you to check that the base functions (displaying welcome message when the program starts, showing a menu of options until the user chooses to exit, entering and clearing usage details, viewing bills under Plans A and B, and exiting the system) have been implemented.

Standard Functionality:

In addition to the Base Functionality, this section includes implementing viewing of bills under Plan C. This new functionality must also be tested.

If you originally wrote and tested the Base Functionality by including all the code in a single method, this phase also requires that you modularize your code, and use method calls and pass data between methods to ensure your program still runs correctly. This means all your code will need to be re-tested to ensure that all the previous functionality still works.

Advanced Functionality:

This phase requires testing the code that gets a date from the user, which tests if that date is on the weekend, and which ignores the length of any calls made on the weekend.

Submission

Your program code and testing documentation should be zipped into a single file and loaded into the Assignment Box provided in Moodle by the due date and time.

Marking Criteria/Rubric

Task	Available Marks	Student Mark
Stage 1: Design Documentation		
Development of an algorithm describing how the program should function		
• All requirements from the Assessment Details section included	1	
• Logical structure	1	
Documented test cases to validate program		
• All requirements from the Assessment Details section included	2	
• Data is validated to ensure user entries are appropriate and incorrect user entries are handled smoothly	1	
Stage 2: Program Development		
A Java program addressing the requirements outlined in the Assignment Details section, including appropriate use of loops, conditional statements, constants and variables:		
Use of coding conventions throughout entire program, including readable and clear layout, following naming conventions and including meaningful and appropriate comments.	2	
Base Functionality:		
• Display of a welcome message when the program starts	1	
• Menu displayed until user chooses to exit	1	
• Enter Usage Details option fully implemented	3	
• Ability to view bills under Plans A and B	2	
• Clear Usage and Exit System to quit the program with a message to the user	2	
Standard Functionality:		
• Plan C implemented	1	
• Code modularized, correctly using method calls and passing data between methods	2	
Advanced Functionality:		
• Untimed phone calls on weekends	3	
Stage 3: Testing		
Documented test results clearly showing all the testing that has been conducted and the results of this testing.		
• Testing of base functionality	1	
• Testing of standard functionality	1	
• Testing of advanced functionality	1	
Total	25	
Scaled to 20% of course mark	20	

Feedback

Assignments will be marked within 2 weeks of submission. Marks will be loaded in fdlGrades, and a completed marking sheet will be available via Moodle.

Plagiarism:

Plagiarism is the presentation of the expressed thought or work of another person as though it is one's own without properly acknowledging that person. You must not allow other students to copy your work and must take care to safeguard against this happening. More information about the plagiarism policy and procedure for the university can be found at http://federation.edu.au/students/learning-and-study/online-help-with/plagiarism.