



<b>MODULE NAME:</b>	<b>MODULE CODE:</b>
<b>PROGRAMMING 1B</b>	<b>PROG6112</b>

**ASSESSMENT TYPE: ASSIGNMENT 1 (PAPER ONLY)**

**TOTAL MARK ALLOCATION: 100 MARKS**

**TOTAL HOURS: 15 HOURS**

*By submitting this assignment you acknowledge that you have read and understood all the rules as per the terms in the registration contract, in particular the assignment and assessment rules in The IIE Assessment Strategy and Policy (IIE009), the intellectual integrity and plagiarism rules in the Intellectual Integrity Policy (IIE023), as well as any rules and regulations published in the student portal.*

**INSTRUCTIONS:**

1. ***No material may be copied from original sources, even if referenced correctly, unless it is a direct quote indicated with quotation marks. No more than 10% of the assignment may consist of direct quotes.***
2. ***No assignment with a similarity index of more than 25%, even if the sources are referenced correctly, will be accepted.***
3. ***Make a copy of your assignment before handing it in.***
4. ***Assignments must be typed unless otherwise specified.***
5. ***All work must be adequately and correctly referenced.***
6. ***Follow all instructions on the assignment cover sheet.***
7. ***This is an individual assignment.***

## Referencing Rubric

Providing evidence based on valid and referenced academic sources is a fundamental educational principle and the cornerstone of high-quality academic work. Hence, The IIE considers it essential to develop the referencing skills of our students in our commitment to achieve high academic standards. Part of achieving these high standards is referencing in a way that is consistent, technically correct and congruent. This is not plagiarism, which is handled differently.

Poor quality formatting in your referencing will result in a penalty of **a maximum of ten percent being deducted from the percentage awarded**, according to the following guidelines. Please note, however, that **evidence of plagiarism in the form of copied or uncited work (not referenced), absent reference lists, or exceptionally poor referencing, may result in action being taken in accordance with The IIE's Intellectual Integrity Policy (0023).**

Markers are required to provide feedback to students by indicating **(circling/underlining) the information that best describes the student's work.**

**Minor technical referencing errors: 5% deduction from the overall percentage** – the student's work contains **five or more errors** listed in the minor errors column in the table below.

**Major technical referencing errors: 10% deduction from the overall percentage** – the student's work contains **five or more errors** listed in the major errors column in the table below.

**If both minor and major errors** are indicated, then 10% only (and not 5% or 15%) is deducted from the overall percentage. The examples provided below are not exhaustive but are provided to illustrate the error.

<b>Required:</b> Technically correct referencing style	<b>Minor errors in technical correctness of referencing style</b> Deduct 5% from percentage awarded	<b>Major errors in technical correctness of referencing style</b> Deduct 10% from percentage awarded
<u>Consistency</u>  <ul style="list-style-type: none"> <li>The same referencing format has been used for all in-text references and in the bibliography/reference list.</li> </ul>	Minor inconsistencies. <ul style="list-style-type: none"> <li>The referencing style is generally consistent, but there are one or two changes in the format of in-text referencing and/or in the bibliography.</li> <li>For example, page numbers for direct quotes (in-text) have been provided for one source, but not in another instance. Two book chapters (bibliography) have been referenced in the bibliography in two different formats.</li> </ul>	Major inconsistencies. <ul style="list-style-type: none"> <li>Poor and inconsistent referencing style used in-text and/or in the bibliography/ reference list.</li> <li>Multiple formats for the same type of referencing have been used.</li> <li>For example, the format for direct quotes (in-text) and/or book chapters (bibliography/ reference list) is different across multiple instances.</li> </ul>
<u>Technical correctness</u>  <ul style="list-style-type: none"> <li>Referencing format is technically correct throughout the submission.</li> <li>Position of the reference: a reference is directly associated with every concept or idea.</li> <li>For example, quotation marks, page numbers, years, etc. are applied correctly, sources in the bibliography/reference list are correctly presented.</li> </ul>	<b>Generally, technically correct with some minor errors.</b> <ul style="list-style-type: none"> <li>The correct referencing format has been consistently used, but there are one or two errors.</li> <li>Concepts and ideas are typically referenced, but a reference is missing from one small section of the work.</li> <li>Position of the references: references are only given at the beginning or end of every paragraph.</li> <li>For example, the student has incorrectly presented direct quotes (in-text) and/or book chapters (bibliography/reference list).</li> </ul>	<b>Technically incorrect.</b> <ul style="list-style-type: none"> <li>The referencing format is incorrect.</li> <li>Concepts and ideas are typically referenced, but a reference is missing from small sections of the work.</li> <li>Position of the references: references are only given at the beginning or end of large sections of work.</li> <li>For example, incorrect author information is provided, no year of publication is provided, quotation marks and/or page numbers for direct quotes missing, page numbers are provided for paraphrased material, the incorrect punctuation is used (in-text); the bibliography/reference list is not in alphabetical order, the incorrect format for a book chapter/journal article is used, information is missing e.g. no place of publication had been provided (bibliography); repeated sources on the reference list.</li> </ul>
<b>Congruence between in-text referencing and bibliography/ reference list</b>  <ul style="list-style-type: none"> <li>All sources are accurately reflected and are all accurately included in the bibliography/ reference list.</li> </ul>	<b>Generally, congruence between the in-text referencing and the bibliography/ reference list with one or two errors.</b> <ul style="list-style-type: none"> <li>There is largely a match between the sources presented in-text and the bibliography.</li> <li>For example, a source appears in the text, but not in the bibliography/ reference list or vice versa.</li> </ul>	<b>A lack of congruence between the in-text referencing and the bibliography.</b> <ul style="list-style-type: none"> <li>No relationship/several incongruities between the in-text referencing and the bibliography/reference list.</li> <li>For example, sources are included in-text, but not in the bibliography and vice versa, a link, rather than the actual reference is provided in the bibliography.</li> </ul>
<b>In summary:</b> the recording of references is accurate and complete.	In summary, at least <b>80%</b> of the sources are correctly reflected and included in a reference list.	In summary, at least <b>60%</b> of the sources are incorrectly reflected and/or not included in reference list.

**Overall Feedback** about the consistency, technical correctness and congruence between in-text referencing and bibliography:

.....

.....

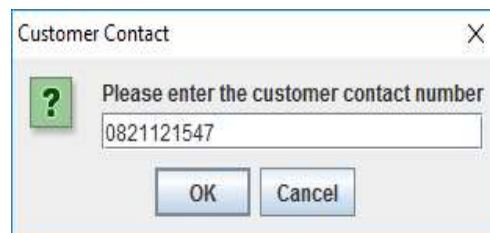
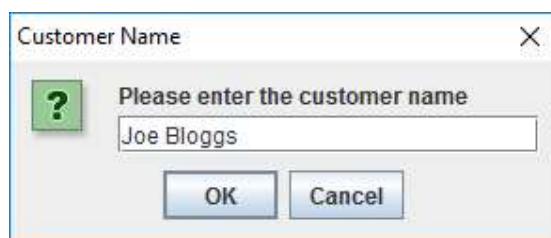
**Learning Area: Inheritance****Question 1****(Marks: 35)**

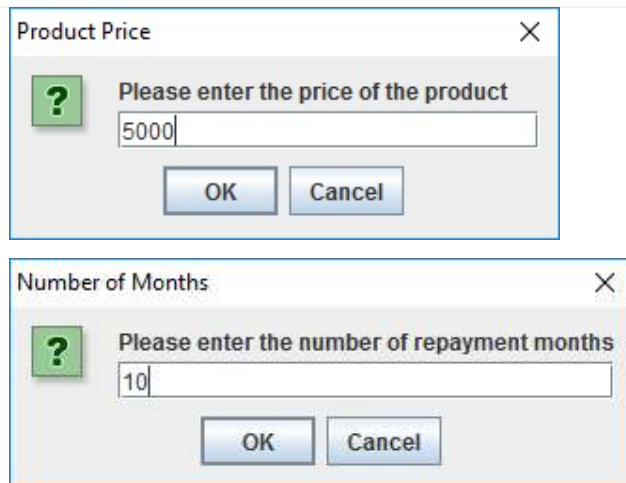
Create a class named **Customer** that will determine the monthly repayment amount due by a customer for a product bought on credit. The class has five fields: customer name, contact number, product price, number of months and the monthly repayment amount. Write get and set methods for each field, except for the monthly repayment amount field. The set methods must prompt the user to enter the values for the following fields: customer name, contact number, product price and number of months. This class also needs a method to calculate the monthly repayment amount (product price divided by the number of months).

Add a **subclass named Finance\_Period** that will determine if a customer will pay interest or not. If the number of months to pay for the product is greater than three, the customer will pay 25% interest, or else no interest applies. The maximum number of months to pay for the product is 12. **Override** the `calculate_repayment()` method by determining if the customer will pay interest or not and calculate the monthly repayment amount.

Create a class called **Customer\_Finance** that contains the logic to test the two classes. Prompt the user for data for the first object where no interest applies and display the results; then prompt the user for data where interest is applicable and display the results.

Sample Screen Shots:





The first dialog box, titled "Product Price", contains a green question mark icon, the text "Please enter the price of the product", a text input field with "5000", and "OK" and "Cancel" buttons. The second dialog box, titled "Number of Months", contains a green question mark icon, the text "Please enter the number of repayment months", a text input field with "10", and "OK" and "Cancel" buttons.

The following output is where interest is applicable:



The "Message" dialog box displays the following information:

- Customer Name: Joe Bloggs
- Customer Contact: 0821121547
- Product Amount: R 5,000
- Repayment Months: 10
- Monthly Repayment: R 625
- Total Due: R 6,250

An "OK" button is at the bottom.

The following output is when interest is NOT applicable:



The "Message" dialog box displays the following information:

- Customer Name: Joe Bloggs
- Customer Contact: 0821121547
- Product Amount: R 5,000
- Repayment Months: 3
- Monthly Repayment: R 1,666.67
- Total Due: R 5,000

An "OK" button is at the bottom.

### Question 1 Mark Allocation

Marking Guideline	Mark	Examiner	Moderator
<p><b>Correct Customer class created</b></p> <p><b>Excellent:</b> Correct variables and methods created — up to a maximum of 8 marks. Meaning all variables are declared, including set methods (which set the variables), get methods and the calculate method (calculates repayment).</p> <p><b>Good:</b> Attempted but minor errors — up to a maximum of 6 marks.</p> <p><b>Developing:</b> Attempted but not correct — up to a maximum of 3 marks.</p> <p><b>Poor:</b> Not attempted at all — 0 marks.</p>	8		
<p><b>Additional calculation class created</b></p> <p><b>Excellent:</b> Subclass with correct calculate_repayment() method that determines if a customer will pay interest or not — up to a maximum of 5 marks.</p> <p><b>Good:</b> Attempted, but minor errors — up to a maximum of 4 marks.</p> <p><b>Developing:</b> Attempted but not correct — up to 2 marks.</p> <p><b>Poor:</b> Not attempted at all — 0 marks.</p>	5		
<p><b>Get and set methods created in second class</b></p> <p><b>Excellent:</b> Get and set methods created in second class — up to a maximum of 4 marks.</p> <p><b>Good:</b> Attempted, but minor errors — up to a maximum of 3 marks.</p> <p><b>Developing:</b> Attempted but not correct — up to 2 marks.</p> <p><b>Poor:</b> Not attempted at all — 0 marks.</p>	4		
<p><b>Input and Output (use of JOptionPane)</b></p> <p><b>Excellent:</b> All input and outputs correct — up to a maximum of 4 marks.</p> <p><b>Good:</b> Attempted, but minor errors - up to a maximum of 3 marks.</p>	4		

<b>Developing:</b> Attempted but not correct — up to 2 marks. <b>Poor:</b> Not attempted at all — 0 marks.			
<b>Customer_Finance class created</b> <b>Excellent:</b> <b>Two objects</b> instantiated and methods applied – up to a maximum of 8 marks. <b>Good:</b> <b>One object</b> instantiated and method(s) applied – up to a maximum of 6 marks. <b>Developing:</b> Attempted but minor errors — up to 4 marks. <b>Poor:</b> Not attempted at all – 0 marks.	<b>8</b>		
<b>Save file correctly</b>	<b>1</b>		
<b>Good Programming Practise</b>	<b>2</b>		
<b>Code Efficiency</b>	<b>1</b>		
<b>Comments</b>	<b>2</b>		
<b>TOTAL</b>	<b>35</b>		

**Question 2****(Marks: 65)**

The software development house you work for has decided to create a prototype based on the game **Scrabble**, *but with a twist*, that will be used on mobile devices. They require a prototype from you that is designed using a Java console application. If the prototype is a success, the application will be developed for the mainstream mobile platforms.

You are firstly required to prompt the user to enter in a one to start the game. If the users decide to play the game, prompt for the two player names.

```
Welcome to the WORD WARS game.

Press (1) To start the game.

Press any other key to exit the game
Enter your selection: 1
*****
Enter player 1 name: Andile
Enter player 2 name: Sam
```

The players will then be presented with the alphabet. Each player has to enter a word with the available letters in the alphabet list. When a player enters a word, the letters of that word get removed from the alphabet list. Only vowels are not removed from the alphabet list. Before a word can be accepted both players must agree if the word entered is valid or not.

```
LETS PLAY WORD WARS!!!
Alphabet letters left: a b c d e f g h i j k l m n o p q r s t u v w x y z
Andile enter your word: java
Enter (y) yes if both players agree on the word
y
Alphabet letters left: a b c d e f g h i k l m n o p q r s t u w x y z
Sam enter your word: logic
Enter (y) yes if both players agree on the word
y
Alphabet letters left: a b d e f h i k m n o p q r s t u w x y z
Andile enter your word:
```

It is suggested that players have a dictionary present to determine the validity of the word. If the word is accepted, the player is awarded one point for each letter in the word.

This letter tally will determine who the winner is when the game is finished. A letter can be used more than once in the creation of a word, but then must be removed from the alphabet list. For example, the word “FILL” contains two “L”, which must be removed from the alphabet list.

Once a letter has been removed from the alphabet list, a player cannot enter a word that contains the missing letters. You are required to display a suitable message for this and prompt the player to enter in a new word.

```
Alphabet letters left: a b d e f h i k m n o p q r s t u w x y z
Andile enter your word: concatenation
YOU ENTERED A WORD THAT CONTAINS A LETTER THAT IS USED OR IS NOT VALID. PLEASE ENTER ANOTHER WORD!
Andile enter your word: |
```

The game continues until a player cannot create any more words with the remaining alphabet list. If a player cannot create a word, they are required to enter in a sentinel value of three question marks “???”. Once a player has quit the game, display who the winner of the game is with their accumulated points or if the game is tied.

```
Andile enter your word: ???
WINNER OF THE GAME IS: SAM with a score of: 5
YOUR NAME HAS BEEN SAVED TO THE HALL OF FAME!!!
THE GAME IS NOW OVER. THANKYOU FOR PLAYING WORD WARS!!!
```

## Question 2 Mark Allocation

Marking Guideline	Mark	Examiner	Moderator
<b>Variables declared and assigned</b> <b>Excellent:</b> All variables are declared and assigned — up to a maximum of 8 marks. <b>Good:</b> Attempted but minor errors — up to a maximum of 6 marks. <b>Developing:</b> Attempted but not correct — up to a maximum of 3 marks. <b>Poor:</b> Not attempted at all — 0 marks.	8		



<b>Input of player names</b> <b>Excellent:</b> Input of players has been executed correctly and efficiently — up to a maximum of 4 marks. <b>Good:</b> Attempted but minor errors — up to a maximum of 2 marks. <b>Developing:</b> Attempted but not correct — up to a maximum of 1 mark. <b>Poor:</b> Not attempted at all — 0 marks.	<b>4</b>		
<b>Code to manipulate the alphabet list</b> <b>Excellent:</b> Code manipulates the alphabetic list correctly and efficiently — up to a maximum of 20 marks. <b>Good:</b> Attempted but minor errors — up to a maximum of 15 marks. <b>Developing:</b> Attempted but not correct — up to a maximum of 10 marks. <b>Poor:</b> Not attempted at all — 0 marks.	<b>20</b>		
<b>Code to display the alphabet list</b> <b>Excellent:</b> Code displays the alphabetic list correctly and efficiently — up to a maximum of 8 marks. <b>Good:</b> Attempted but minor errors — up to a maximum of 6 marks. <b>Developing:</b> Attempted but not correct — up to a maximum of 4 marks. <b>Poor:</b> Not attempted at all — 0 marks.	<b>8</b>		
<b>Code to determine the winner</b> <b>Excellent:</b> Code determines the winner — up to a maximum of 10 marks. <b>Good:</b> Attempted but minor errors — up to a maximum of 8 marks. <b>Developing:</b> Attempted but not correct — up to a maximum of 4 marks. <b>Poor:</b> Not attempted at all — 0 marks.	<b>10</b>		

<b>Play again message to the user</b> <b>Excellent:</b> a play again message appears to the user — up to a maximum of 5 marks. <b>Good:</b> Attempted but minor errors — up to a maximum of 3 marks. <b>Developing:</b> Attempted but not correct — up to a maximum of 2 marks. <b>Poor:</b> Not attempted at all — 0 marks.	<b>5</b>		
<b>Save file correctly</b>	<b>2</b>		
<b>Good programming practise</b>	<b>3</b>		
<b>Code efficiency</b>	<b>3</b>		
<b>Comments</b>	<b>2</b>		
<b>TOTAL</b>	<b>65</b>		