



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

<b>ASSESSMENT TYPE:</b>	<b>EXAMINATION (PAPER ONLY)</b>
<b>TOTAL MARK ALLOCATION:</b>	<b>120 MARKS</b>
<b>TOTAL HOURS:</b>	<b>3 HOURS (+15 minutes reading time)</b>

**INSTRUCTIONS:**

1. Please adhere to all instructions in the assessment booklet.
2. Independent work is required.
3. Five minutes per hour of the assessment to a maximum of 15 minutes is dedicated to reading time before the start of the assessment. You may make notes on your question paper, but not in your answer sheet. Calculators may not be used during reading time.
4. You may not leave the assessment venue during reading time, or during the first hour or during the last 15 minutes of the assessment.
5. Ensure that your name is on all pieces of paper or books that you will be submitting. Submit all the pages of this assessment's question paper as well as your answer script.
6. Answer all the questions on the answer sheets or in answer booklets provided. The phrase 'END OF PAPER' will appear after the final set question of this assessment.
7. Remember to work at a steady pace so that you are able to complete the assessment within the allocated time. Use the mark allocation as a guideline as to how much time to spend on each section.

**Additional instructions:**

1. This is a OPEN BOOK assessment.
2. Calculators are allowed.
3. For open book assessments the students may have open access to all resources inclusive of notes, books (hardcopy and e-books) and the internet. These resources may be accessed as hard copies or as electronic files on electronic devices. All electronic devices batteries must be fully charged before the assessment as no charging of devices will be permitted during the sitting of the assessment. The IIE and associated brands accept no liability for the loss or damage incurred to electronic devices used during open book assessments.
4. Answer All Questions.
5. Instructions for assessments including practical computer work:
  - Use of good programming practice and comments in code is compulsory.
  - Save your application in the location indicated by the administrator (e.g., the Z:\ drive or your local drive).
  - Create a folder as follows: use the module code and your own student number and create a folder with a folder name as per the format shown here:
  - **StudentNumber\_ModuleCode\_Exam**. Save all files (including any source code files, template files, design files, image files, text files, database files, etc.) within this folder.

- *E.g., if your student number is 12345, and you are writing an examination for the module PROG121, create a folder named **12345\_Prog121\_Exam** and use this throughout the session to save all of your files.*
- **Important:** *Upon completion of your assessment, you must save and close all your open files and double click the ExamLog application on your desktop. You must follow the instructions carefully to ensure that the information about the files that you have submitted for this assessment has been logged on the network. Specify the location of your source code on your question paper.*

**GENERAL REQUIREMENTS****(Marks: 10)**

Writing maintainable code in industry is vital. Therefore, you are generally required to:

- Follow good programming practices:
  - Variable types correct;
  - Variable scope correct; and
  - Class and method naming standards are correct.
- Insert comments to show logic and design for the support of code maintainability.
- Code efficiently. Redundant code must be avoided.
- Spend 15 minutes ensuring that your programs meet the general criteria below.

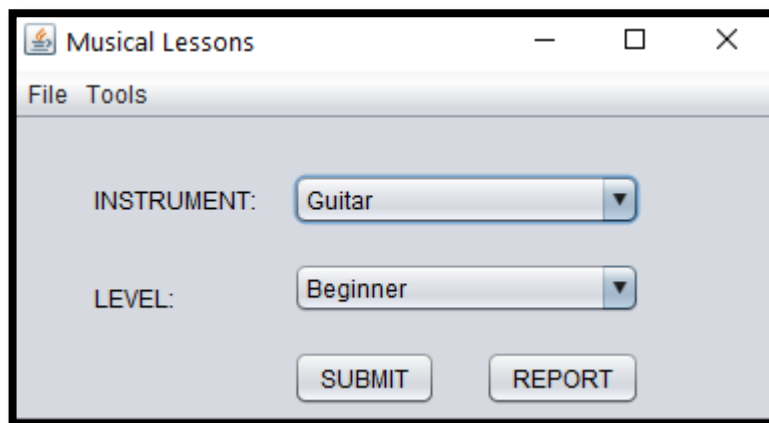
Requirement	Maximum Mark	Examiner's Mark	Moderator's Mark
Good Programming Practice <ul style="list-style-type: none"> <li>• Not followed at all = 0</li> <li>• Average – Minor changes required = 1 – 2</li> <li>• Excellent – No changes necessary = 3</li> </ul>	(3)		
Code Efficiency <ul style="list-style-type: none"> <li>• Poor – Much code is duplicated = 0</li> <li>• Average – Some redundant code = 1</li> <li>• Excellent – Code very maintainable = 2</li> </ul>	(2)		
Comment Statements <ul style="list-style-type: none"> <li>• Poor – No comments = 0</li> <li>• Average – Some comments = 1 – 2</li> <li>• Excellent – All necessary comments were given to show logic = 3</li> </ul>	(3)		
Program(s) compile and execute <ul style="list-style-type: none"> <li>• No – Many changes required = 0</li> <li>• Average – Minor changes required = 1</li> <li>• Yes – No modifications required = 2</li> </ul>	(2)		
<b>TOTAL MARKS</b>	<b>10</b>		

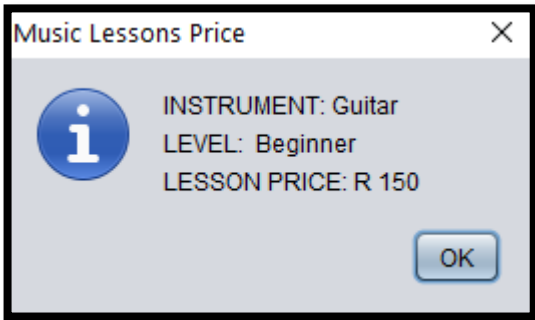
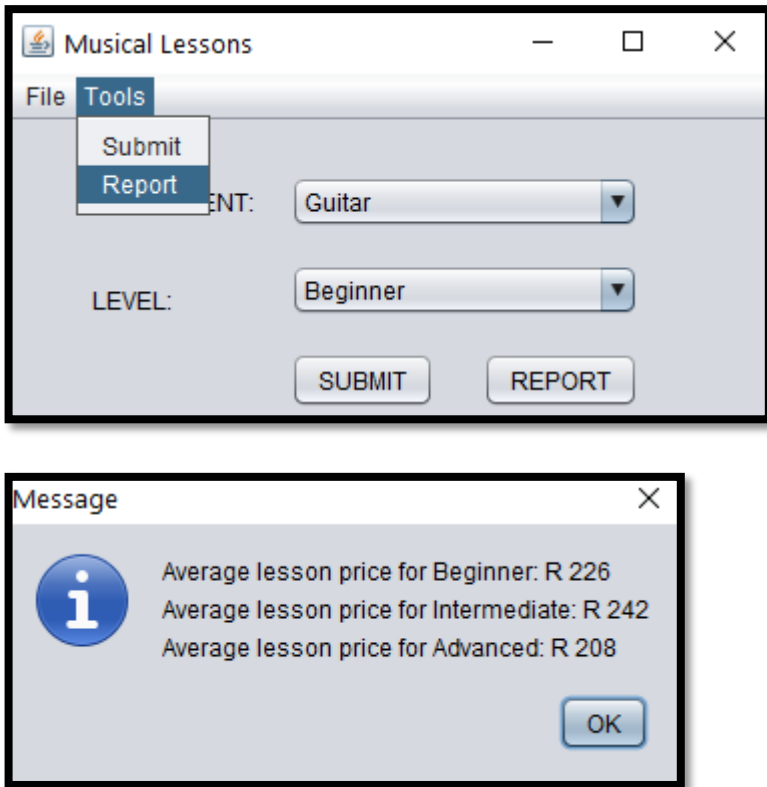
**Question 1****(Marks: 50)**

Develop a Java GUI application that will display the cost of musical lessons for three different instruments (Guitar, Piano, and the Violin).

**Q.1.1** Create two combo boxes to allow a user to select between three different musical instruments and the required musical level on the form. Also, add two buttons, one for submitting the instrument selection and another to display an average musical level report. Once the user has selected a musical instrument and musical level, display the lesson cost for that particular instrument. Use the following table for the musical instruments and lesson levels:

	Guitar	Piano	Violin
Beginner	150	250	280
Intermediate	215	232	280
Advanced	130	185	310

**Sample screenshot**

	
Q.1.2	<p>You must also create a menu system that will allow the user to exit the application under the file menu. Then, under the Tools menu, allow the form submission and display the average level lesson cost. The layout of the form is left to your discretion. Marks will be allocated to the presentation and effectiveness of the design, but the following layout is displayed for your convenience.</p>
	<p><b>Sample screenshot</b></p> 
Q1.3	<ul style="list-style-type: none"> <li>• Comment your program.</li> <li>• Marks will be allocated to the declaration of objects, logic, class definitions, method calls etc.</li> </ul>

Question 1 Mark Allocation	Levels of Achievement				Feedback
	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ marks possible)				
GUI form correctly created	8-10	4-7	2-3	0-1	
Correctly displays the cost of an instrument lesson based on the level selection	10-15	7-9	3-6	0-2	
Output of the average level cost	8-10	4-7	2-3	0-1	
File - Exit menu item created that functions correctly	5	3-4	1-2	0	
Tools - Submit menu created that functions the same as the submit button	5	3-4	1-2	0	
Tools - Report menu created that functions the same as the report button	5	3-4	1-2	0	

**Question 2****(Marks: 20)**

Write a Java application and use a Two-dimensional array that will store five motorbike manufacturer sales, with each manufacturer's total sales.

Your program must:

- Q.2.1** Contain a Two-dimensional array to store three quarter sale amounts for five different motorbike manufacturers and a single array to contain the manufacturers.

BRAND		QUARTER 1	QUARTER 2	QUARTER 3
Triumph		500	100	500
Honda		70	80	200
Suzuki		100	100	200
Yamaha		100	70	50
Ducati		300	100	500

- Q.2.2** Calculate and print the total sales for each manufacturer.

- Q.2.3** Determine if the manufacturer status is Gold or Silver. If the total sales amount of the manufacturer is greater than or equal to three hundred (300), then the manufacturer receives Gold status.

- Q.2.4** Print out each manufacturer quarter sales amount with the total sales amount.

**Sample screenshot**

BRAND	QUARTER 1	QUARTER 2	QUARTER 3	TOTAL	STATUS
Triumph	500	100	500	1100	Gold
Honda	70	80	200	350	Gold
Suzuki	100	100	200	400	Gold
Yamaha	100	70	50	220	Silver
Ducati	300	100	500	900	Gold

Question 2 Mark Allocation	Levels of Achievement				Feedback
	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ marks possible)				
2D Array was created to store the motorbike manufacturer sale amounts.	5	3-4	1-2	0	
Single array to store the manufacturer names.	5	3-4	1-2	0	
Calculate the total manufacturer sales.	5	3-4	1-2	0	
Determine the manufacturer status.	5	3-4	1-2	0	



**Question 3****(Marks: 40)**

Write a Java GUI application that will inform a user of medical issues related to snake bites. The application must display a fatality ranking and how soon medical assistance is required if a specific type of snake bites an individual.

**Q.3.1** On the form, create a combo box and a list box. The combo box must display the type of snake, and the list box must display the medical information. Also, create a submit button that, when clicked, will show the snake bite medical information. Finally, include a clear button to clear the list box.

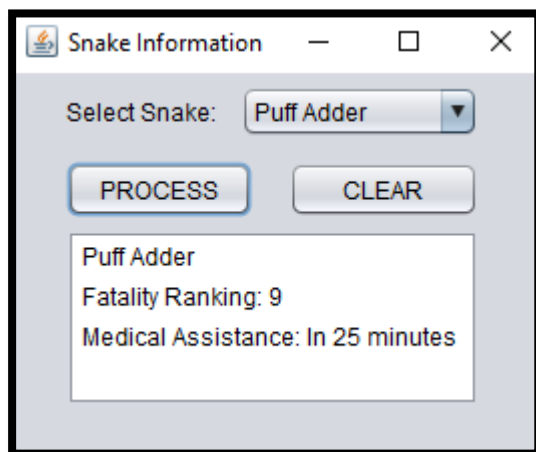
**Q.3.2** Create a sequential file (snakes.txt) that contains data for the following fields:

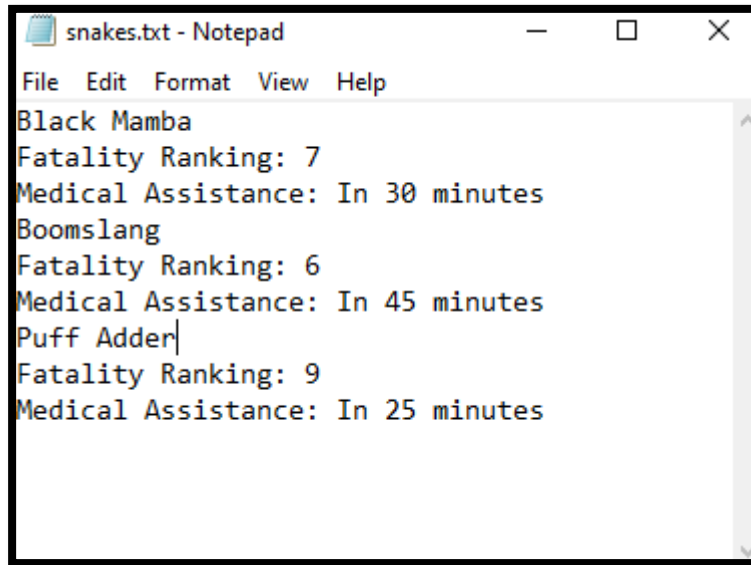
- The type of snake.
- The fatality ranking.

How quickly medical assistance is required

**Q.3.3** When the application starts, load the type of snake from the snakes.txt file into the combo box.

**Q.3.4** When the submit button is clicked, load the snake bite information into the list box from the sequential file based on the snake selected.

**Sample screenshot**

**Sample Student Text File:**


```

snakes.txt - Notepad
File Edit Format View Help
Black Mamba
Fatality Ranking: 7
Medical Assistance: In 30 minutes
Boomslang
Fatality Ranking: 6
Medical Assistance: In 45 minutes
Puff Adder
Fatality Ranking: 9
Medical Assistance: In 25 minutes
  
```

Question 3 Mark Allocation	Levels of Achievement				Feedback
	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ marks possible)				
GUI form created with all objects	8-10	4-7	2-3	0-1	
Sequential file created and populated	8-10	4-7	2-3	0-1	
Reading from the sequential file performed correctly	8-10	4-7	2-3	0-1	
Populating the combo box and list box with the type of snake and bite information from the text file	8-10	4-7	2-3	0-1	

**END OF PAPER**