ROBERT GORDON UNIVERSITY ABERDEEN

Assessment Brief

Academic Year	**2024-25**
Semester	**Y1 (Semester 02)**
Module Number	**CM1601**
Module Title	**Programming Fundamentals**
Assessment Method	**Coursework**
Deadline (time and date)	**21st of July 2025 11:00 PM**
Submission	**Assessment Dropbox in the Module Study Area in Campus Moodle.**
Word Limit (see Assessment Word Limit Statement)	-
Module Co-ordinator	**Dinusha Ruwan Kumara**

What knowledge and/or skills will I develop by undertaking the assessment?

1. Demonstrate competence in the algorithmic approach to problem solving.
2. Design, code, compile, test and execute fundamental programming concepts using a high-level programming language.
3. Construct robust, maintainable programs that use object-orientated analysis and design principles.

On successful completion of the assessment students will be able to achieve the following Learning Outcomes:

1. Design, code, compile, test and execute fundamental programming concepts using a high-level programming language.
2. Construct robust, maintainable programs that use object-orientated analysis and design principles.

Please also refer to the Module Descriptor, available from the module Moodle study area.

What is expected of me in this assessment?

Task(s) – content

This coursework is a continuation of Coursework 1. You are required to use Java and Object-Oriented Programming (OOP) principles to develop a Graphical User Interface (GUI) application for John's Super Market, a department store.

Objective

Your task is to develop a JavaFX-based GUI application for John's Super Market, supporting a range of inventory and dealer management functionalities. In addition to the application, you must submit a comprehensive report documenting your design decisions, implementation, testing, and adherence to software quality principles.

System Requirements / Features

The application should launch with a JavaFX GUI and provide the following features:

- **Display low-stock items at startup:** When the application starts, it should automatically display all inventory items where the quantity is below a configurable threshold 'x'. Each item should have its own 'x' value defined in the system.
- Add new item details to the inventory.
- Delete item details from the inventory.
- Update existing item details.
- View all items in the inventory, sorted by item category, and display the current total item count or value.
- Save item details to a text file at any point.
- Randomly select four suppliers (dealers) from a file.
- Display full details of the selected suppliers, sorted by location.
- Display all items provided by a selected supplier.
- Exit the application cleanly.

Coursework Sections	**Detailed content**	**Learning Outcome**	**Topic/Week Module Activity**
Display Low-Stock Items at Startup	When the application starts, it should automatically identify and display all inventory items where the current quantity is below a configurable low-stock threshold 'x'. This 'x' value must be defined for each individual item within the system (e.g., as part of the item's stored details).	2,3	Lecture 2 - Operators and Expressions Lecture 4 – Control Flow Statements Lecture 10 – File Handling
Adding item details.	The system should allow the owner to add the details of items to his system by prompting the required information. Item Code, Item Name, Item Brand, Price, Quantity, Category, Purchased Date and an Item Image. Example: 001, Speaker, Samsung, Rs 21966.00, 10, Speakers, 2025/02/07, img1.jpg	1	Lecture 2 – Operators and Expressions Lecture 7 - Methods

What is expected of me in this assessment?

Viewing the items	The system should display all items ordered by item category, and then by item-ID within each category, in ascending order. This table should be neatly formatted with all details, including the thumbnail image. Display the current total item count and total monetary value of the inventory.	1	Lecture 7: Java Strings
Deleting item details	The system should allow the user to delete an item using a user-friendly mechanism (e.g., selecting from a list, searching by item code). Provide clear feedback on the success or failure of the deletion.	1	Lecture 3 – Java Decision Making
Updating item details.	The system should allow the user to update an item (e.g., by searching for its code). Ensure all fields are updatable and validated. Provide clear feedback on the success or failure of the update.	1	Lecture 2 - Operators and Expressions
Saving/Loading Item Data	The system should allow the owner to save the details of the inventory items to a text file at any point (e.g., when an item is added, deleted, or updated). The system should also be able to load the current inventory data from the text file upon application startup to enable resume capabilities. Implement robust exception handling for file operations.	1	Lecture 2 - Operators and Expressions Lecture 4 – Control Flow Statements Lecture 5 – Control Flow Statements II Lecture 6- Exception Handling
Selecting four dealers randomly	The System should simulate a random draw and select four dealers accordingly from a text file. Once selection is done, display the selected dealer details. (Text file(s) should be created by yourself accordingly).	1	Lecture 8- Collections
Displaying all the details of the randomly selected dealers	The System should display all the details of the randomly selected dealers and sorted according to the location. Use your own algorithm for sorting. (Do not use standard sorting methods)	1,2	Lecture 10 – File handling
Display the items of the given dealer	System should be able to load the current data from the text file to enable resume capabilities.	1,2	Lecture 10 – File handling
Exit the application cleanly.			

Task(s) - format

1. All the functionalities mentioned above need to be fully implemented. Code should be submitted as a single zip file to the link in assessment section. Check the following key points before you start the implementation

   a. A GUI system to enable the user input with validations should be implemented.
   b. Information entered should be saved in text file(s).
   c. You are not allowed to use databases for this coursework.
   d. When sorting data, you are not allowed to use inbuilt Java libraries and marks will be given only for your own algorithms.
   e. All the functionalities which were developed earlier in your last coursework need to be implemented again.
   f. You are allowed to use Java FX for implementing the user interfaces. You have the freedom to design the interfaces.
   g. When implementing the application, you are instructed to use classes and objects and data structures with OOP concepts for
      a. Classes and objects
      b. Encapsulation
      c. Inheritance
      d. Abstraction and polymorphism
   h. Create a Junit test suit to cover the whole scenario. Include in the report.

2. You should submit a report containing the following.
   a. Write a test plan to cover the whole scenario. Each test case should have a name, inputs, expected output and the actual output.
   b. Your complete java code for each class with a brief description.

   Explain what you have done in order to ensure the robustness and the maintainability of your source code. Mainly explain whether you followed the guidelines you were taught during the code quality lecture sessions. Further you have followed SOLID principles when designing the system or not. Feel free to refer online resources when justifying.

3. Coursework Report Format:
   a. Your report should be word-processed using Times New Roman, font size 12, with 1.5 line spacing, properly referenced, and uploaded as one final PDF document to the Assessment drop box in the Assessment section of the Moodle page.

4. Report Structure
   a. Title Page
   b. Contents Page
   c. Flow charts
   d. Introduction to functions with code
   e. J-units, Test plan and test cases
   f. Robustness and the maintainability
   g. Assumptions
   h. Conclusion
   i. Reference list
   j. Appendices (if appropriate – 6 pages maximum)

How will I be graded?

A grade will be provided for each criterion on the feedback grid which is specific to the assessment.

The overall grade for the assessment will be calculated using the algorithm below.

A	At least 50% of the feedback grid to be at Grade A, at least 75% of the feedback grid to be at Grade B or better, and normally 100% of the feedback grid to be at Grade C or better.
B	At least 50% of the feedback grid to be at Grade B or better, at least 75% of the feedback grid to be at Grade C or better, and normally 100% of the feedback grid to be at Grade D or better.
C	At least 50% of the feedback grid to be at Grade C or better, and at least 75% of the feedback grid to be at Grade D or better.
D	At least 50% of the feedback grid to be at Grade D or better, and at least 75% of the feedback grid to be at Grade E or better.
E	At least 50% of the feedback grid to be at Grade E or better.
F	Failing to achieve at least 50% of the feedback grid to be at Grade E or better.
NS	Non-submission.

ROBERT GORDON UNIVERSITY ABERDEEN

Feedback grid

GRADE	A	B	C	D	E	F
DEFINITION / CRITERIA (WEIGHTING)	**EXCELLENT (Outstanding Performance)**	**COMMENDABLE / VERY GOOD (Meritorious Performance)**	**GOOD (Highly Competent Performance)**	**SATISFACTORY (Competent Performance)**	**BORDERLINE FAIL**	**UNSATISFACTORY (Fail)**
Display low-stock items at startup (10 x %)	Application automatically displays all low-stock items correctly and clearly upon startup, accurately applying the configurable 'x' threshold for each individual item. The display is highly user-friendly and seamlessly integrated into the GUI. Robust validation and comprehensive error handling for 'x' values are evident.	Application automatically displays most low-stock items correctly upon startup, generally applying the configurable 'x' threshold for each individual item. The display is clear and user-friendly. Basic validation for 'x' values is present.	Application attempts to display low-stock items at startup, showing some items based on a threshold. The 'x' value might be a general system-wide threshold rather than fully configurable per item, or its per-item configuration is partially implemented. The display is functional but may lack polish.	Application makes an attempt to display low-stock items at startup, but the functionality is limited or contains noticeable errors. The threshold 'x' is likely fixed, not correctly applied, or not configurable per item. The display may be basic or hard to read.	The low-stock item display is implemented but is largely non-functional, or only a very small subset of items is processed correctly. There are significant errors in logic or display.	The low-stock item display at startup feature is not implemented, or it fails to run.
A GUI menu system with validations (10 x %)	All the Options are displayed on a Java FX GUI neatly and correctly. Users can select the option by clicking, navigating to a page or tab. Inputs are validated. The student demonstrates excellent knowledge on the task trough viva. Beautifully crafted GUIs.	More than 7 Options are displayed on a JAVA FX GUI correctly. Users can select the option by clicking, navigating to a page or tab. basic input validation can be seen. The student demonstrates very good knowledge on the task through viva. GUIs are designed nicely.	More than 6 Options are displayed on a JAVA FX GUI correctly. Users can select the option by clicking, navigating to a page or tab. basic input validation. The student demonstrates good knowledge on the task trough viva. Good design of GUIs.	More than 5 Options are displayed on a JAVA FX GUI correctly. Users can select the option by clicking. minor input validation. The student demonstrates adequate knowledge on the task through viva. Satisfactory GUIs	Less than 5 Options are displayed on a Console menu. Users can only select a few options via input. no input validations can be seen. The student does not demonstrate adequate knowledge on the task through viva.	No GUI menu to be seen. user inputs are not considered. Demonstrates no knowledge on the task. Program does not run.

GRADE	A	B	C	D	E	F
Add, delete & update item details **(10x %)**	Adding, deleting, and updating of item details can be done by searching. Duplicate item records are handled, and all the inputs are correctly validated. sub-menu for each function works flawlessly. The student demonstrates excellent knowledge on the task trough viva.	Adding, deleting, and updating of item details can be done by a single attribute. duplicate item records are handled, and all the inputs are correctly validated. sub- menu for each function works. The student demonstrates very good knowledge on the task trough viva	Adding, deleting, and updating of item details can be done. No duplicate item records are handled. Basic inputs are validated. sub-menu for each function works. The student demonstrates good knowledge on the task trough viva	Adding, deleting, and updating of item details can be done. No duplicate item records are handled. no inputs are validated. sub-menu for each function works. The student demonstrates adequate knowledge on the task trough viva	only two of the functions out of three can be done. No duplicate item records are handled. Basic inputs are not validated. sub- menu for each function works with errors. The student demonstrates low knowledge on the task trough viva	Functions do not work. Demonstrates no knowledge on the task. Program does not run.
Item table and dealer table **(10x %)**	Item and dealer tables are neatly formatted to imitate a tabular format. Data are sorted according to the Points and date respectively. Student has implemented two sorting algorithms for the Items and dealer table. All the required columns can be seen in the tables. The student demonstrates excellent knowledge on the task trough viva.	Item and dealer tables are neatly formatted. Data are sorted according to the Points and date respectively. Student has implemented at least one sorting algorithms for the Items or dealer table. required columns can be seen in the tables. The student demonstrates very good knowledge on the task trough viva.	Item and dealer tables are displayed. Data are sorted according to the Points and date respectively. Student has implemented at least one sorting algorithms for the Items or dealer table. Most of the required columns can be seen in the output. The student demonstrates good knowledge on the task trough viva.	Item and dealer tables are displayed. Data are sorted according to the Points and date respectively but by using the default sort function. Most of the required columns can be seen in the output. The student demonstrates satisfactory knowledge on the task trough viva.	Item or dealer table is printed on to the console with nor formatting. No sorting can be seen, or sorting is incorrect. Most of the required columns are missing in the output. The student demonstrates low knowledge on the task trough viva.	Item or dealer table is not printed on to the console. student demonstrates no knowledge on the task trough viva.
Random Dealer (10x %)	Random dealer is generated with all the Items in the system. Duplicate date has been validated. Dealer details are stored in the file correctly. Student has developed an algorithm for this function. All the validations are done. The student demonstrates excellent knowledge on the task trough viva.	Random dealer is generated with all the Items in the system. Duplicate date has been validated. The student demonstrates very good knowledge on the task trough viva.	Random dealer is Points are updated after each dealer. The student demonstrates good knowledge on the task trough viva.	Random dealer is generated with most of the Items in the system No validations can be seen. The student demonstrates satisfactory knowledge on the task trough viva.	Random dealer is generated with no relevance to the existing data. The student demonstrates unsatisfactory knowledge on the task trough viva.	Random dealer function is not implemented. student demonstrates no knowledge on the task trough viva.

GRADE	A	B	C	D	E	F
File handling (10x %) Grade:	Serializable text files have been used to store the relevant data. Necessary validations have been implemented and possible duplicate errors were addressed. Exceptions were handled in the code. Resume capability of the system have been implemented. The student demonstrates excellent knowledge on the task trough viva.	Clear text or xml files have been used to store the relevant data. Necessary validations have been implemented and possible duplicate errors were addressed. Resume capability of the system have been implemented. The student demonstrates very good knowledge on the task trough viva.	Clear text files have been used to store data. validations were implemented. Resume capability of the system have been implemented. The student demonstrates good knowledge on the task trough viva.	At least one text file has been used to store data. Basic validations can be seen. Resume capability of the system have been implemented to a functioning level. The student demonstrates satisfactory knowledge on the task trough viva.	Attempts were made to code the file handling but not working. File has been created without data. The student demonstrates unsatisfactory knowledge on the task trough viva.	File handling is not implemented. student demonstrates no knowledge on the task trough viva.
Report & Test Plan (15x %) Grade:	A Detailed, fully covered test plan can be seen in the report. Excellent justification is given in the report on selecting test cases. Report has all the required details with references. Code is added to the report and clear detailed descriptions are given. Excellent formatting can be seen throughout the report. J-Units are correct and covered.	A test plan covering most of the functions can be seen in the report. good justification is given in the report on selecting test cases. Report has the required details with references. Code is added to the report with clear descriptions. no formatting errors can be seen. Both Flow charts are correct. J-Units are correct and covered.	A test plan covering most of the functions can be seen in the report. Basic justification is given in the report on selecting test cases. Report has the required details. Code is added to the report with brief descriptions. minor formatting errors can be seen. J-Units are correct, and majority of the functions are covered.	A test plan covering main functions can be seen in the report. Some justification is given in the report on selecting test cases. Report has the basic details. Code is added to the report with short descriptions. Minor formatting errors can be seen. J-Units are correct, and half of the functions are covered.	Incomplete report. Test plan does not cover even the main functions. Inadequate justifications are given. No references given. only few code fragments attached. Major formatting errors can be seen throughout the document. No J-Units	No test plan to be found. No justification given. No code is attached, or code attached as screen shots. No J-units
OOP Usage (15x %) Grade	Abstraction, Inheritance, Polymorphism and Encapsulation has been used throughout the CW implementation. Interfaces has been used. The student demonstrates excellent knowledge on OOP trough viva.	Al least 3 OOP concepts has been used. At least one interface has been used. The student demonstrates very good knowledge on OOP trough viva.	Abstraction and Encapsulation has been used throughout the CW implementation. At least one interface has been used. The student demonstrates good knowledge on OOP trough viva	Abstraction or Encapsulation has been used in the CW implementation. The student demonstrates satisfactory knowledge on OOP trough viva	No visible OOP concept usage throughout the usage. Even if used, purpose has been misled. The student demonstrates unsatisfactory knowledge on OOP trough viva.	No OOP Usage. student demonstrates no knowledge on the OOP trough viva.

GRADE	A	B	C	D	E	F
Code Quality and improvements (10x %)	Code comments, in code references, Proper naming conventions, Indentation and use of proper coding structures can be seen. Smart Improvements on the given specification is evident.	Code comments, Proper naming conventions, Indentation can be seen. Many Improvements on the given specification is evident.	Code comments, Proper naming conventions, Indentation can be seen. At least one Improvements on the given specification is evident.	Code comments, Proper naming conventions, Indentation can be seen.	No comments to be seen. indentation is not consistent. Major naming errors are visible.	No comments to be seen. No indentation of the code. has no idea on naming conventions.

What else is important to my assessment?

What is plagiarism?

"Plagiarism is the practice of presenting the thoughts, writings or other output of another or others as original, without acknowledgement of their source(s) at the point of their use in the student's work. All materials including text, data, diagrams or other illustrations used to support a piece of work, whether from a printed publication or from electronic media, should be appropriately identified and referenced and should not normally be copied directly unless as an acknowledged quotation. Text, opinions or ideas translated into the words of the individual student should in all cases acknowledge the original source" (RGU 2022).

What is collusion?

"Collusion is defined as two or more people working together with the intention of deceiving another. Within the academic environment this can occur when students work with others on an assignment, or part of an assignment, that is intended to be completed separately" (RGU 2022).

For further information please see Academic Integrity.

What is the Assessment Word Limit Statement?

It is important that you adhere to the Word Limit specified above. The Assessment Word Limit Statement lists what is included and excluded from the word count, along with the penalty for exceeding the upper limit.

What if I'm unable to submit?

- The University operates a Fit to Sit Policy which means that if you undertake an assessment then you are declaring yourself well enough to do so.
- If you require an extension, you should complete and submit a Coursework Extension Form. This form is available on the RGU Student and Applicant Forms page.
- Further support is available from your Course Leader.

What else is important to my assessment?

What additional support is available?

- RGU Study Skills provide advice and guidance on academic writing, study skills, maths and statistics and basic IT.
- RGU Library guidance on referencing and citing.
- The Inclusion Centre: Disability & Dyslexia.
- Your Module Coordinator, Course Leader and designated Personal Tutor can also provide support.

What are the University rules on assessment?

The University Regulation 'A4: Assessment and Recommendations of Assessment Boards' sets out important information about assessment and how it is conducted across the University.