FOP2 Mini Project – 2024

Description
Your task for this mini project will be to design and develop a simple point of sale computer program for a Coffee shop. Your system should include a **number based menu driven system** that is built using an inventory file that contains the following information.

Inventory file format:
[menu item], [price]

Inventory file example:
Americano, 3.00
Cappuccino, 3.50
:
Espresso, 2.00

This will allow the coffee shop owner to modify their coffee menu items and modify prices if they need to.

When customers enter the coffee shop they will select a coffee from the menu and the coffee shop owner will then process the customers transaction using the following steps:

1. Pick coffee item from the system menu
2. Select **CASH** or **CARD** transaction
    a. If cash transaction selected, then the amount tendered will be entered and a receipt will be generated on screen.
    b. If card transaction selected, then the card type (Visa/Master) will be selected and a receipt will be generated on screen.

The coffee shop system must keep a **record of all transactions**. All transactions must be stored while the program is in use and then written to a transaction file **when the user exists the system**. The transaction file will have different entries depending on the transaction type (CASH/CARD), see format below:

Transaction file
[Date and time stamp], [Item Purchased], [Price], [Amount tendered], [Change given]
[Date and time stamp], [Item Purchased], [Price], [Card type]

Requirements
Your system should be designed using the programming techniques covered in your lectures. The system should be easy to use and all user input should be **validated** to ensure there are no errors. **You must demonstrate your code as part of your submission.**

Plagiarism
This is an individual assessment and plagiarised work will not be accepted. Plagiarised work includes copying from other students (in whole or part thereof) or copying code from the web

(in whole or part thereof). You may not use AI generated code (CoPilot etc) as part of your submission. AI generated code can be very easily spotted. All code submitted must be your own work.

Any evidence of plagiarism will be treated very seriously and will **result in a grade of zero for all parties concerned**. A signed copy of the Plagiarism Declaration form must accompany your submission for this assessment. **Your work will not be graded without a signed declaration form.**

Deadline
Mini project deadline is **Sunday 14th of April 2024 @ 9pm**. The upload link will be available on MOODLE. Please upload a single .ZIP file containing your source, inventory and transaction files along with your documentation.

Assessment Weighting
This assessment is worth **40%** of your overall grade.

NOTES: All of the techniques necessary to complete the assessment will be covered in the lectures and labs. Therefore, engagement in the lab sessions will be necessary to complete this assessment. Additional video tutorials to support this assessment can be found on the MOODLE page.

Marking Rubric

	Weighting
Loading menu from CSV inventory file 1. *Hard coded Strings - [0]* 2. *Hard coded ArrayList<MenuItem> objects - [5]* 3. *Reading from file into ArrayList<MenuItem> - [10]*	5%
Menu functionality (purchasing coffees) 1. *Menu items not working, no exit option (hardcoded), no purchasing implemented - [0]* 2. *Working menu and partial/incorrect purchasing implemented (for example only CARD or incorrect change) - [5]* 3. *Working menu and full purchasing implemented (CASH/CARD) with correct change calculated - [10]*	10%
Storing transaction objects in ArrayList 1. *Transactions not recorded - [0]* 2. *Transactions partially/incorrectly recorded (for example written to file directly) - [5]* 3. *Transaction objects added to ArrayList<Transaction> for each purchase - [10]*	10%
Writing transaction ArrayList to file when the user Exists system 1. *No transactions written to file - [0]* 2. *Partial/incorrect transactions written to file (for example, incorrect data fields, separate files used, incorrect file format/CSV formatting, file overwritten) - [5]* 3. *Transaction ArrayList<Transaction> appended to file in correct format when used exits - [10]*	10%
Use of Object-Oriented techniques (classes, objects, inheritance) 1. *Monolithic code, problem not decomposed into separate classes/methods, no inheritance used - [0]* 2. *Minimal classes, methods but no inheritance - [5]*	10%

3. *Problem broken into appropriate classes/methods and inheritance used for Transaction objects with toString() for file output (CASH/CARD) - [10]*	
User input validation *1. No user input validation (input type mismatch and value mismatch i.e minus values allowed) - [0]* *2. Partial/incorrect user input validation - [5]* 3. *All input validated correctly with meaningful error message and retry - [10]*	10%
Overall code style and quality (variable naming, indentation, comments) 1. Sloppy or no indentation, poor class/method/variable names, little or no code comments - [0] 2. Readable code but lacking style consistency - [5] **3.** Consistent readable/indented code with meaningful class/method/variable names with appropriate level of code comments - [10]	10%
Software demo (ability to answer questions) 1. Unfamiliar with program functionality, cannot locate code, cannot explain code - [0] 2. Familiar with code, can locate code functionality, clearly own work but lacking in-depth explanation - [10] 3. Very familiar with code, clearly students own work, and can provide in-depth explanation of code functionality - [20]	25%
Additional functionality Marks awarded for additional functionality added by student. For example, additional code added to do something interesting with the transaction data, tracking stock levels etc.	10%
Total	