3.46 Assessment

Work in progress. Please do not use for assessment, yet!

Introduction

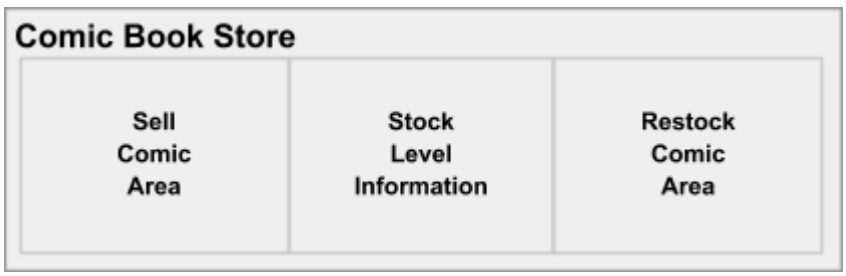
Write a program for a comic book store to keep track of stock for comic books. The program allows the user to sell and restock a comic, updating the stock levels of the comics accordingly.

This is an individual assessment activity. You will be given 8 classes of in-class time to complete the planning of the program, and to complete the programming component. You are allowed one sheet of A4 paper of handwritten notes to help you through this assessment.

See Appendix 1 for planning guide.

Interface Structure

Here is a rough guideline for how the interface can be structured.



Program Details

- The comic book stores the following comics:
  - Super Dude - Starting with 8 in stock
  - Lizard Man - Starting with 12 in stock
  - Water Woman - Starting with 3 in stock
- The user should be able to sell a comic one at a time, reducing the stock by one.
  - The interface should notify the user if the comic has been sold successfully.
  - The interface should notify the user with an error message if the comic has not been sold if there is not enough stock.
- The interface should display:
  - The number of comics sold.
  - The current stock levels of all comics (at once). If the stock levels change at any point, the interface should update.
- The user should be able to restock a chosen comic.
  - The user should be able to input how many copies the comic is being restocked with. For example, restock 10 copies of Super Dude at once.
  - There is no limit to the amount of comics the store can stock.
- The program should display relevant error messages for appropriate situations.

Note: There does not need to be any functionality for the user to add a new comic book to the program. However available comic books should easily be editable by editing the program code.

Final Submissions

At the end of the allotted time, hand in the following for assessment:

- your planning document
- a print-out of your final source code
- an electronic version of your source code
- screenshots of example output with annotations

Assessment schedule

	Required Elements	**Evidence**
Achieved	Designing and implementing a program that includes variables, an indexed data structure (a list), and a modular structure including details of the procedural structures of the modules.	Has a working program.
	Including a working graphical user interface with different sources of event generating components and event handling.	Program has a functional graphical user interface.
	Using classes and objects to encapsulate data and methods.	Evidence in program code.
	Setting out the program code clearly and documenting the program with comments.	Evidence in program code.
	Testing and debugging the program to ensure it works on a sample of expected input cases.	Evidence in program code and planning.

	Required Elements	Evidence
Merit	Using well-chosen modular and procedural structures, scope and encapsulation for data and methods, graphical user interface and event handling mechanisms.	Evidence in program code.
	Documenting the program with variable and module names and comments that accurately describe code function and behaviour.	Evidence in program code.
	Following a disciplined design and implementation process.	Evidence in planning and program.
	Develops with documented cycles of incremental development.	Evidence in planning and program.
	Implements comprehensive testing process, to ensure that the program works on inputs that include both expected and boundary cases.	Evidence in planning document.

	Required Elements	Evidence
Excellence	Ensuring that the overall modular and procedural design, graphical user interface, and event handling design, are a well-structured, logical decomposition of the task, and that the program is flexible and robust.	Evidence in planning and program.
	Setting out the program code concisely and documenting the program with comments that explain and justify decisions.	Evidence in planning and program.
	Comprehensively testing and debugging the program in an organised and time effective way to ensure the program is correct on expected, boundary and invalid input cases.	Evidence in planning and program.

Appendix 1: Planning Guide

Task 1: Identify user inputs

What program functions can the user trigger through the interface?

Task 2: Identify information to be displayed

What information will the interface need to display to the user?

Task 3: Sketch interface design

Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.

Task 4: Identify any classes required

Explain what the class will represent, plus listing what information will be stored in the class and any functions the class will have.

Task 5: Identify any constants or existing data if required

Task 6: Identify indexed data structures

Task 7: Determine what calculations are necessary

Write out the calculations the program will have to compute.

Task 8: Develop a modular structure for your program

Describe any functions that the computer program will have, identifying any sub-functions where required.

Task 9: Define the functions identified

Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.

Task 10: Document test cases for testing the program

Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.

Task 11: Refine the plan

Note any modifications here when iterating through the development cycles.