Swinburne University of Technology
School of Science, Computing and Emerging Technologies

SWE30003

Software Architectures and Design
Semester 1, 2025

Assignment 3 – Object Design Implementation and Reflection

(Group Work; Worth 25 marks)

Due: Electronic submission (via Canvas):
Normal (preferred): 11.59pm, **Sunday 1 June, 2025**;
Extended (no penalty): 11.59pm, **Friday 6 June, 2025** (no later submissions).

Detailed Design and Implementation:

In Assignment 2, you were given the task to come up an initial object-oriented design of *An Online Electronics Store*. Based on that initial design, in this assignment (Assignment 3), you are required to (1) carry out a detailed design and implementation for the system, and then (2) reflect on the initial design based on your experience with the detailed design and implementation.

Here are a few *ground rules* for this assignment:

(1) Detailed Design and Reflection

- Refine and extend the initial high-level object-oriented design from Assignment 2, to arrive at a detailed design suitable for direct object-oriented implementation. This may include, but not limited to, refining/adjusting the existing design, adding UI design and adding database design (in an object oriented manner when appropriate).
- You are also allowed to further change the design if, during the implementation process, you find that the initial or extended design is incomplete, incorrect, or inadequate for the problem at hand.
- All changes to the original design (from Assignment 2) during the detailed design stage and the implementation stage *must be justified and documented accordingly*. Please note that changes, in general, may not only affect the static structure of your design (i.e., the class diagram or similar), but also affect class responsibilities, the bootstrap process, and/or interaction patterns/scenarios at runtime.
- In addition, you must provide a *reflection* on the quality of your initial design from Assignment 2. This may include, but is not limited to, the following:
    - Which aspects of the problem did your initial design address adequately?
    - Which aspects were missing from your initial design?
    - What errors were introduced in the initial design?
    - How much interpretation of the initial design was required (i.e., how *ambiguous* was the initial design)?
    - How did you change the initial design to address the omissions, errors and/or ambiguities (if applicable)?

  - ***Lessons learnt***: Given the experience gained from your detailed design and implementation, how would you tackle a high-level initial OO design problem differently next time?
  - ***Architecture Style(s)***: What is the architecture style(s) of your system design? Explain and discuss the specific components, connections and constraints between them, according to the identified style(s). Note that the architectural components are at a higher level of abstraction than objects/classes, ie, a component may consist of more than one class.

(2) Implementation

- The implementation can be completed in **an object-oriented language** *of your choice*. The source code of your implementation must be included in your submission.
- No particular coding standard is prescribed for use in your implementation, but make sure that your code **follows a 'standard'** required for professionally developed software (and provide a reference) – part of the marks will be allocated to the use of a suitable coding standard!
- As a simplification, the implementation **does not** need to support payment options as we cannot have a banking system to validate transactions. The implementation must allow the users to carry out their activities (except for the actual payments, for which some simple message will be sufficient).
- To further simplify the implementation task, you may choose to implement part of the system (instead of the whole system). However, the implementation needs to cover **at least four (4) areas of business operation fully** and **state them clearly in your submission** (see the case study description for Assignment 1); in this regard, you need to consider the dependency between these implemented areas, so that the chosen areas are fully functional, even when having dependency on other areas.
- A simple user interface (graphical or textual) will be sufficient, but please note the usability requirements identified in Assignment 1. It may help to label/mention each of the input requirements clearly and provide appropriate validation on inputs. For example, name field cannot be blank, and correct data type should be used.
- If preferred, files may also be used for persistent data storage, instead of using databases.
- You may develop the system as a web-based application or a separate desktop/mobile application.

(3) Execution and Operation

- Use a number of scenarios to demonstrate ***all the (implemented) system capabilities***. Provide brief descriptions of how the user(s) can enter various information required for the different scenarios and steps, including various options (if applicable).
- To demonstrate that your implementation ***works correctly***, for each scenario, provide screen shots of your application that, amongst others, illustrate (i) an 'empty' UI (graphical or textual) at the beginning of the scenario, (ii) takes correct input, (iii) validation of incorrect input, (iv) change or deletion of input when the customer/user had a change of mind, and (v) successful completion of the scenario. (For the payment step, only need to give some message indicating that it has been processed, and no need to carry out actual payment processing). *Note that instead of using screen shorts, you may record a short video that shows all the required scenarios and steps with narration in a structured way.*
- You must explicitly state which *platform* (operating system, IDE etc.) you used for development and testing of your implementation, and provide a brief description of how your program is to be deployed and run.

- *You must provide evidence* that your implementation compiles, runs, and correctly follows the requirements. Only in exceptional circumstances will a marker re-compile and run your code. It is your responsibility to provide, in the submission, ***sufficient evidence*** that the code compiles correctly, the system accepts only correct inputs, and the relevant user requests can be processed as expected.

Note that **any clarification questions** should be posted to the Assignment 3 discussion forum on Canvas, and emailed questions will not be answered unless it is of a personal nature.

Marking Guidelines:

This assignment will be assessed using the following guidelines (120 points):

- Detailed object-oriented design with detailed description (including justification) of any refinements, extensions, changes and non-changes made to the original design from Assignment 2 (*30 points*)
- Discussion of the quality of the original design from Assignment 2 (*20 points*)
- Lessons learnt from the detailed design and implementation experience (*10 points*)
- Architecture style(s) of your system design (*10 points*)
- Completed implementation (max *50 points*)
    - Source code and executable code, including suitable coding standard (20 points)
    - Evidence of compilation (5 points) and correct execution (25 points)

Please note that **if**
(1) *no implementation, an obviously incomplete implementation, or a significantly incorrect implementation* is provided in your submission,
(2) the original design from Assignment 2 (ie, the whole assignment 2 submission) is not included (as basis of discussion and self-contained-ness), **or**
(3) the changes to the design are not adequately justified and documented,
zero marks will be given to the design, discussion and reflection parts of the assignment, and the implementation will be judged based on its merits.

A detailed mark sheet will be provided on Canvas.

Submission Details:

The submission is in electronic form only. An ***electronic copy*** of your solution must be submitted through Canvas by the due date and time (see the front page of this specification). It must include all the items that are specified above (see also the marking sheet). The submission should have four parts:

1. The source code and executable code of your solution must be packaged in a single ZIP or RAR file (or equivalent), and
2. The remaining parts (see the marking guidelines above, i.e., excluding the source code and executable code) are in a single pdf document, including your assignment 2 submission as an appendix. In preparing these parts of your submission, you need to **orgarnise/structure your submission according to the marking guidelines/sheet, and answer the relevant aspects as stipulated in the marking guidelines/sheet specifically (and in the assignment specification, in general)**. For example, for the second part (detailed object oriented design), you need to include the final class

diagram capturing your final detailed OO design (in readable form, when printed), and provide the change details and justifications for the class level, the responsibility level and the dynamic aspects (bootstrap and scenarios), with corresponding headings. For the sixth part, a video can be used as the evidence of compilation and correct execution.

3. The appropriately completed and signed "Assignment and Project Cover Sheet" declaration form.
4. **"A contribution document"**, signed by *all* group members, which
   a. lists the amount of time spent by each member on each significant part of the assignment,
   b. describes briefly and specifically the contributions made by each group member, and
   c. provides evidence showing that the assignment is done through *true* group collaboration, e.g., discussions and group reviews of all major parts of the assignment, and simple group meeting minutes (time, place, attendees, issues discussed, decisions made, etc).

Note: For this assignment, students are to work in groups of three or four, i.e., the same groups as in Assignment 2. Permission by the Unit of Study convenor is required to *change groups*, **well before** the submission deadline with a "good" reason. Extensions to the submission deadline can only be granted for genuine reasons and the Unit of Study convener must be contacted *at least 48 hours* prior to the submission deadline or the earliest possible time.

Also note that to allow more time for this assignment, its deadline has been **pre-extended by 5 days without penalty**, while students are still encouraged to submit by the earlier "normal" week 12 deadline. Consequently, no further late submissions than the extended deadline will be allowed (unless the Unit of Study convener has approved a further extension). That is, any late submission beyond the extended deadline will be given a zero result and *no* feedback may be given on the respective submission.

The Unit of Study convener and/or the tutor reserve the right to call in any student/group to further explain and demonstrate their submission if there are doubts about the authorship of the presented solution and/or the completeness thereof.

Swinburne University of Technology

School of Science, Computing and Emerging Technologies

ASSIGNMENT AND PROJECT COVER SHEET

Subject Code: SWE30003 Unit Title: Software Architectures and Design

Assignment number and title: 3, Design Implementation Due date: 1 June 2025

Tutorial Day and time: ____________________ Original Project Group:________

Lecturer: Prof Jun Han, Tutor:_________________

To be completed as this is a group assignment
We declare that this is a group assignment and that no part of this submission has been copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for us by another person.

ID Number	Name	Signature

Marker's comments:

Total Mark:_________________

Extension certification:

This assignment has been given an extension and is now due on ________________________

Signature of Convener:____________________