****

SC2002: OBJECT ORIENTED DESIGN AND PROGRAMMING

AY 24/25 SEMESTER 2 GROUP PROJECT

**Build-To-Order (BTO) Management System**

Date of submission: 22 April 2025

**Declaration of original work**

We hereby declare that the attached group assignment has been researched,

undertaken, completed, and submitted as a collective effort by the group members

listed below.

We have honoured the principles of academic integrity and have upheld the Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no

marks will be awarded for the assessed work. In addition, disciplinary actions may be Taken.

| Name | Course | Lab Group | Signature / Date | Role |
| --- | --- | --- | --- | --- |
| Josquin Ang JuanZhe (U2421048J) | CSC | FDAB - Group 3 | 21/4/25 | Project manager  Lead UML designer  Lead developer |
| Kieran (U2421311C) | DSAI | FDAB - Group 3 | 22/4/25 | Lead developer |
| Koh Hui Hsin Jacie (U2230688C) | SoH | FDAB - Group 3 | 22/4/25 | Report manager |
| Hao Yize (U2420464D) | DSAI | FDAB - Group 3 | 22/4/25 | UML designer |
| Katrina Paulson (U2421374H) | CSC | FDAB - Group 3 | 22/4/25 | UML designer |

Chapter 1: Requirement Analysis & Feature Selection

1.1 Understanding the Problem and Requirements

Our team started the project by reading the assignment document in detail, extracting the main ideas of the project and our final product requirements. Our product needs to assign the correct permissions for users to view, apply for, and manage BTO projects.

The explicit requirements come in 2 main parts: system requirements and access requirements. System requirements pertain to the functionalities of the system, while access requirements focus on the permissions granted to its different users.

1. System Requirements

Main system must manage the users and their access to the BTO project information.

* Allow for login and authentication, and changing of password: On startup, the system should load the database information from the provided csv files.
* Allow for filtering of projects output when requested by the user: Some filters include age, marital status, and type of flat chosen for applicants.

1. Access Requirements

3 main types of users with different access levels: Applicant, Officer, and Manager.

1. Applicant

View and apply to projects that match their age and marital status, manage their application, and make enquiries. Applicants can track the status of their application. If successful, they can book only one flat via request to the HDB Officer. Applicants can withdraw booking requests.

1. HDB Officer

Inherit from Applicant because they have the same (but more) capabilities. Officer can answer enquiries, resolving booking requests and updating flat availability pertaining to their assigned projects. Officers cannot apply to a project if they had already applied for that same project, have registered for another project within the same application period, or are an officer for that applied project. Officers can also view projects open for officer positions regardless of marital status. They cannot edit project details. They are also able to generate receipts automatically upon approval of applicant’s booking details.

1. HDB Manager

Managers have the highest access. They can create, modify and delete projects, handle officer registration and approval of applicants' requests and withdrawals. Managers cannot be an applicant themselves for any project and can only manage one active project at a time.

Implicit expectations:

* Applicants should only be able to edit the description of their enquiry or delete it prior to receiving a response from the HDB staff.
* Locations of the available BTO projects were not provided to us, so our group took the initiative to create a dummy dataset for these locations

Ambiguous points:

* Brief did not indicate if applicants can modify flat type after submission, we followed the actual BTO way of not allowing changes after submission.
* Changing of age or marital status during runtime was not mentioned. Given the context that info comes from Singpass, the information given to us is the latest and will not change. If the applicant wants to change their booking, they must withdraw their current application and submit a new one.

1.2 Deciding on Features and Scope

Our team focused all of our efforts to delivering the core features as required:

* Login and authentication
* Role-based access control to the system features
* Applicant
  + Apply, book, and withdraw requests as well as make, edit, and delete enquiries
    - Viewing of projects automatically checks user particulars before showing only the projects they are eligible for
    - Can check the status of their currently active application
    - Enquiry actions are passed through the enquiry manager
      * Cannot edit the Enquiry after it has been answered
* Officer
  + Register to join project as an officer
  + View status of their registration
  + Apply for a flat as an applicant
  + Generate receipt for a project they are managing
  + Reply to applicant enquiries
  + View project details of their project
  + Update project status to “booked” for successful applications (when flat booking is completed);
  + Update number of available units per flat type (when applicant secures unit);
  + Update applicants’ profile with their selected flat type
* Manager
  + View, create, edit project listings
  + Approve/Reject officer applications to their projects
  + Generate filterable report for applicant bookings
  + Approve/Reject applicant booking requests
  + View and reply to enquiries

Optional features we implemented:

* Multiple filters can be applied to the databases at once
* Multiple filters can be applied to the project viewing at once

Our group decided that filtering projects by location and flat type would be optional. While this feature would enhance user experience, it is not essential to core system functionality.

One excluded feature is the ability for HDB Officers to withdraw their registrations to BTO projects. Should officers assigned to the project choose to withdraw late, there may not be officers to fill the vacancies before the registration stage closes. This could affect the project timelines as new officers will have to be reassigned to the project.

Chapter 2: System Architecture & Structural Planning

2.1 Planning the System Structure

We identified required data, validation rules, expected behavior, and system flow, then grouped theminto modules based on user responsibilities, which became the system’s core components. We were able to find these core logical components:

1. Login and Authentication;

Handles user logins via Singpass and authenticates the user ID and password against the system’s list, then stores the user’s age and marital status for eligibility checking;

1. Project Management Module;  
   Supports the creation, editing, toggling visibility, and deletion of BTO projects by managers. It also maintains the following project attributes: name, neighborhood, flat types, unit counts, application window, and assigned officers;
2. Application Processing Module;

Used by applicants to apply for eligible projects, as well as managers to approve or reject applications and withdrawals. It supports application status tracking, submission history viewing, withdrawal requests and eligibility checks;

1. Officer Registration & Project Assignment Module;

Allows managers to manage approval or rejections and conflict checks for officers’ registrations to projects. It also allows officers to register for available projects;

1. Flat Booking Module;

This module allows officers to update an applicant's status to "booked". It also allows them to track and indicate applicants’ selected flat types, edit the flat availability count, as well as generate booking receipts with applicant and project details;

1. Enquiry & Feedback System Module;

This module allows applicants to submit, edit (before reply), or delete their enquiries. Officers and managers can also use this to view and reply to enquiries (based on whether they have been assigned the applicant’s specific project);

1. Report Generation Module;

Used by managers to generate customisable reports on applicants booked/applied to their project – they can filter these reports by data collected such as flat type, age, marital status, and project.

2.2 Reflection on Design Trade-offs

We balanced simplicity with extensibility when making design decisions. For report generation, we considered using a factory design pattern for future customizability, implementing Generator objects within a factory class. However, we decided to prioritise simplicity instead: the added abstraction of a factory seemed unnecessarily complex given the limited customisation required now. Hence, direct instantiation of report classes would suffice. This achieves the current project requirements with room for future refactoring.

Chapter 3: Object-Oriented Design

3.1 Class Diagram (with Emphasis on Thinking Process):

Main classes were identified from recurring nouns and user roles in the problem description.

We cross-checked user roles and workflows with how the actual BTO system functions, helping us assign clear responsibilities to each class and distinguish the core user types. Finally, we identified class relationships based on object behavior, ownership, and lifespan:

* Aggregation (“has-a”): If one is made up of other but can exist independently;
* Composition: If one cannot exist without another;
* Inheritance (“is-a”): If one is a specialised version of another;
* Association (“has-a”): If one simply includes or utilises another.

Core classes and responsibilities:

1. User-Role Classes;

* Applicant: The base user capable of making applications and enquiries;
* HDB\_Officer: Has an “is-a” relationship with Applicant, therefore, extends applicant functionalities; however, it is also restricted to officer capabilities;
* HDB\_Manager: Most privileged user responsible for system-wide actions like approving applications, assigning officers, and managing listings;

1. System Functionality Classes;

* Project: Encapsulates the listing of flats and their attributes, in other words, it “has-a” list of attributes such as flat types and unit counts;
* Application: Represents the applicant's submission for a project, hence associated with Applicant and Project (but not an association class);
* Receipt: Indicates successful booking, only generated at the end of an application process;
* Enquiry: Allows applicants to ask questions and tracks responses;

1. Centralised Manager Classes;

These singleton manager classes manage the logic and actions of system-wide tasks:

* EnquiryManager: Stores and tracks the progress of applicant enquiries;
* ReceiptGenerator: Generates a receipt after a flat is successfully booked, and therefore, “has-a” reference to Application for application status;

1. Filter Interface and Subclasses;

Allow dynamic filtering for viewing objects. Polymorphism was used for the IFilter interface to all of the implementations. By applying the Strategy Pattern, we enabled runtime selection of filters without changing the core logic, making it easy to update and expand;

1. Data Storage and Management Classes;

Info is stored in Database classes that inherit from an abstract Database class. This applies the open-closed principle for scalability and extendability without modifying base Database class.

1. System Entry and Integration Classes;

MainApp and HDB\_System:

* MainApp: Launches the system and sets up the main components;
* HDB\_System: Brings together the different databases in *Data Storage and Management Classes* and handles communication between the user interface and the core logic of the application.

3.2 Sequence Diagrams (with Emphasis on Thinking Process)

1. Officer Project Registration;

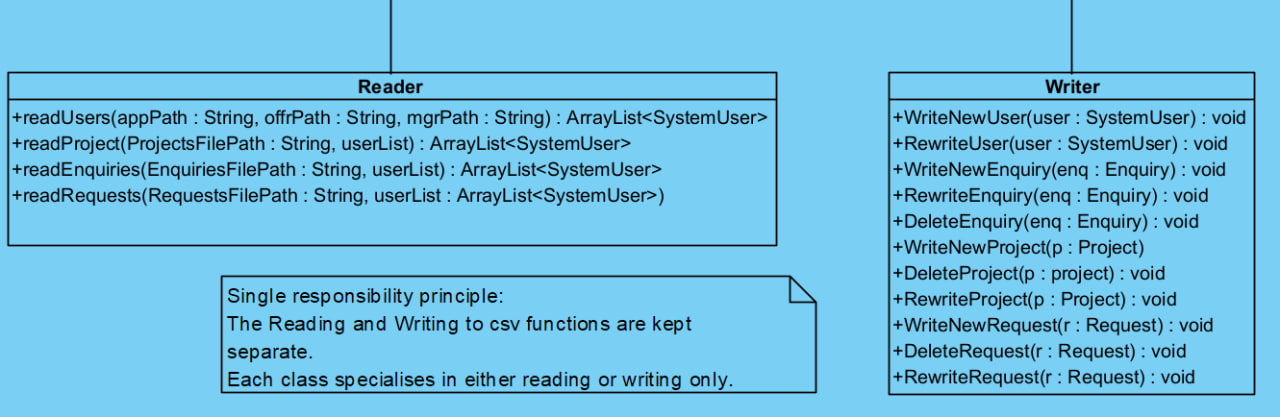
Cross-role logic unique to officers (applying to BTO projects), eligibility checks on projects applied to, and RequestDB. This shows how our system implements dual-role users with the correct role restrictions and maintains data consistency;

1. Applicant Enquiry;

Displays the end-to-end lifecycle of an enquiry, from creation to resolution – demonstrating the role-based reply permissions, as well as the validation logic of enquiry edits. This also helps us to communicate how our user-initiated features interact with the system functionality and centralised manager classes.

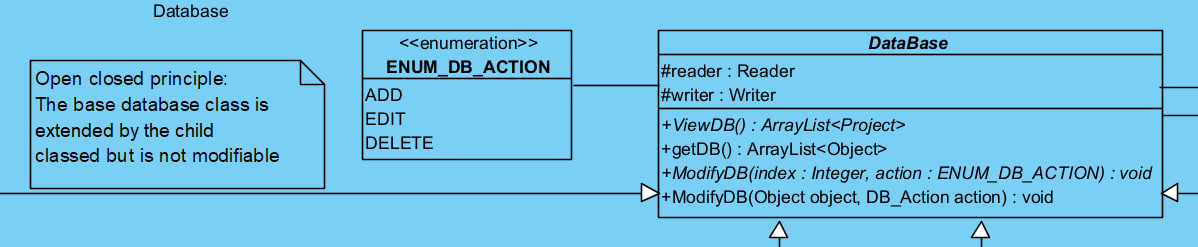
3.3 SOLID principles

3.3.1 Single responsibility principle: reader and writer to csv



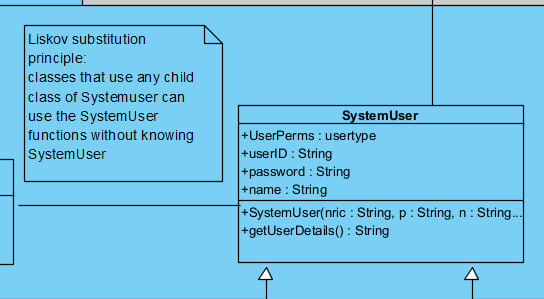
This separation made the code easier to understand and update, allowing for reading and writing to be changed without affecting one another.

3.3.2 Open-closed principle: Database is not modifiable, but functionality is extendable



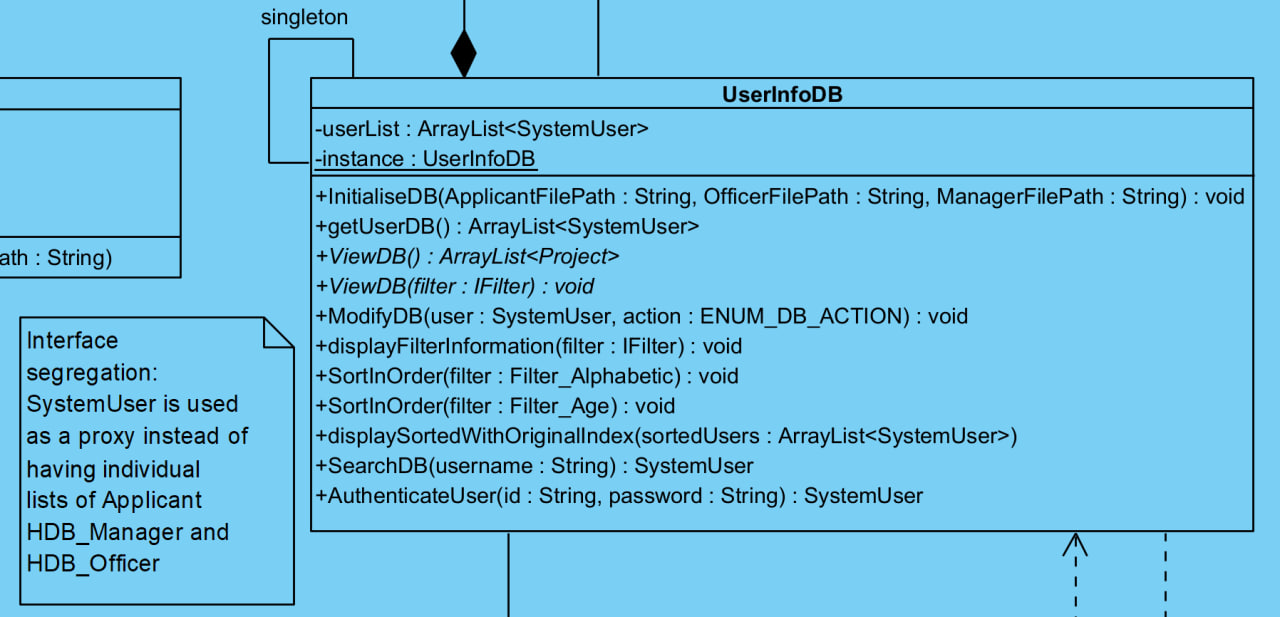
The extending of the Database class to implement its behaviours within subclasses allows us to add new types of databases without the risk of bugs in the core logic of the program.

3.3.3 Liskov Substitution principle: Children of SystemUser can still use its functions



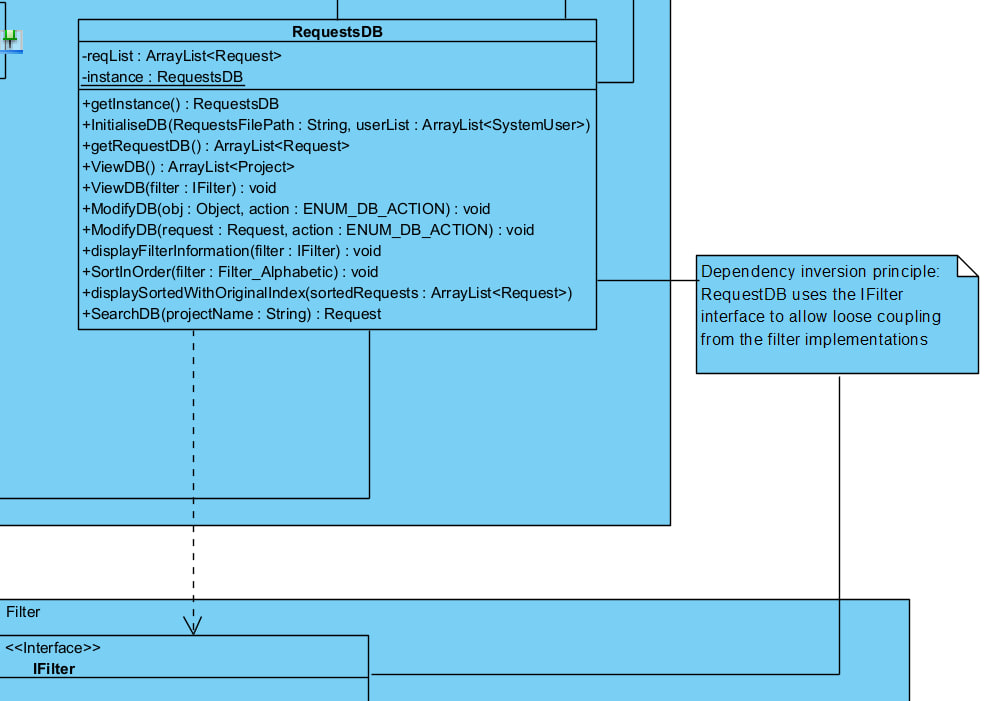
This allows the user subclasses to be used interchangeably whenever a SystemUser is expected, making the system more flexible and shared logic to work regardless of user type.

3.3.4 Interface segregation principle: SystemUser is used as the interface



Using SystemUser as the interface allows UserInfoDB to ally filters and perform operations generically through IFilter, which reduces dependencies within the system.

3.3.5 Dependency inversion principle:



Using the IFilter interface instead of specific filter classes keeps filtering logic modular and flexible, allowing for new filters to be added without changes to the database logic.

Chapter 4: Implementation (Java)

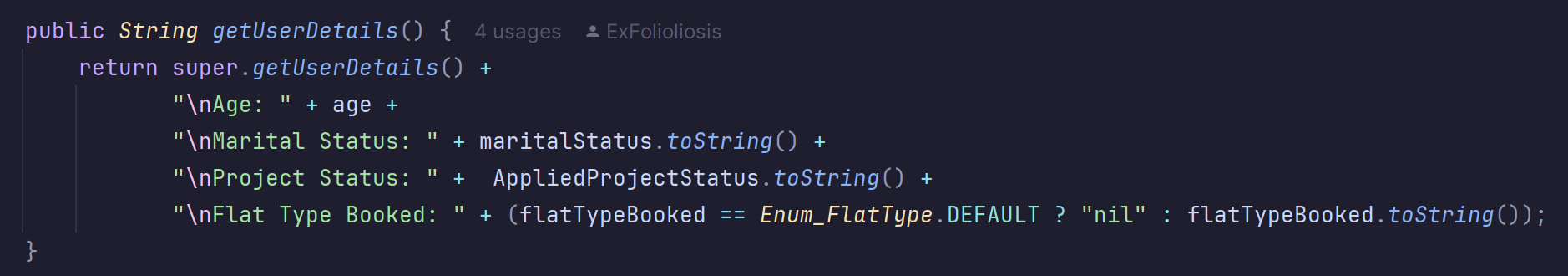
4.1 Tools Used

Java 17 for development, IntelliJ, Eclipse, and Visual Studio Code as IDEs, and GitHub for version control.

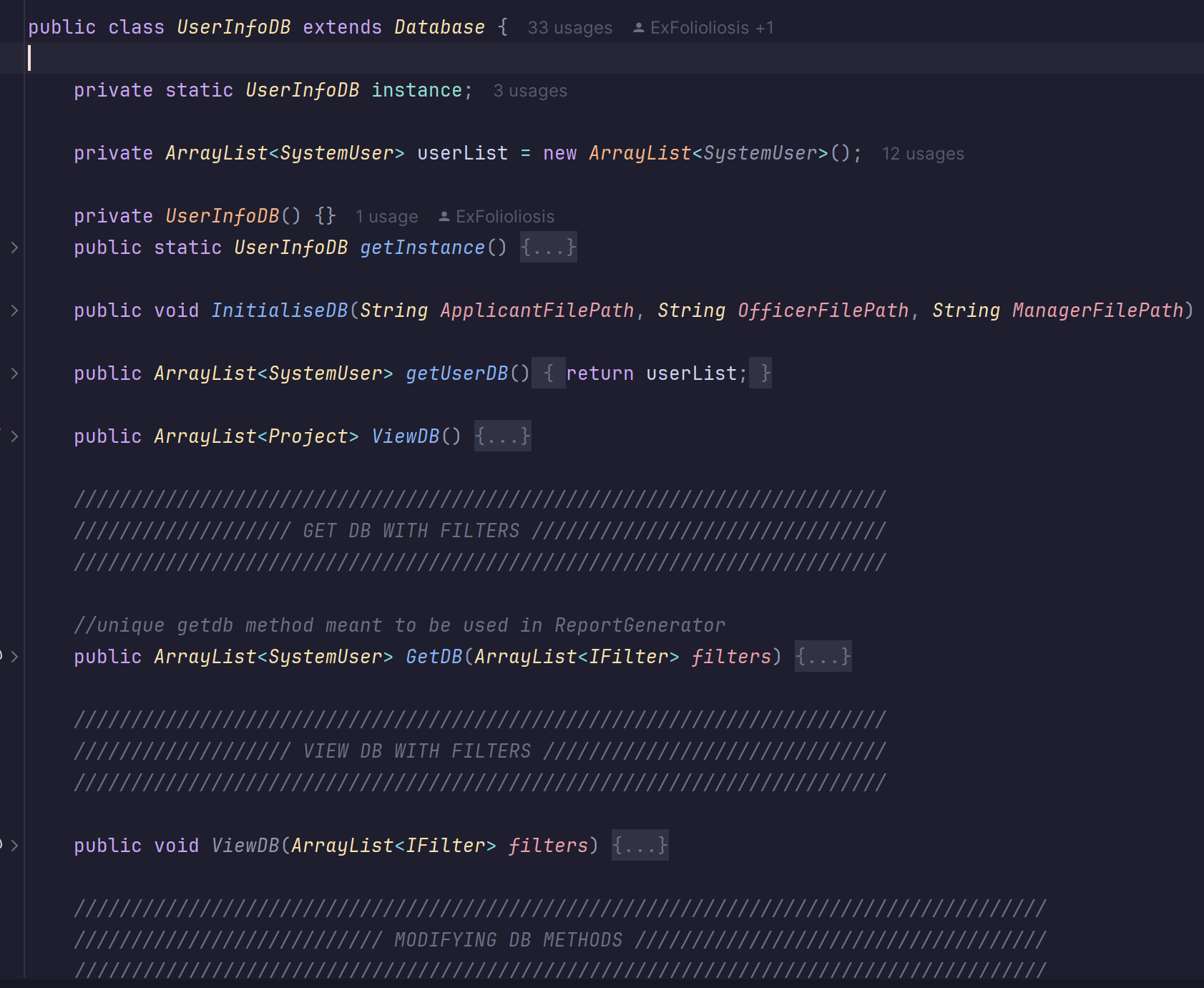
4.2 Sample Code Snippets:

This section details our code snippets relevant to the five requirements within the report brief.

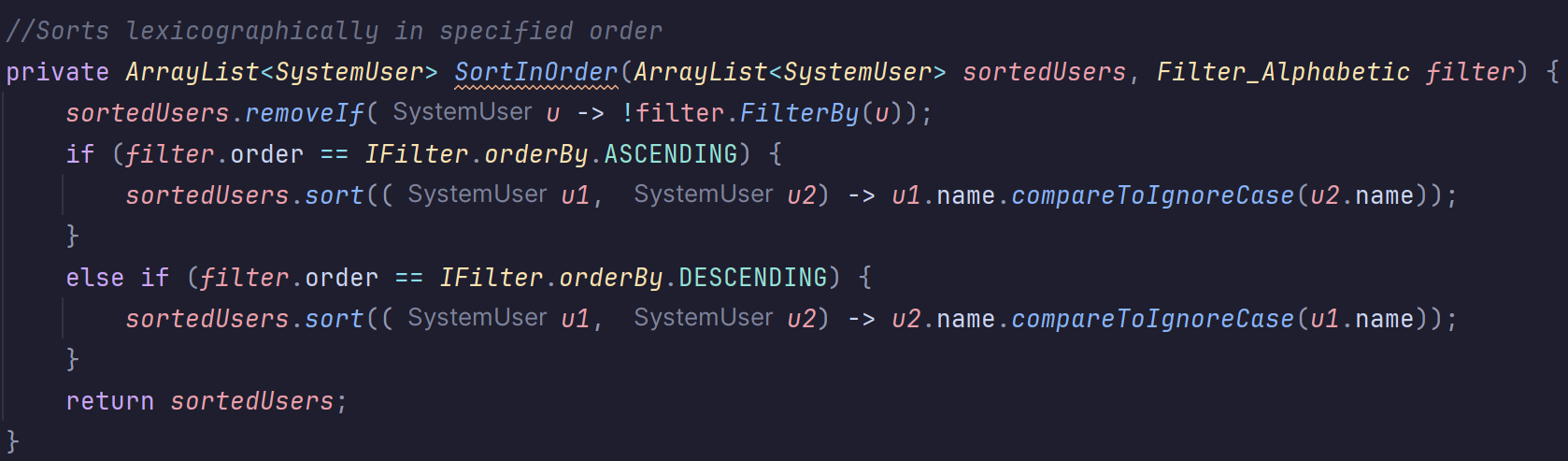
• Encapsulation: can only see the full user details by calling this function.



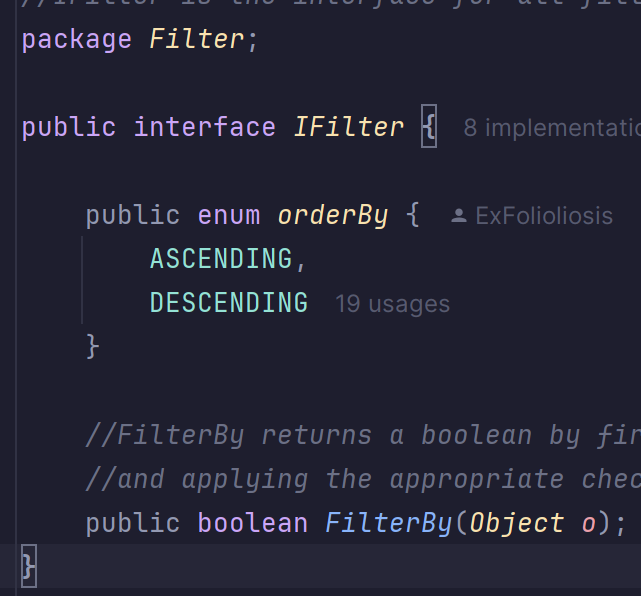
• Inheritance: UserInfoDB inherits all of Database’s methods and properties, with its own specialised behaviours.



• Polymorphism: uses SystemUser with FilterBy() from various filter types, demonstrating interchangeable behavior using a common interface.

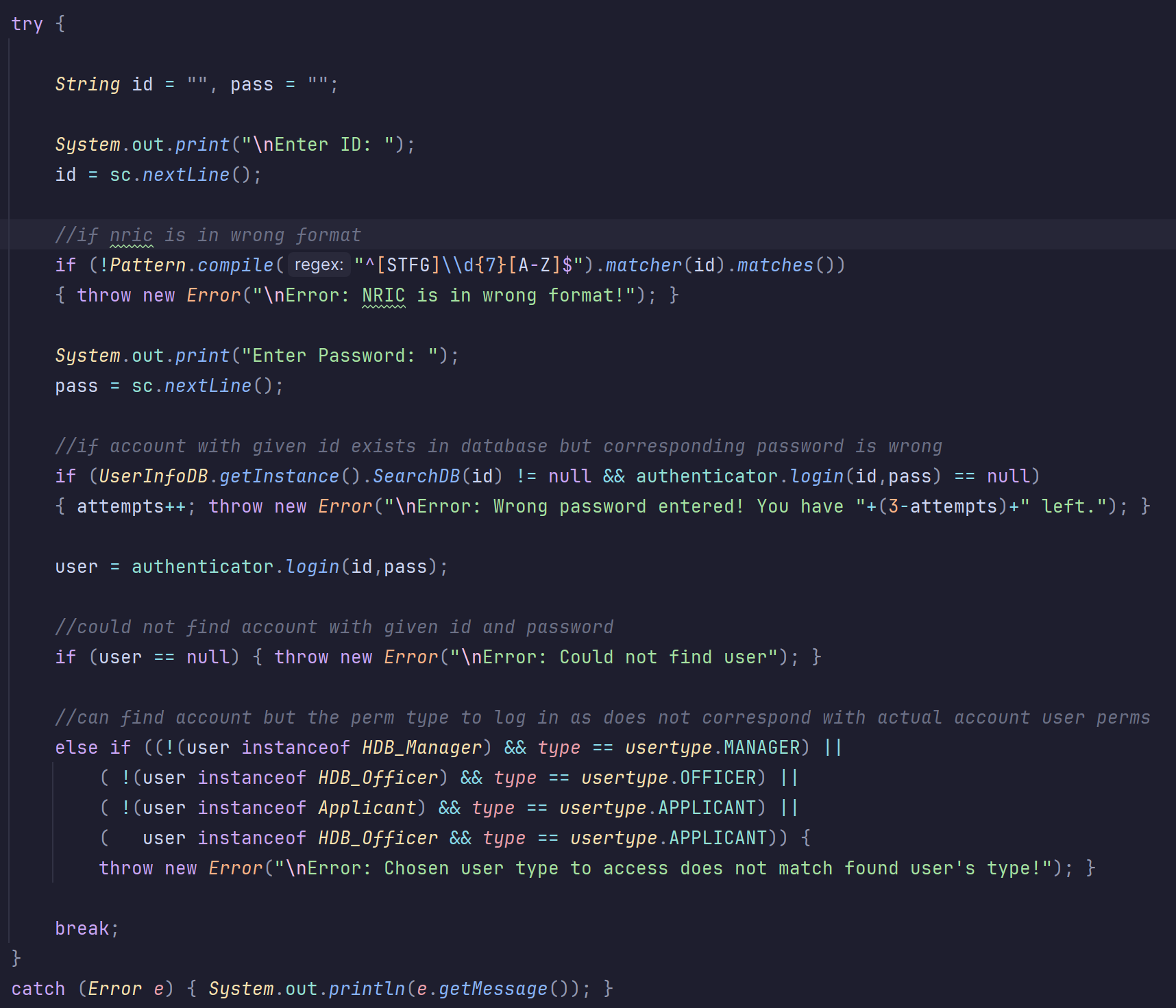


• Interface use: Filters like Filter\_Age implement IFilter, which enables interchangeable use via the shared FilterBy() method and orderBy structure.





• Error handling: login authorisation uses a try-catch block and custom error messages to catch invalid login inputs and guide user correction.



(Auth menu)

Chapter 5: Testing

5.1 Test Strategy

Unit testing done on all features before integration with other parts of the system

Follow the test cases to check for correct functionality

5.2 Test Case Table

| No. | Test case | Description |
| --- | --- | --- |
| 1 | User login | User can log in |
| 2 | Invalid login details | User cannot log in with wrong credentials |
| 3 | Password change | User can change password after logging in |
| 4 | Project display based on applicant’s marital status and project visibility(filterable) | Only show applicants projects they are allowed to apply for (order can be decided by filtering) |
| 5 | Apply for project | Only apply for projects they can see, and able to check details once approved |
| 6 | Applicant Enquiry | Able to add, edit, delete, view enquiries |
| 7 | One booking per application | Users can only get 1successful booking for each successful application |
| 8 | Withdraw | Able to submit a withdraw request |
| 9 | Officer application | Officer must pass all checks before they can apply for a project as an applicant |
| 10 | Officer joined project | Able to view details of the project the joined |
| 11 | Officer check request | Able to view their own request status |
| 12 | Officer View handled bookings | View bookings they have resolved |
| 13 | Update flat selection | Adjust the number of available flats |
| 14 | Reply to enquiry | Officer is able to answer enquiry |
| 15 | Manager Modify project listing | Manager is able to create, edit, and delete project listings |
| 16 | view other projects | View all projects |
| 17 | Manager handle officer request | Accept/deny requests to join a project |
| 18 | Manager handle applications | Accept/deny application, booking, and withdraw requests from applicants |
| 19 | Generate report | Generate a report of users |
| 20 | Manager Enquiries | View and reply to all enquiries |

6. Documentation

6.1 Javadoc:

All public classes/methods documented with Javadoc

6.2 Developer Guide:

How to set up the environment and build the project.

7. Reflection & Challenges

• What went well

Project and report was completed on time

• What could be improved

Some team members sometimes take very long to respond to messages. Most of the coding was done by 2 people. Quality of submitted diagrams and code was highly variable.

• Individual contributions

* Josquin
  + Class diagram, Sequence diagram, Code implementation, Version control management
* Kieran
  + Class diagram, Code implementation
* Yize
  + Class diagram, Sequence diagram, Code implementation
* Jacie
  + Class diagram, Report
* Katrina
  + Class diagram, Sequence diagram, Code implementation, Report

• Lessons learned about OODP

* Proper planning and consideration for interactions is important to reduce mental load and confusion later on in the implementation process.

8. Appendix

• GitHub link: <https://github.com/Dangomimikyu/SC2002_Group3_FDAB>