1. Abstract Classes & Interfaces

The User class is declared abstract to serve as a general template for all users (students, lecturers, officers), ensuring each must define their own transport request logic. Interfaces such as Serviceable and Trackable define behaviors common to all vehicles.

2. Encapsulation

All sensitive data (e.g., driver license, user details) is private and accessed via public getter/setter methods. This protects against unauthorized or invalid modifications.

3. Inheritance

Student, Lecturer, and TransportOfficer inherit common attributes and behaviors from the base User class. This promotes reusability and reduces code redundancy.

4. Polymorphism

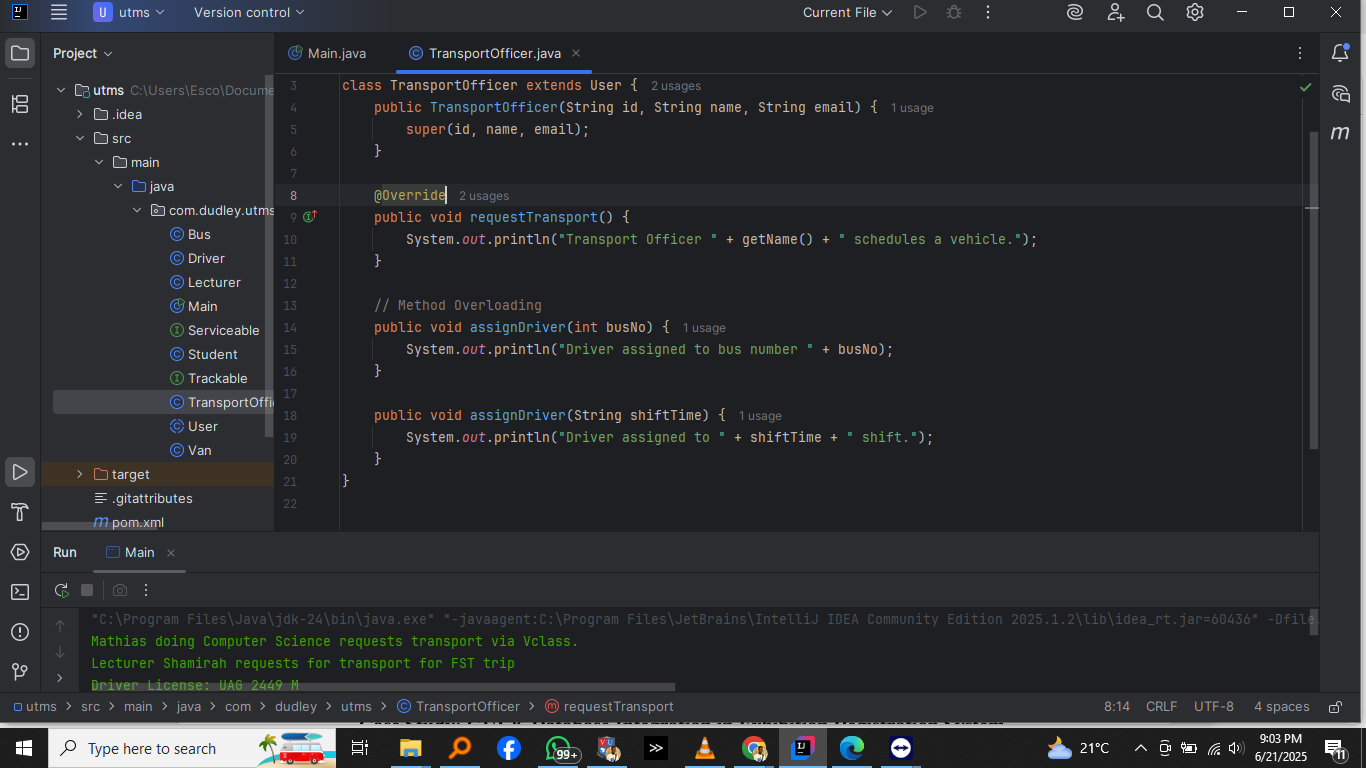
Overriding (requestTransport()) and overloading (assignDriver()) showcase both runtime and compile-time polymorphism. Different classes implement methods tailored to their role while maintaining a shared interface.

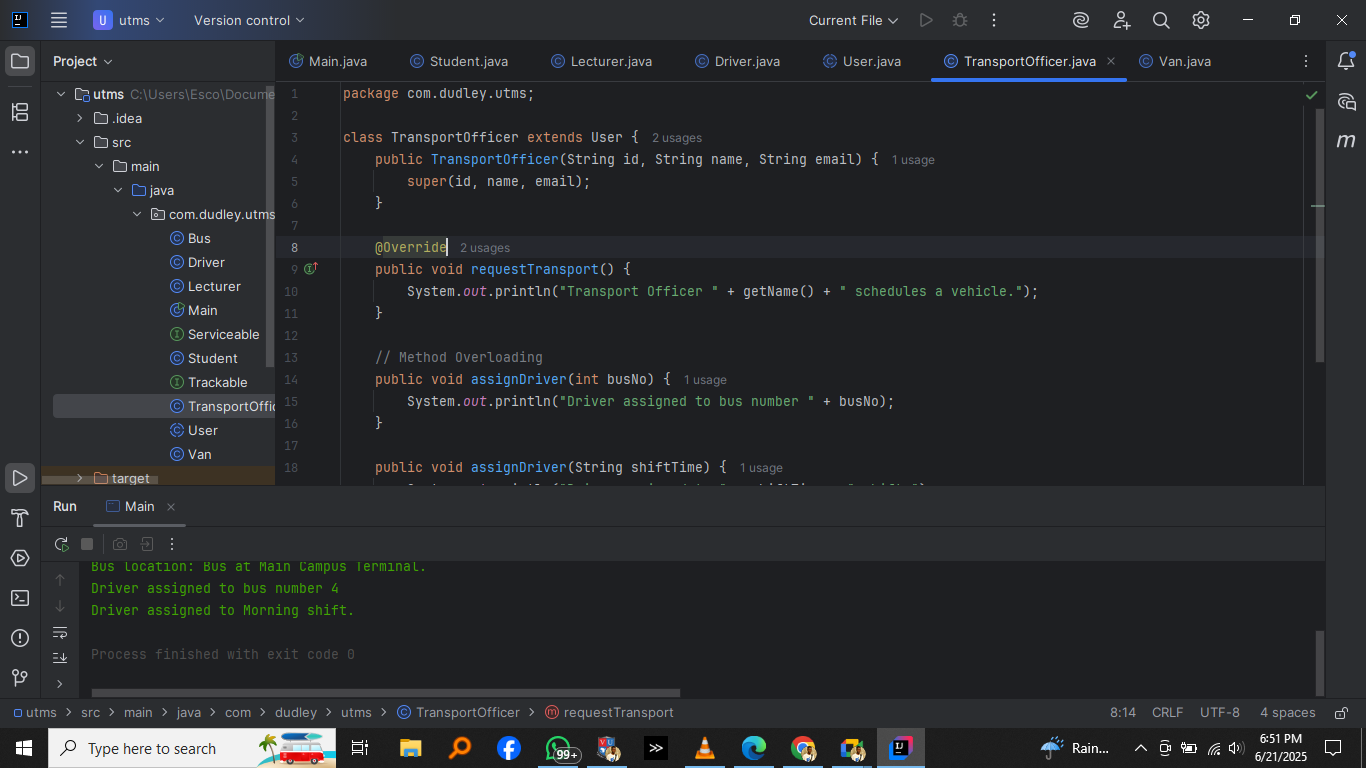
5. Interfaces & Vehicle Behaviors

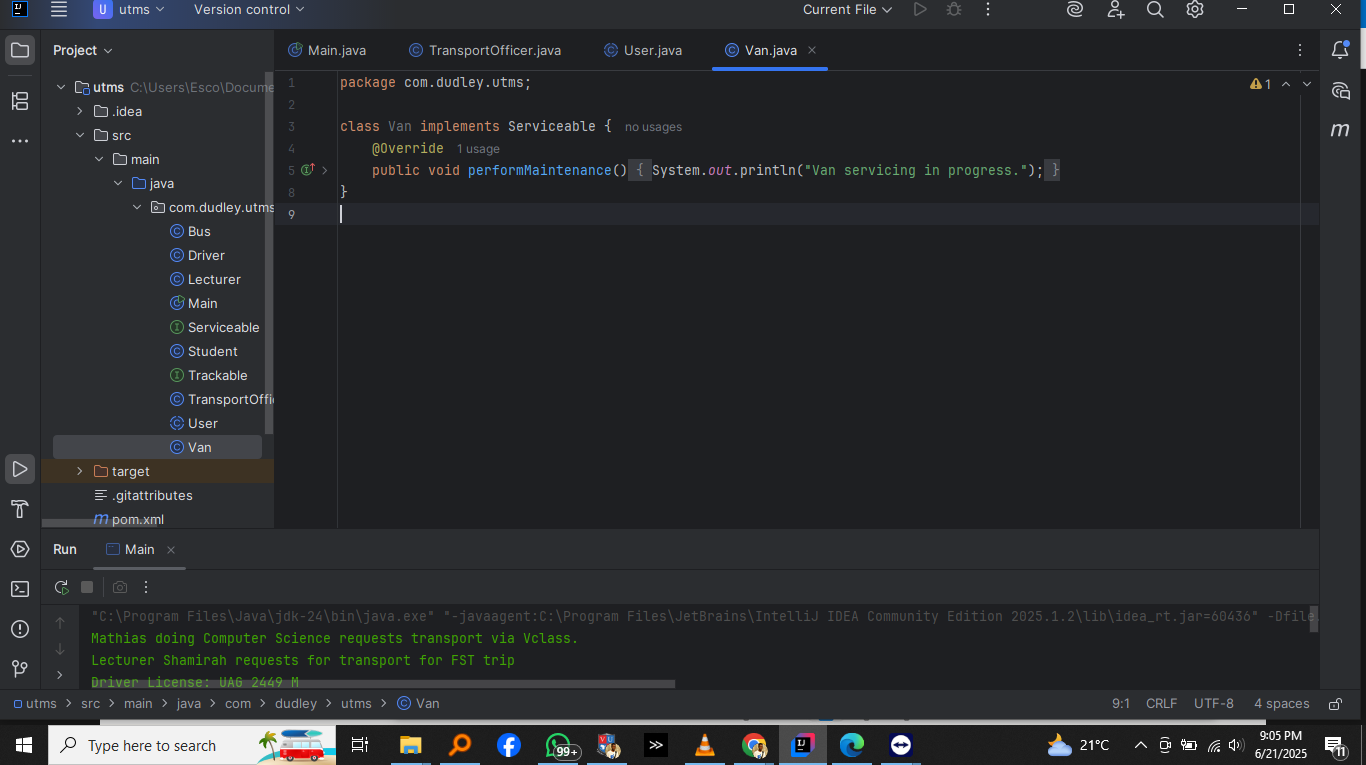
Bus and Van implement interfaces to allow standardized maintenance and tracking. This ensures different vehicle types can be managed uniformly using the same interface references.

6. Method Overriding & Overloading

Customized requestTransport() logic per user type improves realism and clarity. Overloaded assignDriver() adds scheduling flexibility based on time or vehicle type.







Link to github qn1 repository: https://github.com/dudleywamutanga/UTMS.git