**Group 3 -Cat 2 Semester Project OOP**

134455

145454

140156

145519

**REPORT**

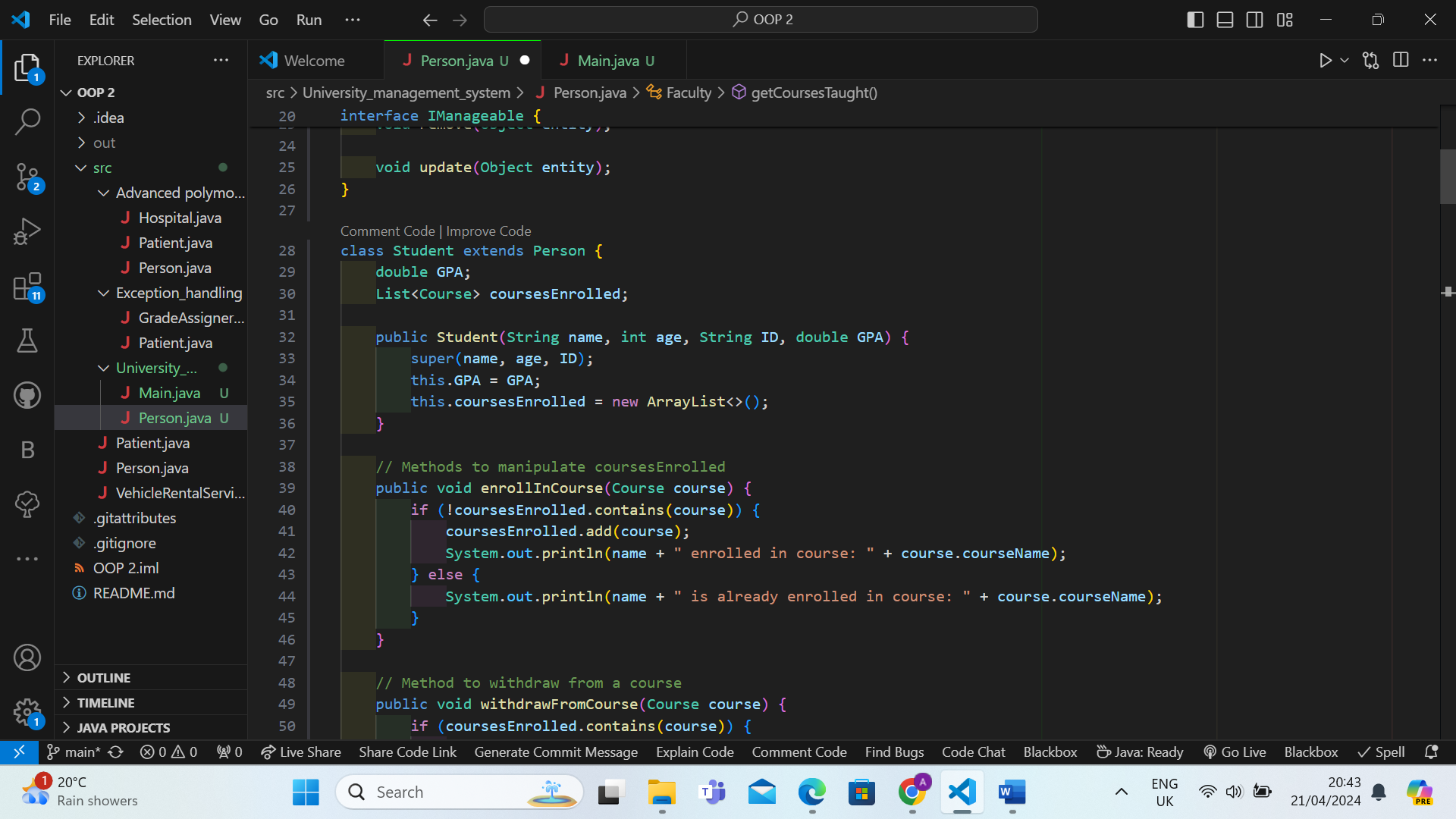
The goal was to create a simplified University Management System that effectively models the key entities within a university setting. We aimed to demonstrate proficiency in Object-Oriented Programming principles such as inheritance, abstraction, encapsulation, polymorphism, and exception handling. By designing abstract classes like Person and interfaces like IManageable, we aimed to provide a flexible and scalable architecture for managing students, faculty, courses, and departments. Through method overriding and overloading, we strived to achieve polymorphic behavior, allowing for diverse functionalities under a unified interface.

The following key OOP principals were implemented.

**1.Inheritance:**

Inheritance is evident in the relationship between the Person class and its subclasses Student and Faculty. Both Student and Faculty inherit common attributes such as name, age, and ID from the Person class.

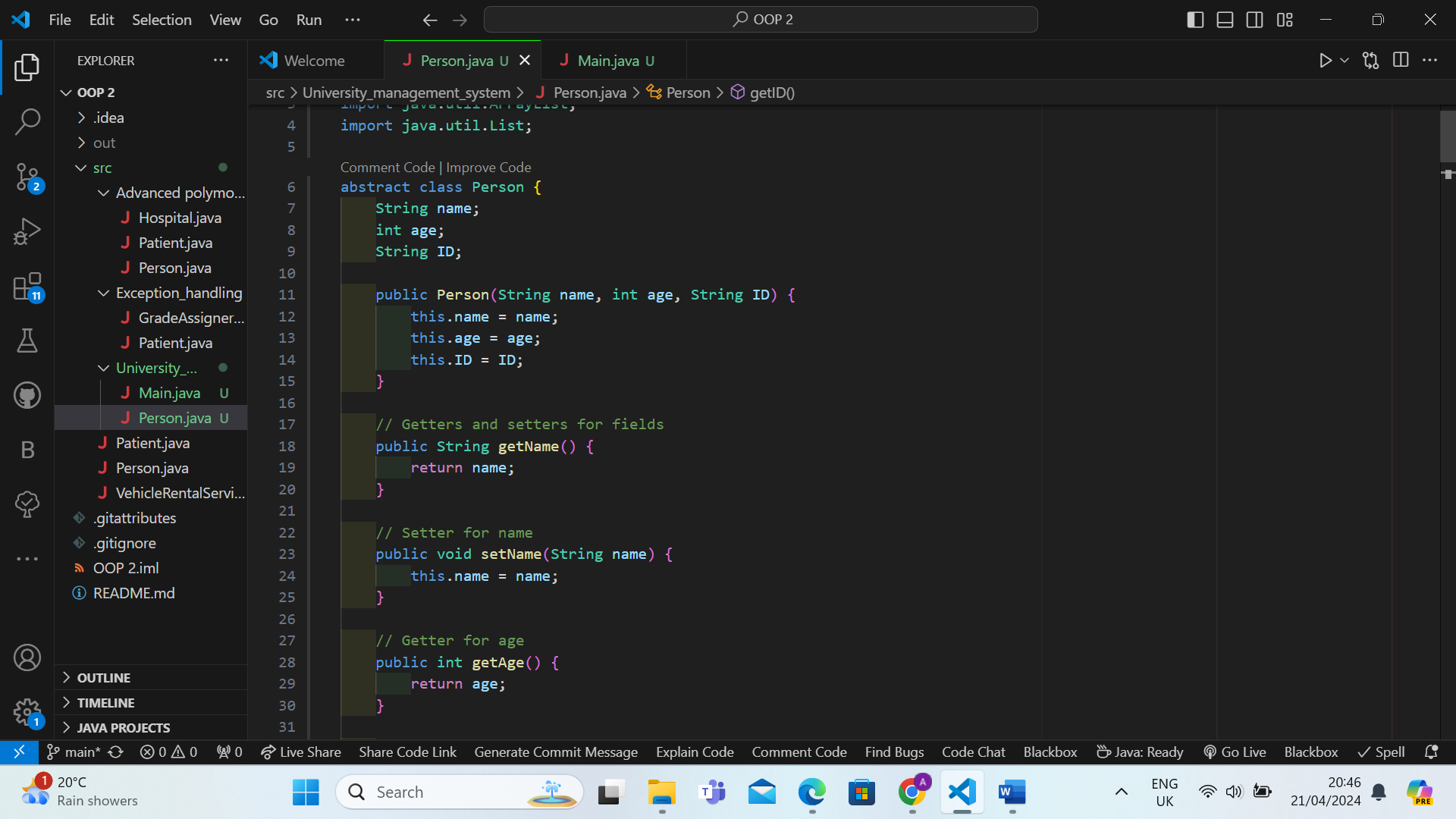
Example: Both Student and Faculty classes extend the Person class.



**2.Abstraction:**

Abstraction is achieved through the use of abstract classes and interfaces. The Person class serves as an abstract representation of a person in the university, defining common attributes and methods.

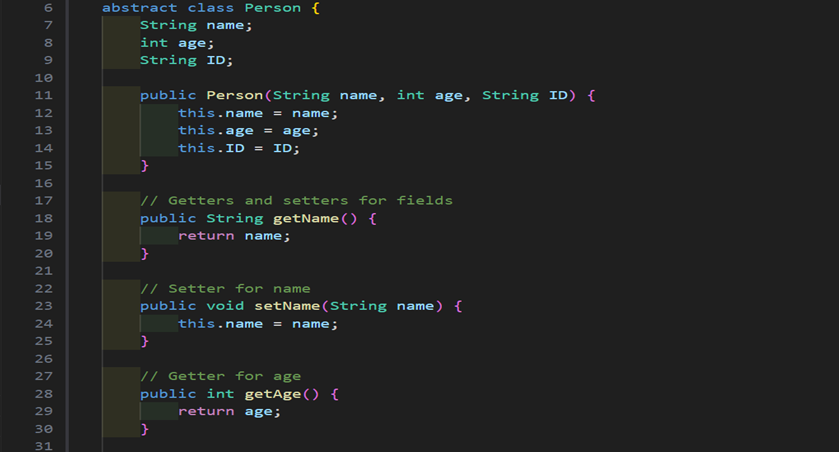
Example: The Person class is abstract and provides a blueprint for subclasses to implement specific functionality.



**3.Encapsulation:**

Encapsulation is achieved by encapsulating the data within classes and providing access to it through getter and setter methods.

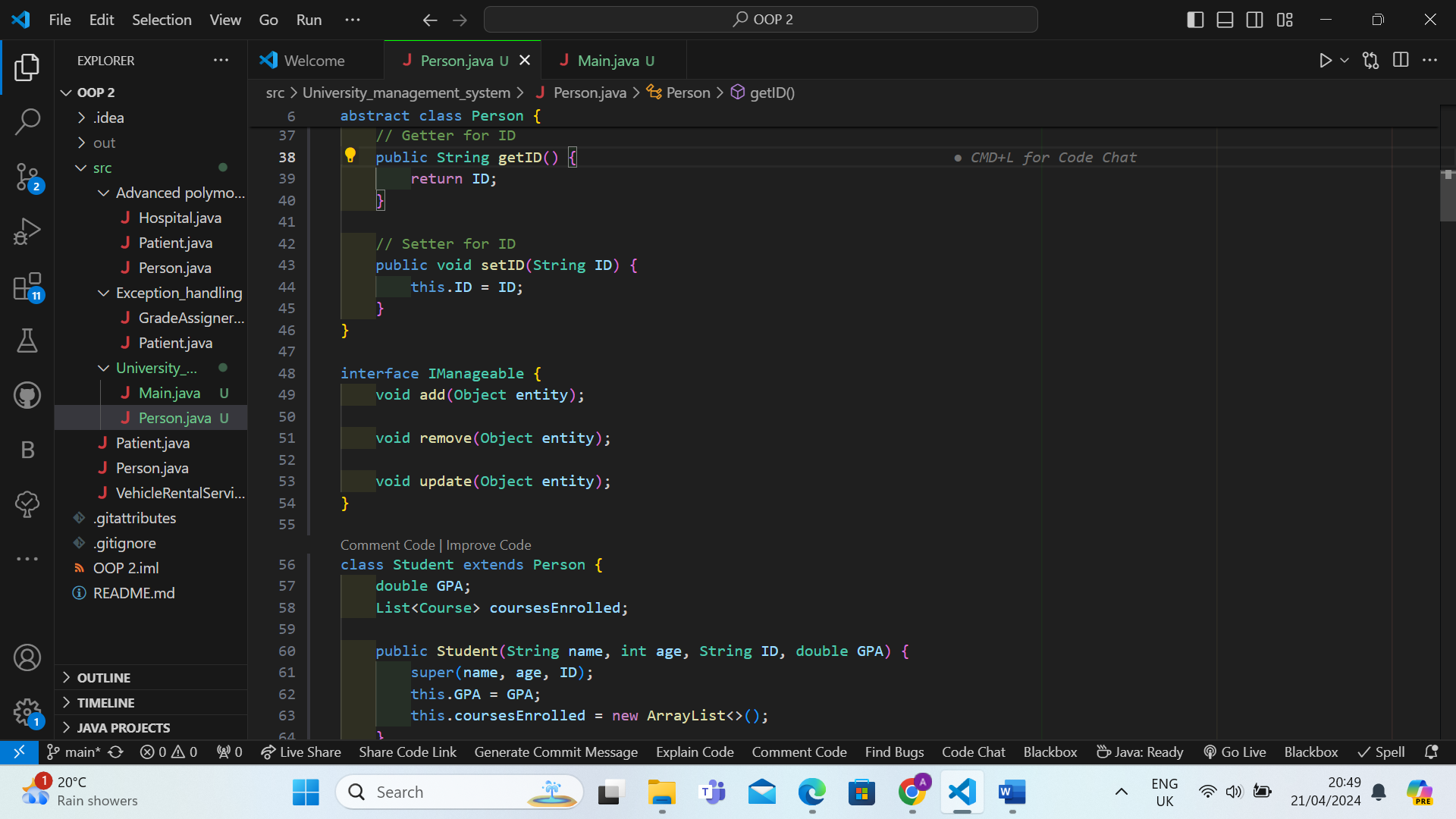
Example: The fields name, age, and ID in the Person class are encapsulated and accessed through getter and setter methods.



**4.Polymorphism:**

Polymorphism is demonstrated through method overriding and overloading in the project. Method overriding occurs when subclasses provide a specific implementation of a method inherited from a superclass, while method overloading occurs when multiple methods with the same name but different parameters exist within the same class.

Example: The add, remove, and update methods in the UniversityManagement class override the corresponding methods defined in the IManageable interface.



A screen shot of a computer

Description automatically generated

**5.Exception Handling:**

Exception handling is implemented to manage errors and unexpected situations in the code. Custom exceptions are created to handle specific scenarios, such as invalid data entries.

Example: The InvalidDataException class is used to handle exceptions related to invalid data entries.

