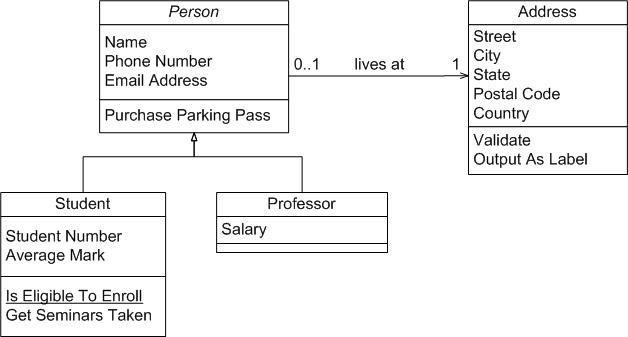
UTS QUESTIONS

OBJECT-BASED PROGRAMMING PRACTICUM

1. Identify the following diagram class, make complete improvements and in accordance with the rules for writing the diagram class.



**My Answer :**

1. Class Diagram Identification :
2. Person Class :

* Attributes :
* Name: String = Name of the person
* PhoneNumber: String = Phone number
* Email Address: String - Email address
* Methods:
* Purchase Parking Pass(): Function to purchase a parking pass

Relationship: Has a relationship with the Address class (a person may have 0 or 1 address).

The attributes in the Person class use the String data type because they store text, such as names, emails, and telephone numbers that can be written in various formats, for example "0812-3456-7890" or "+62 812-3456-7890"

1. Address Class:

* Attributes:
* street: String - Street name.
* city: String - City.
* state: String - Province/State.
* postalCode: String - Postal code.
* country: String - Country.
* Methods:
* validate(): to check if the address is valid.
* outputAsLabel(): to display the address in label format.

The Address class uses the String data type because street names, cities, states, and postal codes are stored as text. Postal codes are stored as Strings to support different formats, including those starting with zero.

1. Student Class (Derived from Person):

* Attributes:
* studentNumber: int - Student number.
* averageMark: double - Average mark.
* Methods:
* isEligibleToEnroll(): Checks if the student is eligible to enroll.
* getSeminarsTaken(): Gets the list of seminars attended by the student.

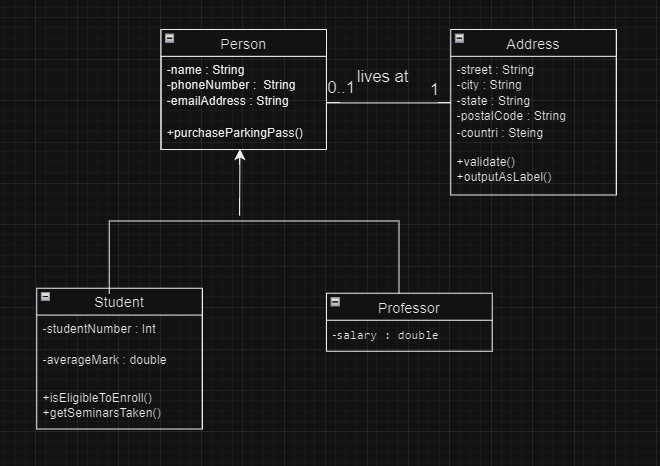
1. Professor Class (Derived from Person):

* Attributes:
* salary: double - Lecturer's salary.

1. Rule Based Repair:
2. Attributes and methods must be written with a lowercase letter at the beginning.
3. Modifier: Each attribute and method needs to have access (public/private/protected) added.

* Example: private String name, public void purchaseParkingPass()

1. The relationship between Person and Address is an association, because an Address still exists without having a Person.



1. Improved Class Diagram:
2. The Person and Address classes have an aggregation relationship, because a Person can have 0 or 1 addresses.
3. The inheritance of Student and Professor from Person is implemented with the concept of inheritance, where both inherit attributes and methods from Person.
4. Create a diagram class that uses multilevel inheritance and create the program code!

hierarchical inheritance in the context of a transaction system. We create a:

* a base class Transaction,
* an intermediate class TransactionOnline
* a derived class TransactionCardCredit.

1. Relasi Class
2. Transaction (Base Class)

* Attributes :
* transactionid: String
* amount: double
* Methods:
* processTransaction(): Processes transactions based on ID and amount.

Relations: This class is the base class for all types of transactions.

1. TransactionOnline (Intermediate Class)

Inheritance: Inherits from Transaction

* Attributes:
* paymentmethod: String
* Methods:
* validationPayment(): Validates the payment method.

Relations: This class is derived from Transaction11 and adds functionality for online transactions.

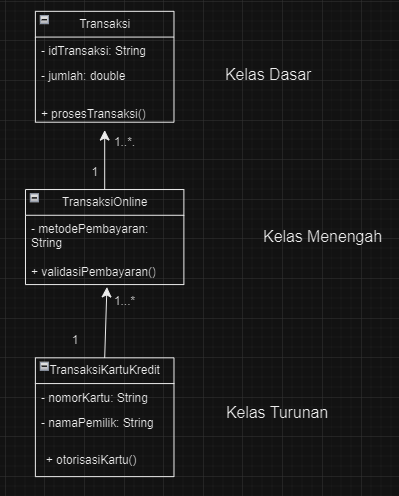
1. CreditCardTransaction (Derived Class )

Inheritance: Inherits from TransactionOnline

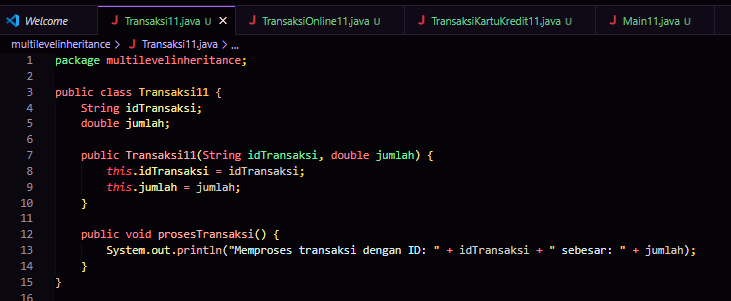
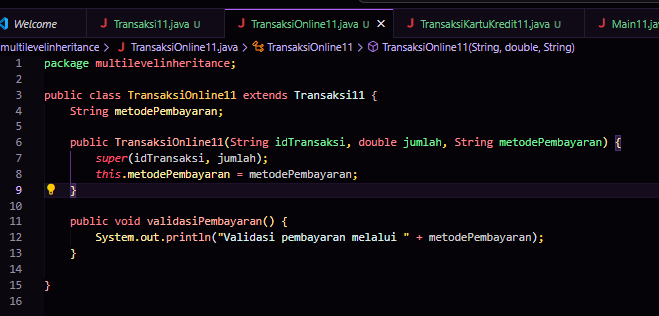
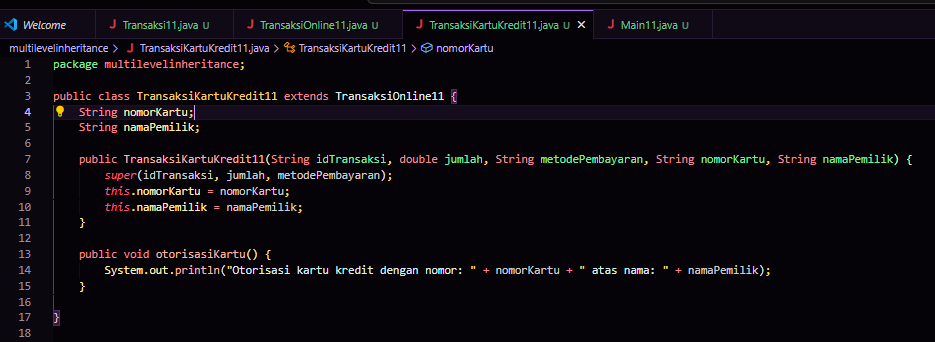
* Attributes:
* cardNumber: String
* ownerName: String
* Methods:
* authorizeCard(): Authorizes a credit card

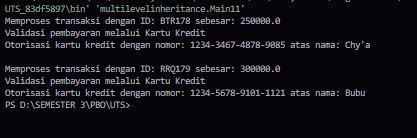
Relations: This class is derived from TransactionOnline, specifically for transactions using credit cards.

1. multilevel inheritance Diagram

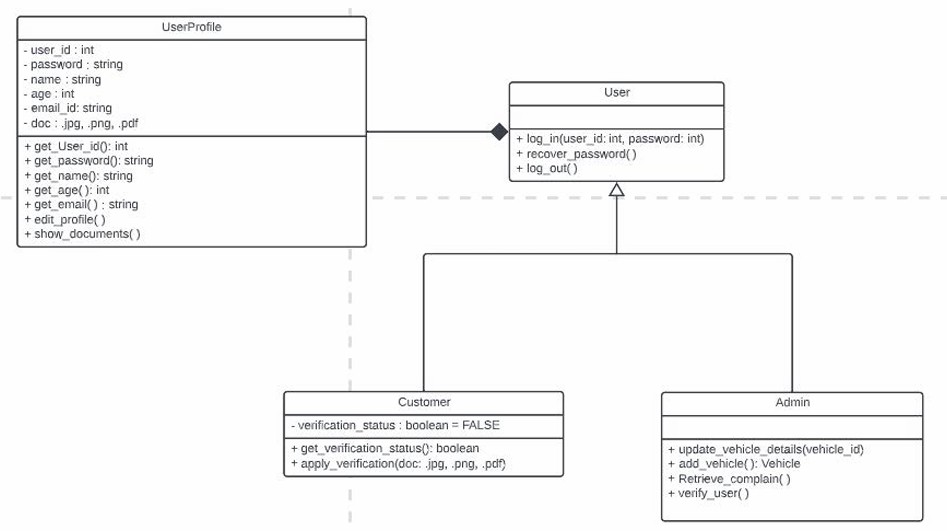


* Class diagram explanation
* The up arrow (^) indicates an inheritance relationship.
* Transaction: a base class that provides a common structure for all types of transactions.
* TransactionOnline: an intermediate class that adds features to the base class.
* TransactionCardCredit : a derived class that has specific features for a type of transaction

1. Program Code
2. Class Transaction (Base Class)
3. TransactionOnline (Intermediate Class)
4. ****CreditCardTransaction (Derived Class )
5. Main
6. Output



1. Please identify the class diagram by providing an explanation of the concept of inheritance, the relationship between classes and the following system flow, create a program code from the following class diagram!



1. concept of inheritance

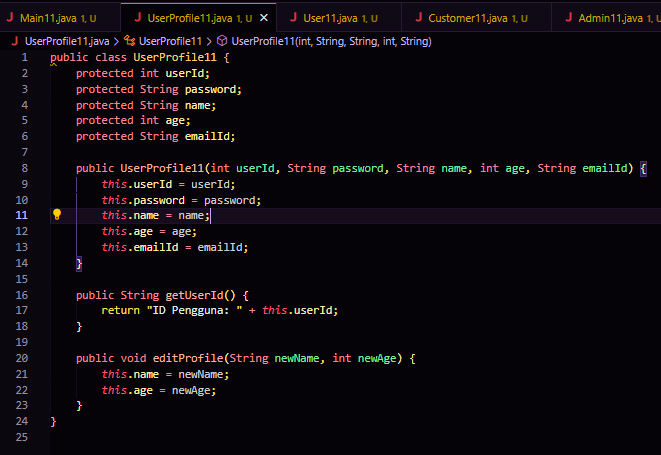
* UserProfile (Parent Class): The main class that stores general information about a user, such as user ID, password, name, age, email, and documents. This class also has methods for getting user ID, password, editing profile, and viewing documents.
* User (Derived Class): This class inherits features from UserProfile and adds specific functions for users, such as logging in, recovering password, and logging out. Its main focus is on the user login process.
* Customer (Derived Class): This class also inherits UserProfile and has attributes and methods related to customers, such as verification status and methods for verifying user documents.
* Admin (Derived Class): This class is designed for admin-specific tasks, such as updating vehicle data, adding new vehicles, and verifying users.

1. Relationship Between Classes

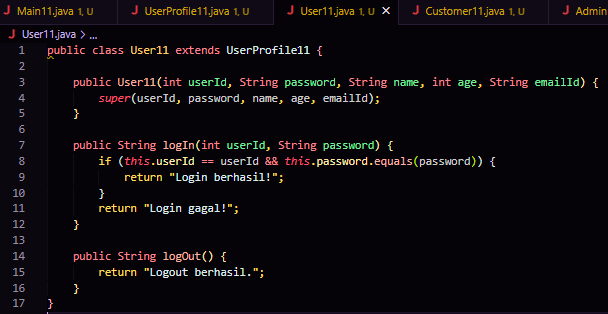
* UserProfile is the base class, and User is an instance that inherits all attributes and methods from UserProfile
* Customer and Admin are subclasses of User, each having additional functionality related to their specific role.

1. Pemrograman

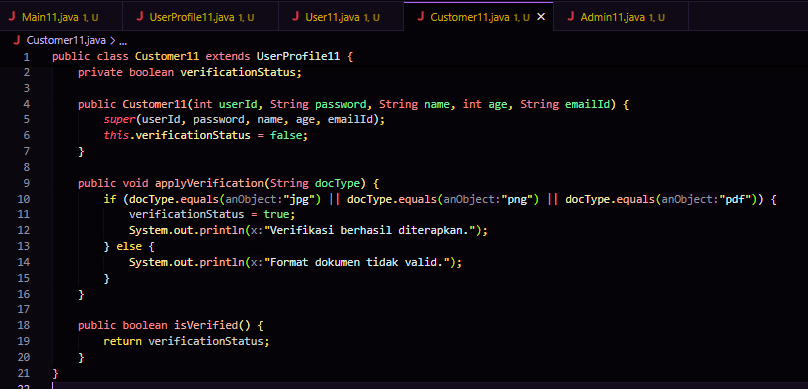
* Kelas UserProfile



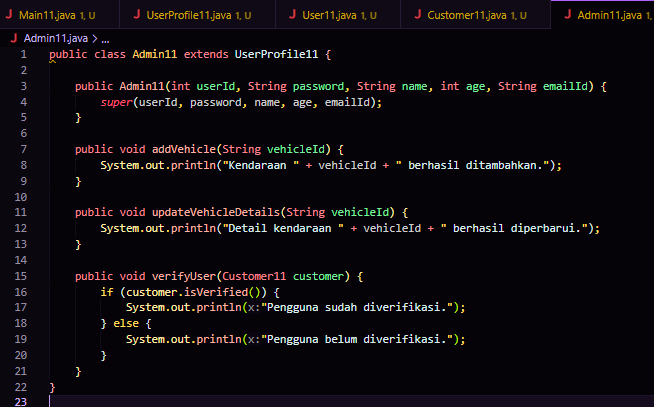
* Kelas User



* Kelas Customer



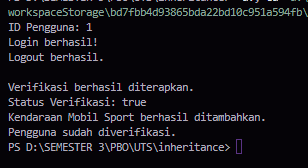
* Kelas Admin



* Kelas Main



* Output



1. System Flow

* User Log In: Users log in to the system using their credentials. Every user (either admin, customer, or general user) uses the login function inherited from UserProfile via User.
* Customer Submit Verification: If the user is a Customer, they can submit verification with valid documents. If the documents are eligible, their verification status changes.
* Admin Verify User: Admin can verify the user (customer) based on the submitted documents, using the verify\_user() function.
* Admin Handle Vehicles: Admin can add or update vehicle information in the system.

1. **Penjelasan Output Program**

* **ID Pengguna: 1**
  1. Ini berasal dari metode getUserId() di objek User11. Metode ini menampilkan ID pengguna, yaitu 1.
* **Login berhasil!**
  1. Ini adalah hasil dari metode logIn(1, "password123"). Metode ini memeriksa apakah userId dan password cocok. Karena sesuai, maka hasilnya adalah "Login berhasil!".
* **Logout berhasil.**
  1. Ini hasil dari metode logOut() di objek User11. Metode ini menunjukkan bahwa pengguna telah berhasil keluar dari sistem.
* **Verifikasi berhasil diterapkan.**
  1. Ini berasal dari metode applyVerification("jpg") di objek Customer11. Metode ini mengecek apakah format dokumen valid (dalam hal ini "jpg"). Karena valid, verifikasi berhasil.
* **Status Verifikasi: true**
  1. Ini berasal dari metode isVerified() di objek Customer11. Metode ini menunjukkan status verifikasi pengguna. Karena verifikasi berhasil, hasilnya adalah true.
* **Kendaraan Mobil Sport berhasil ditambahkan.**
  1. Ini adalah output dari metode addVehicle("Mobil Sport") di objek Admin11. Metode ini menampilkan pesan bahwa kendaraan "Mobil Sport" telah ditambahkan.
* **Pengguna sudah diverifikasi.**
  1. Ini berasal dari metode verifyUser(customer) di objek Admin11. Metode ini memeriksa status verifikasi pelanggan. Karena statusnya true, maka ditampilkan pesan bahwa pengguna sudah terverifikasi.

**---- Good Luck ----**