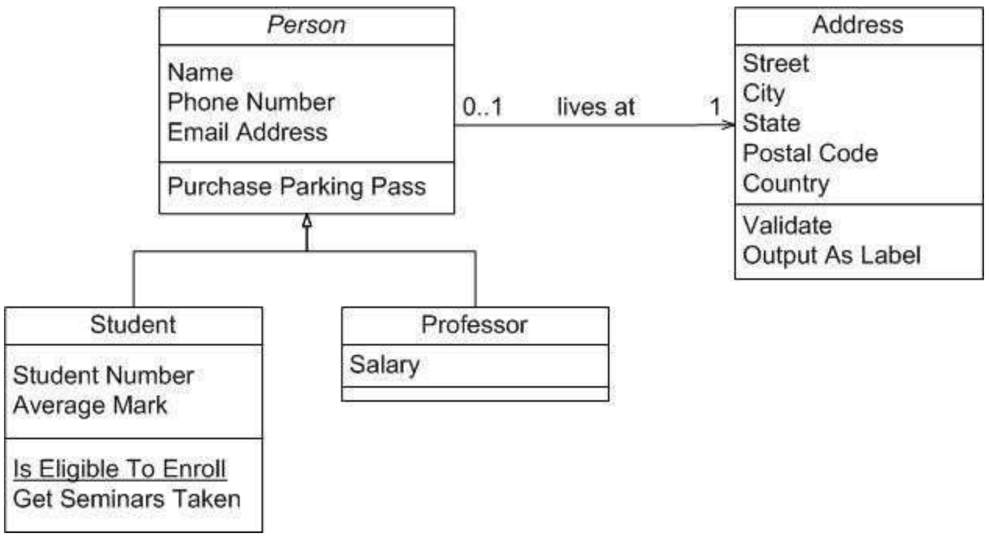
UTS QUESTIONS
OBJECT-BASED PROGRAMMING PRACTICUM

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 :

   a. 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"

b. 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.

c. 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.

d. Professor Class (Derived from Person):

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

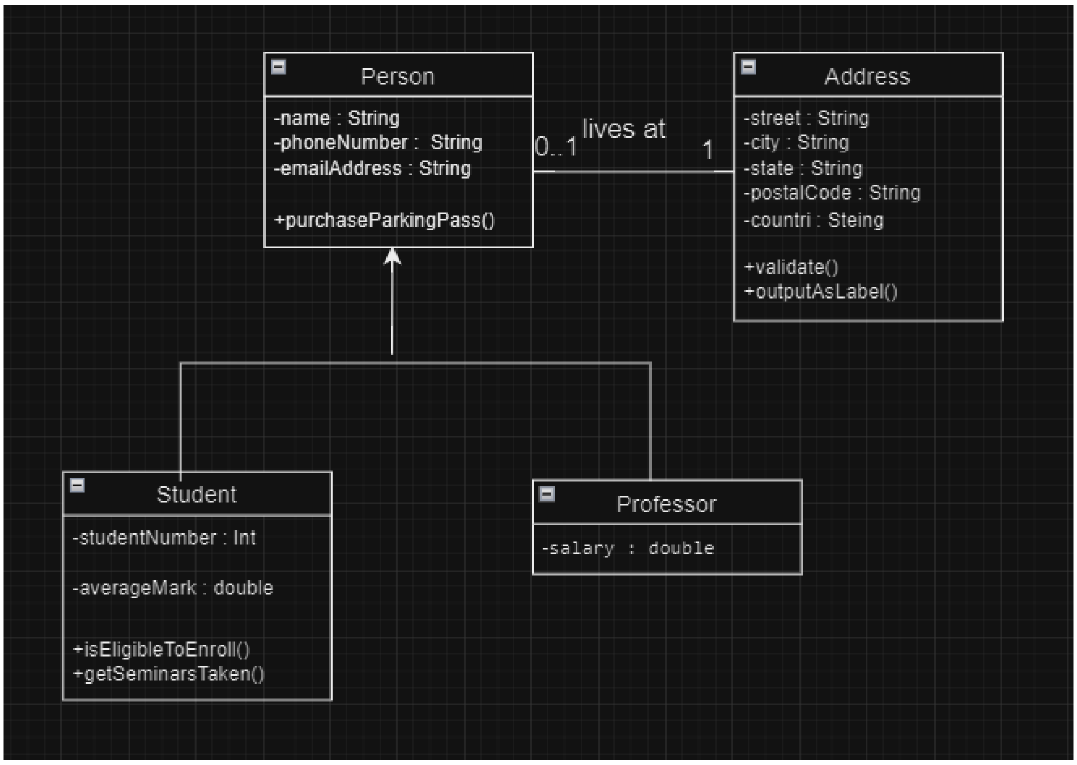
2. Rule Based Repair:

   a. Attributes and methods must be written with a lowercase letter at the beginning.

   b. Modifier: Each attribute and method needs to have access (public/private/protected) added.

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

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



3. Improved Class Diagram:

   a. The Person and Address classes have an aggregation relationship, because a Person can have 0 or 1 addresses.

   b. The inheritance of Student and Professor from Person is implemented with the concept of inheritance, where both inherit attributes and methods from Person.

2. 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

      a. 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.

      b. 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.

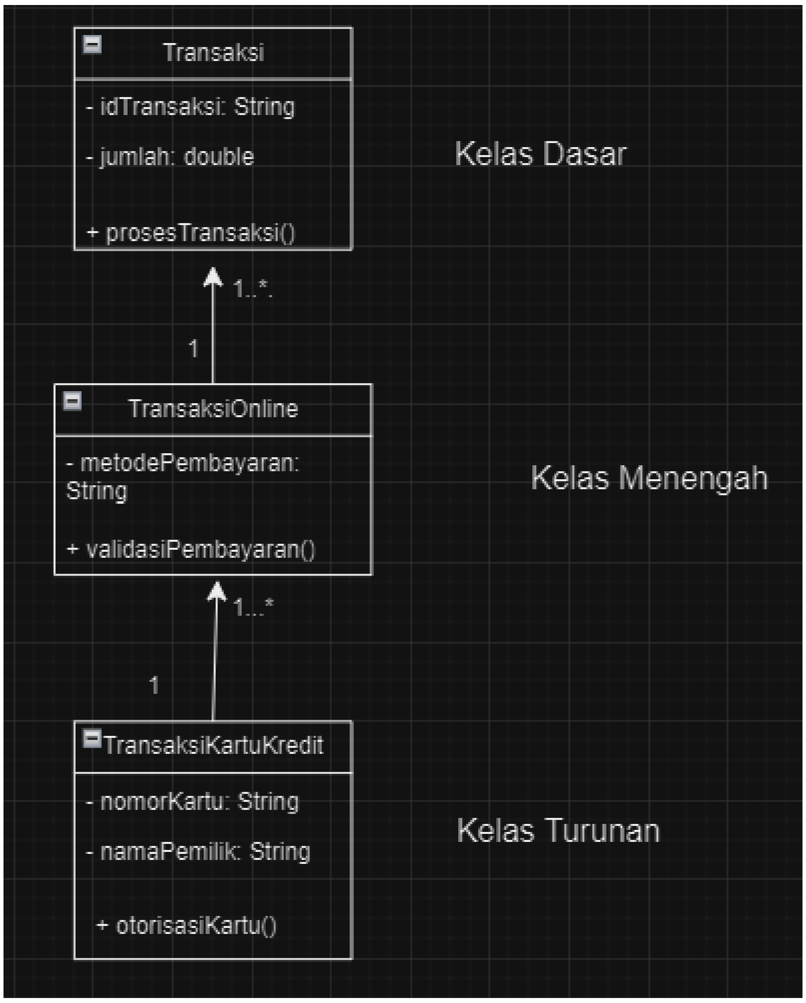
c. CreditCardTransaction (Derived Class)

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.

2. multilevel inheritance Diagram

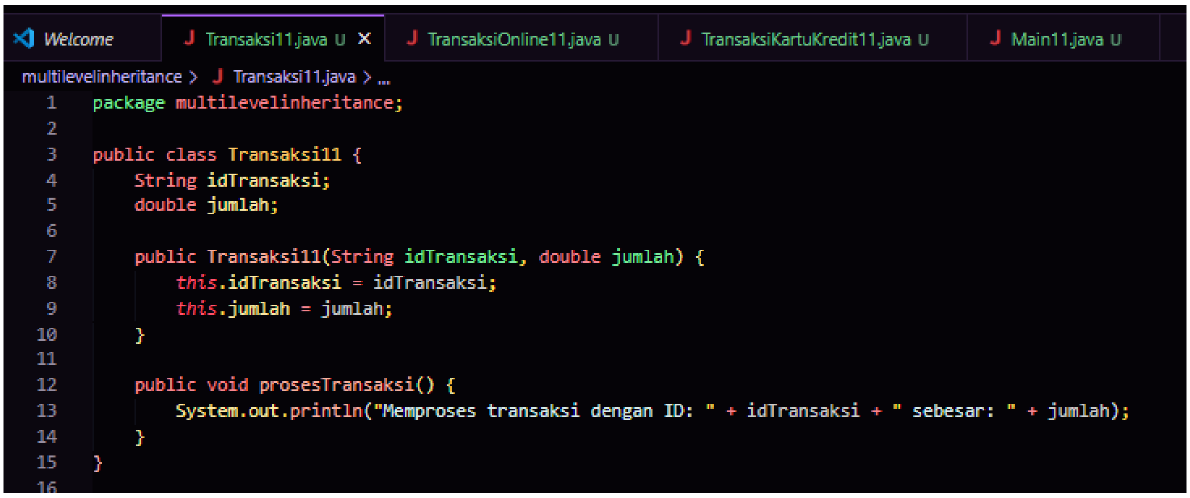


➢ Class diagram explanation

- 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

3. Program Code

a. Class Transaction (Base Class)

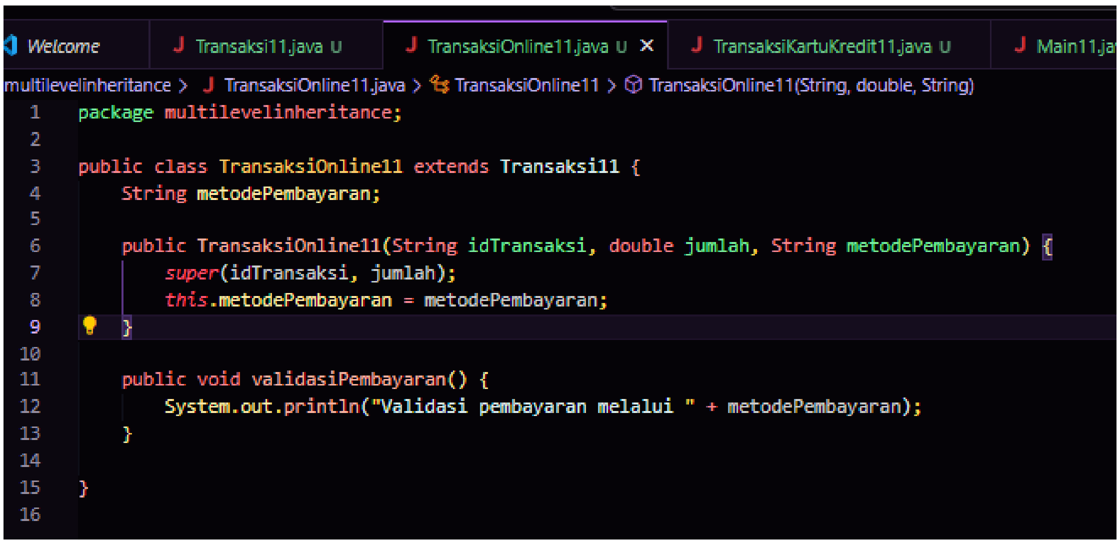


Welcome | Transaksi11.java U × | TransaksiOnline11.java U | TransaksiKartuKredit11.java U | Main11.java U

multilevelinheritance > Transaksi11.java > ...

```java
package multilevelinheritance;

public class Transaksi11 {
    String idTransaksi;
    double jumlah;

    public Transaksi11(String idTransaksi, double jumlah) {
        this.idTransaksi = idTransaksi;
        this.jumlah = jumlah;
    }

    public void prosesTransaksi() {
        System.out.println("Memproses transaksi dengan ID: " + idTransaksi + " sebesar: " + jumlah);
    }
}

```

b. TransactionOnline (Intermediate Class)

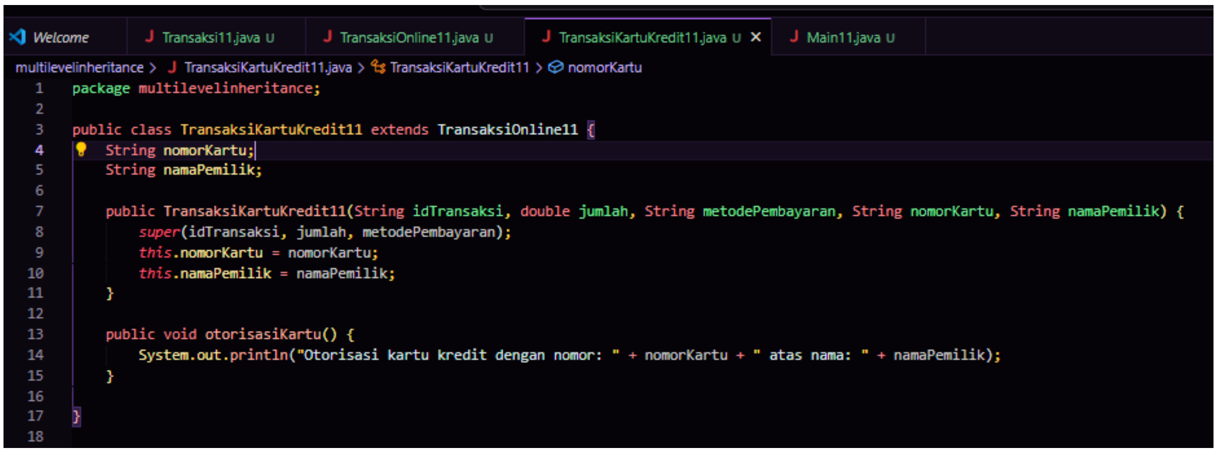


Welcome | Transaksi11.java U | TransaksiOnline11.java U × | TransaksiKartuKredit11.java U | Main11.ja

multilevelinheritance > TransaksiOnline11.java > TransaksiOnline11 > TransaksiOnline11(String, double, String)

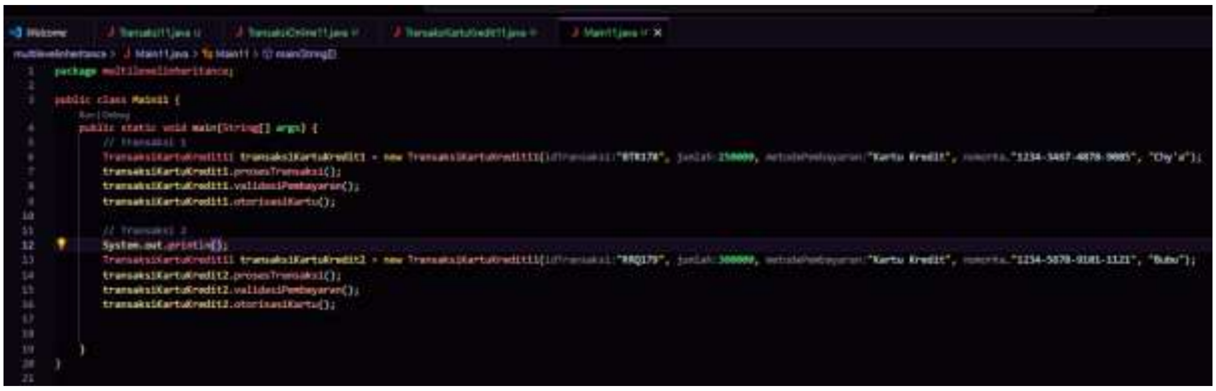
```java
package multilevelinheritance;

public class TransaksiOnline11 extends Transaksi11 {
    String metodePembayaran;

    public TransaksiOnline11(String idTransaksi, double jumlah, String metodePembayaran) {
        super(idTransaksi, jumlah);
        this.metodePembayaran = metodePembayaran;
    }

    public void validasiPembayaran() {
        System.out.println("Validasi pembayaran melalui " + metodePembayaran);
    }

}

```

c. CreditCardTransaction (Derived Class)


Welcome
Transaksi11.java U
TransaksiOnline11.java U
TransaksiKartuKredit11.java U
Main11.java U
multilevelinheritance > TransaksiKartuKredit11.java > TransaksiKartuKredit11 > nomorKartu

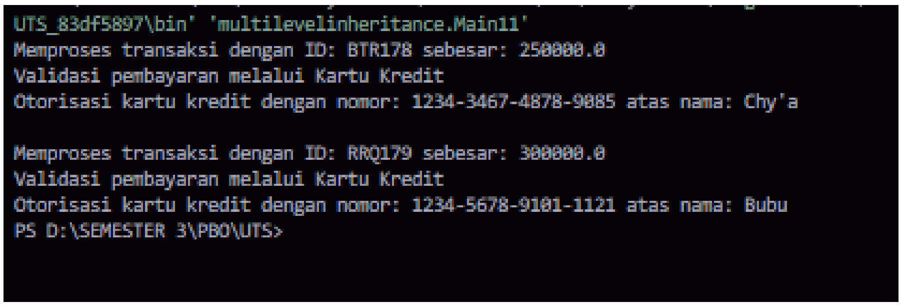
```java
package multilevelinheritance;

public class TransaksiKartuKredit11 extends TransaksiOnline11 {
    String nomorKartu;
    String namaPemilik;

    public TransaksiKartuKredit11(String idTransaksi, double jumlah, String metodePembayaran, String nomorKartu, String namaPemilik) {
        super(idTransaksi, jumlah, metodePembayaran);
        this.nomorKartu = nomorKartu;
        this.namaPemilik = namaPemilik;
    }

    public void otorisasiKartu() {
        System.out.println("Otorisasi kartu kredit dengan nomor: " + nomorKartu + " atas nama: " + namaPemilik);
    }

}
```

d. Main


Welcome
Transaksi11.java U
TransaksiOnline11.java U
TransaksiKartuKredit11.java U
Main11.java U
multilevelinheritance > Main11.java > Main11 > main(String[])
Run | Debug

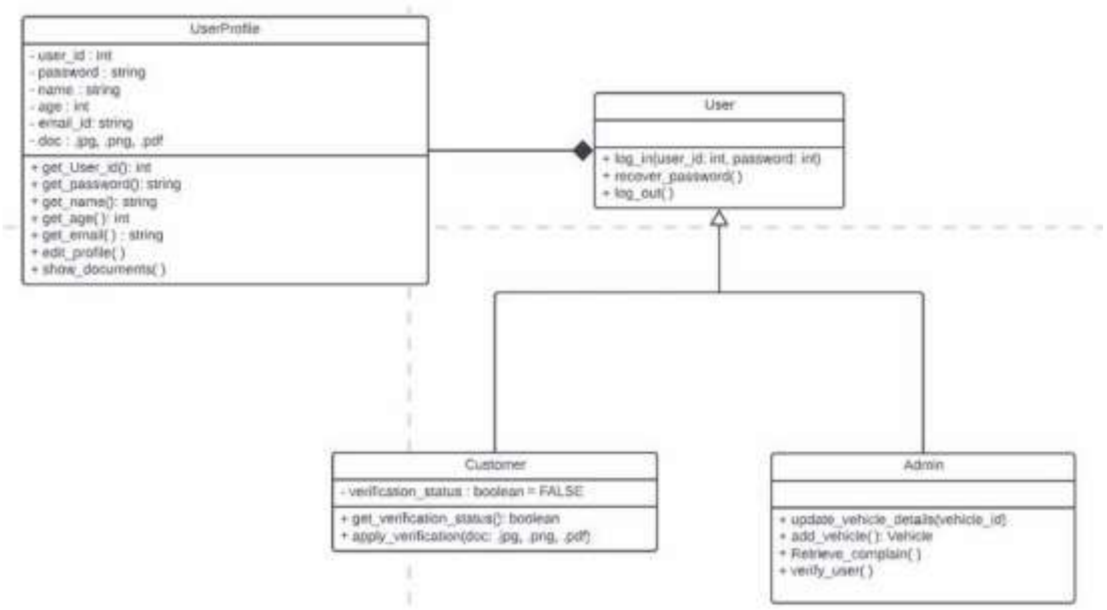
```java
package multilevelinheritance;

public class Main11 {
    public static void main(String[] args) {
        // Transaksi 1
        TransaksiKartuKredit11 transaksiKartuKredit1 = new TransaksiKartuKredit11("BTR178", 250000, "Kartu Kredit", "1234-3467-4878-9885", "Chy'a");
        transaksiKartuKredit1.prosesTransaksi();
        transaksiKartuKredit1.validasiPembayaran();
        transaksiKartuKredit1.otorisasiKartu();

        // Transaksi 2
        System.out.println();
        TransaksiKartuKredit11 transaksiKartuKredit2 = new TransaksiKartuKredit11("RRQ179", 300000, "Kartu Kredit", "1234-5678-9101-1121", "Bubu");
        transaksiKartuKredit2.prosesTransaksi();
        transaksiKartuKredit2.validasiPembayaran();
        transaksiKartuKredit2.otorisasiKartu();


    }
}
```

e. Output



```
UTS_83df5897\bin' 'multilevelinheritance.Main11'
Memproses transaksi dengan ID: BTR178 sebesar: 250000.0
Validasi pembayaran melalui Kartu Kredit
Otorisasi kartu kredit dengan nomor: 1234-3467-4878-9885 atas nama: Chy'a

Memproses transaksi dengan ID: RRQ179 sebesar: 300000.0
Validasi pembayaran melalui Kartu Kredit
Otorisasi kartu kredit dengan nomor: 1234-5678-9101-1121 atas nama: Bubu
PS D:\SEMESTER 3\PBO\UTS>
```

a. 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!



a. 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.

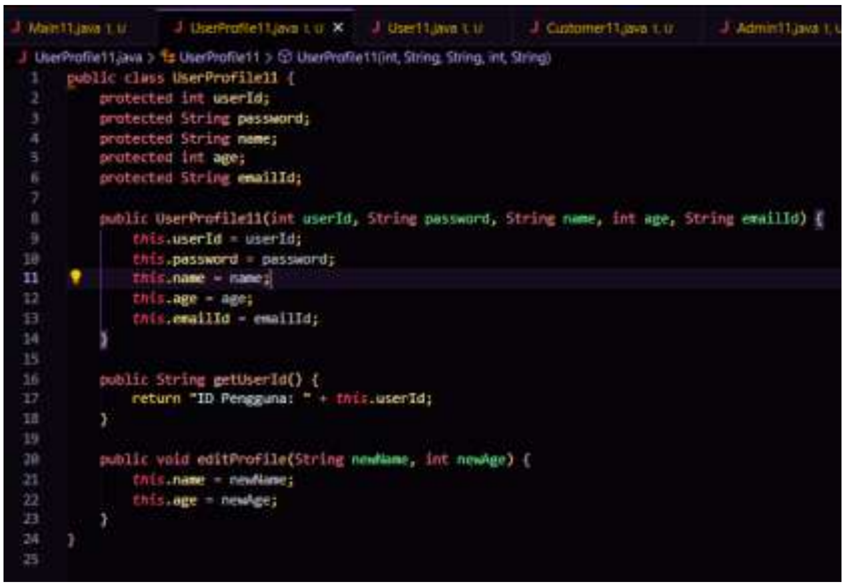
- Customer (Derived Class): This class also inherits UserProfile and has attributes and methods related to customers, such as verification status and methods for verifying user documents.

- Admin (Derived Class): This class is designed for admin-specific tasks, such as updating vehicle data, adding new vehicles, and verifying users.

b. 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.

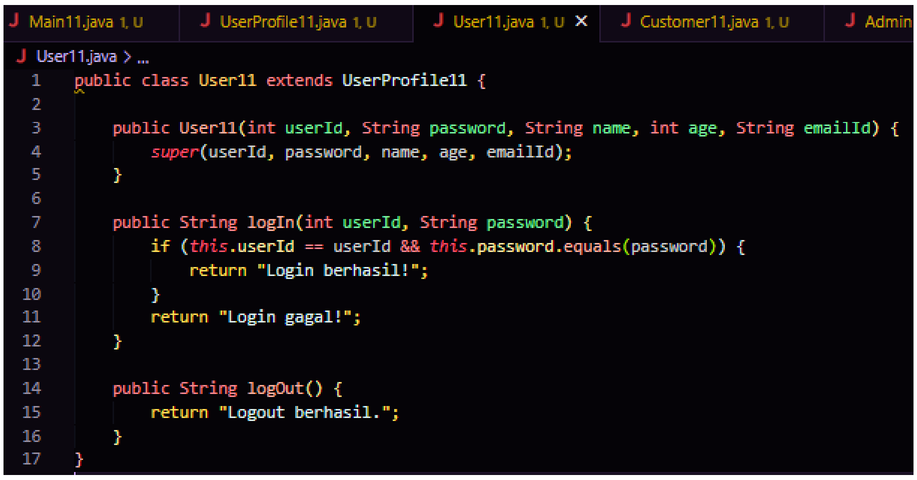
c. Pemrograman

   - Kelas UserProfile



```java
public class UserProfile11 {
    protected int userId;
    protected String password;
    protected String name;
    protected int age;
    protected String emailId;

    public UserProfile11(int userId, String password, String name, int age, String emailId) {
        this.userId = userId;
        this.password = password;
        this.name = name;
        this.age = age;
        this.emailId = emailId;
    }

    public String getUserId() {
        return "ID Pengguna: " + this.userId;
    }

    public void editProfile(String newName, int newAge) {
        this.name = newName;
        this.age = newAge;
    }
}
```

- Kelas `User`

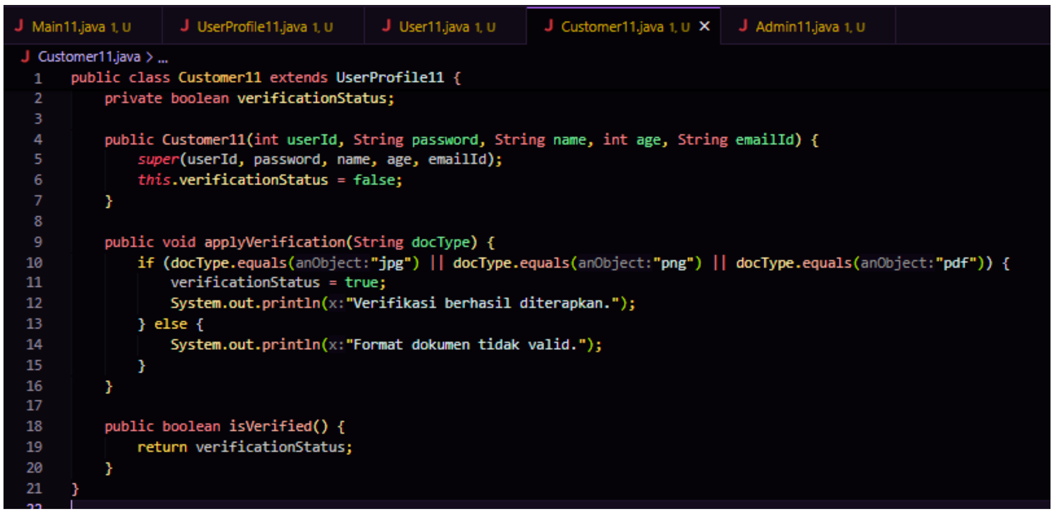


Main11.java 1, U · UserProfile11.java 1, U · User11.java 1, U × · Customer11.java 1, U · Admin

User11.java > ...

```java
public class User11 extends UserProfile11 {

    public User11(int userId, String password, String name, int age, String emailId) {
        super(userId, password, name, age, emailId);
    }

    public String logIn(int userId, String password) {
        if (this.userId == userId && this.password.equals(password)) {
            return "Login berhasil!";
        }
        return "Login gagal!";
    }

    public String logOut() {
        return "Logout berhasil.";
    }
}
```

- Kelas `Customer`

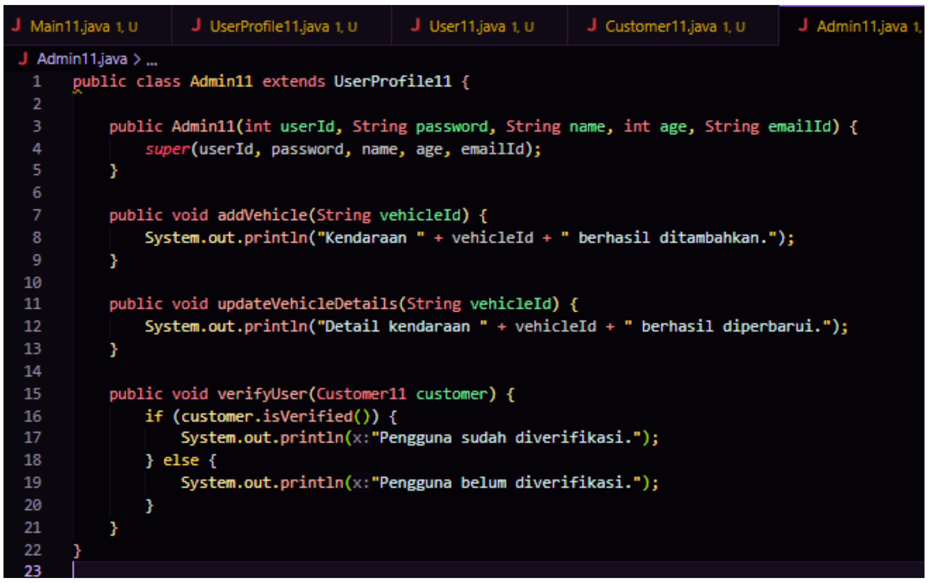


Main11.java 1, U · UserProfile11.java 1, U · User11.java 1, U · Customer11.java 1, U × · Admin11.java 1, U

Customer11.java > ...

```java
public class Customer11 extends UserProfile11 {
    private boolean verificationStatus;

    public Customer11(int userId, String password, String name, int age, String emailId) {
        super(userId, password, name, age, emailId);
        this.verificationStatus = false;
    }

    public void applyVerification(String docType) {
        if (docType.equals(anObject:"jpg") || docType.equals(anObject:"png") || docType.equals(anObject:"pdf")) {
            verificationStatus = true;
            System.out.println(x:"Verifikasi berhasil diterapkan.");
        } else {
            System.out.println(x:"Format dokumen tidak valid.");
        }
    }

    public boolean isVerified() {
        return verificationStatus;
    }
}
```

- Kelas Admin

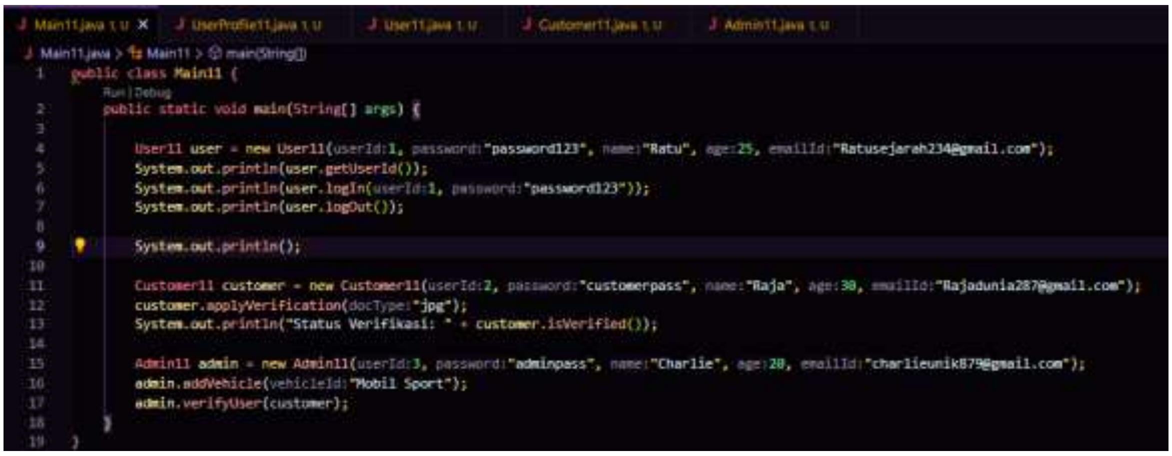


Main11.java 1, U · UserProfile11.java 1, U · User11.java 1, U · Customer11.java 1, U · Admin11.java 1,

Admin11.java > ...

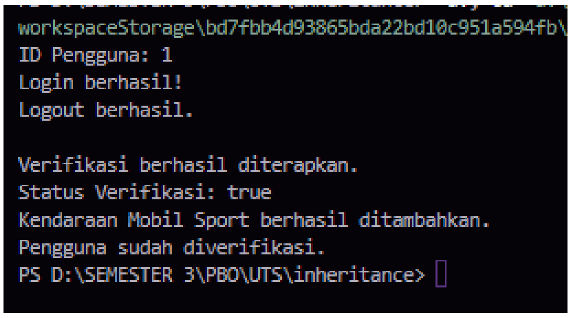
```java
public class Admin11 extends UserProfile11 {

    public Admin11(int userId, String password, String name, int age, String emailId) {
        super(userId, password, name, age, emailId);
    }

    public void addVehicle(String vehicleId) {
        System.out.println("Kendaraan " + vehicleId + " berhasil ditambahkan.");
    }

    public void updateVehicleDetails(String vehicleId) {
        System.out.println("Detail kendaraan " + vehicleId + " berhasil diperbarui.");
    }

    public void verifyUser(Customer11 customer) {
        if (customer.isVerified()) {
            System.out.println(x:"Pengguna sudah diverifikasi.");
        } else {
            System.out.println(x:"Pengguna belum diverifikasi.");
        }
    }
}
```

- Kelas Main



```java
public class Main11 {
    public static void main(String[] args) {

        User11 user = new User11(userId:1, password:"password123", name:"Ratu", age:25, emailId:"Ratusejarah234@gmail.com");
        System.out.println(user.getUserId());
        System.out.println(user.logIn(userId:1, password:"password123"));
        System.out.println(user.logOut());

        System.out.println();

        Customer11 customer = new Customer11(userId:2, password:"customerpass", name:"Raja", age:30, emailId:"Rajadunia287@gmail.com");
        customer.applyVerification(docType:"jpg");
        System.out.println("Status Verifikasi: " + customer.isVerified());

        Admin11 admin = new Admin11(userId:3, password:"adminpass", name:"Charlie", age:20, emailId:"charlieunik879@gmail.com");
        admin.addVehicle(vehicleId:"Mobil Sport");
        admin.verifyUser(customer);
    }
}
```

- Output



```
workspaceStorage\bd7fbb4d93865bda22bd10c951a594fb\
ID Pengguna: 1
Login berhasil!
Logout berhasil.

Verifikasi berhasil diterapkan.
Status Verifikasi: true
Kendaraan Mobil Sport berhasil ditambahkan.
Pengguna sudah diverifikasi.
PS D:\SEMESTER 3\PBO\UTS\inheritance>
```

d. 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.

---- Good Luck ----