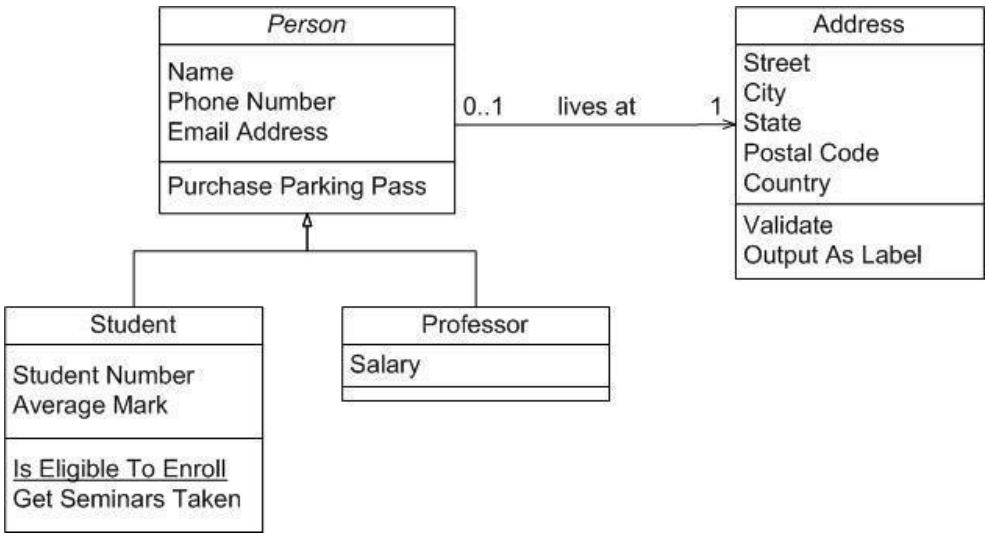
**Name : Hanin Mariam Abiyyah Hendrik**
**Class : 2G - SIB**
**NIM : 2341760154**

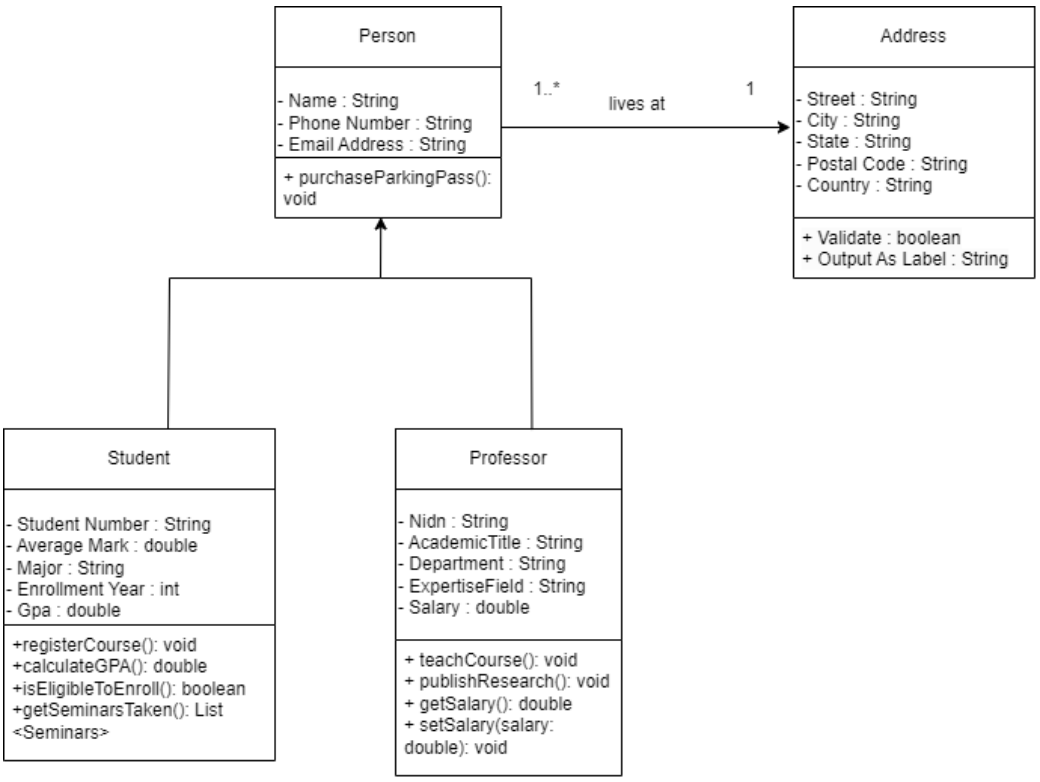
https://github.com/huniens/PBO/tree/main/MIDTERM%20TEST

## 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.

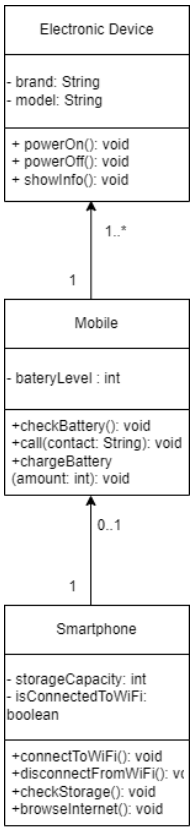



Answer:



I changed it by marking which attribute (-) and which method (+). I also added some attributes and methods. This diagram shows a system that connects student and faculty data on campus. It stores everyone's personal, academic, and address information. In addition, the system has features for activities such as registering courses, calculating GPA, and managing lecturer salaries.

2. Create a diagram class that uses multilevel inheritance and create the program code!




**Class relationship**

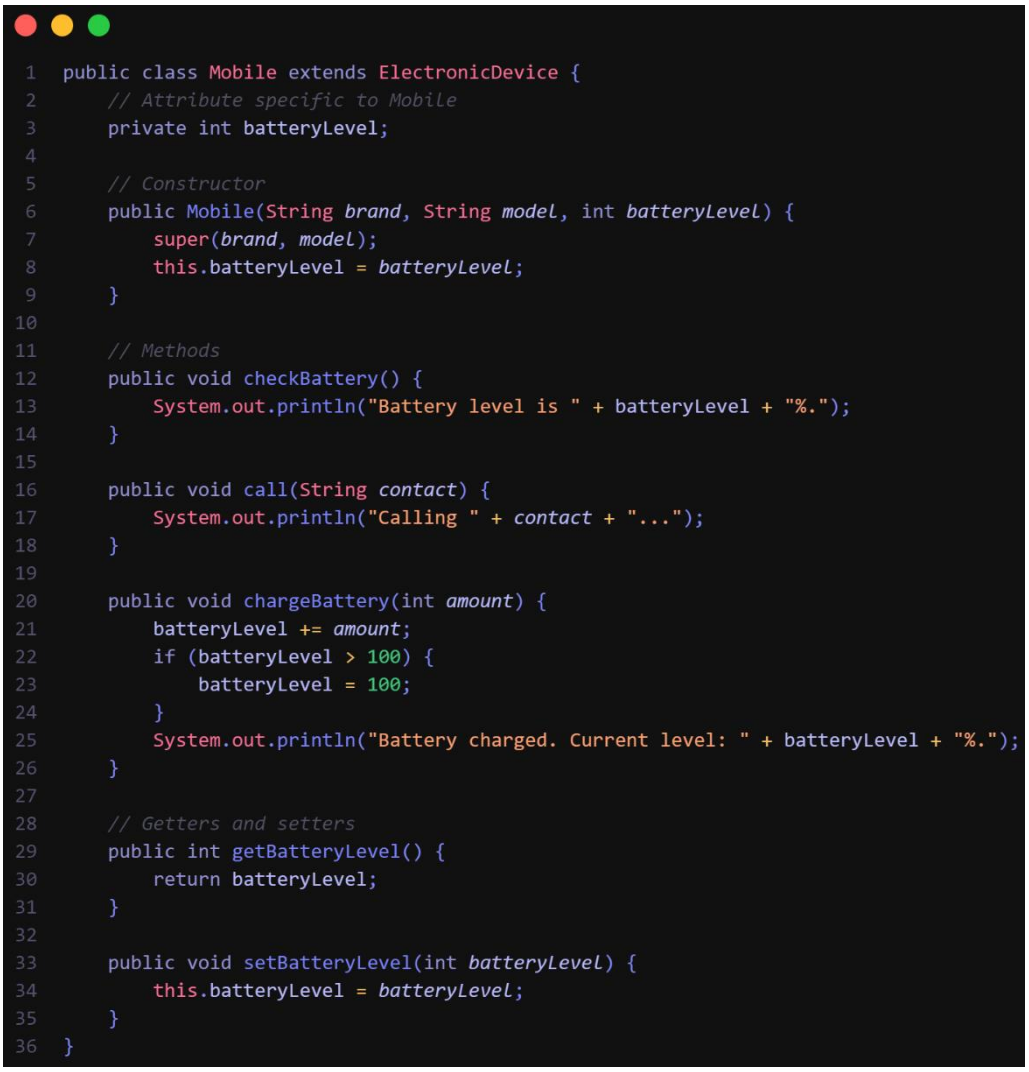
1. Electronic Device (1..*) --- Mobile (1)
   So, each Electronic Device (such as a laptop or tablet) can be connected to one or more Mobile (cellphone), but each Mobile is only connected to one Electronic Device. So, one device can have many phones, but one phone is only connected to one device.

2. Mobile (0..1) --- Smartphone (1)
   Now, in the relationship between Mobile and Smartphone, one Mobile may not have a Smartphone (0..1), or if it does, it is only one at most. So, one mobile can have or not have a smartphone, but one smartphone is only for one mobile.

**Program code**

```java
public class ElectronicDevice {
    // Attributes
    private String brand;
    private String model;

    // Constructor
    public ElectronicDevice(String brand, String model) {
        this.brand = brand;
        this.model = model;
    }

    // Methods
    public void powerOn() {
        System.out.println(brand + " " + model + " is now ON.");
    }

    public void powerOff() {
        System.out.println(brand + " " + model + " is now OFF.");
    }

    public void showInfo() {
        System.out.println("Brand: " + brand + ", Model: " + model);
    }

    // Getters and setters
    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }
}
```

ElectronicDevice is a base class with attributes for brand and model,and basic methods like powerOn(), powerOff(), and showInfo(). In essence, it serves as the blueprint for all electronic devices in its subclasses.



```java
public class Mobile extends ElectronicDevice {
    // Attribute specific to Mobile
    private int batteryLevel;

    // Constructor
    public Mobile(String brand, String model, int batteryLevel) {
        super(brand, model);
        this.batteryLevel = batteryLevel;
    }

    // Methods
    public void checkBattery() {
        System.out.println("Battery level is " + batteryLevel + "%.");
    }

    public void call(String contact) {
        System.out.println("Calling " + contact + "...");
    }

    public void chargeBattery(int amount) {
        batteryLevel += amount;
        if (batteryLevel > 100) {
            batteryLevel = 100;
        }
        System.out.println("Battery charged. Current level: " + batteryLevel + "%.");
    }

    // Getters and setters
    public int getBatteryLevel() {
        return batteryLevel;
    }

    public void setBatteryLevel(int batteryLevel) {
        this.batteryLevel = batteryLevel;
    }
}
```

Mobile inherits all attributes and methods from ElectronicDevice and adds new attributes like batteryLevel, as well as specific methods for Mobile, such as checkBattery(), call(), and chargeBattery().

```java
public class Smartphone extends Mobile {
    // Attributes specific to Smartphone
    private int storageCapacity;
    private boolean isConnectedToWiFi;

    // Constructor
    public Smartphone(String brand, String model, int batteryLevel, int storageCapacity) {
        super(brand, model, batteryLevel);
        this.storageCapacity = storageCapacity;
        this.isConnectedToWiFi = false; // initially not connected to WiFi
    }

    // Methods
    public void connectToWiFi() {
        isConnectedToWiFi = true;
        System.out.println("Connected to WiFi.");
    }

    public void disconnectFromWiFi() {
        isConnectedToWiFi = false;
        System.out.println("Disconnected from WiFi.");
    }

    public void checkStorage() {
        System.out.println("Storage capacity: " + storageCapacity + "GB.");
    }

    public void browseInternet() {
        if (isConnectedToWiFi) {
            System.out.println("Browsing the internet...");
        } else {
            System.out.println("Please connect to WiFi to browse the internet.");
        }
    }

    // Getters and setters
    public int getStorageCapacity() {
        return storageCapacity;
    }

    public void setStorageCapacity(int storageCapacity) {
        this.storageCapacity = storageCapacity;
    }

    public boolean isConnectedToWiFi() {
        return isConnectedToWiFi;
    }

    public void setConnectedToWiFi(boolean connectedToWiFi) {
        isConnectedToWiFi = connectedToWiFi;
    }
}
```

Smartphone, inheriting from Mobile, also gets all attributes and methods from Mobile (including those inherited from ElectronicDevice). Smartphone adds attributes like storageCapacity and isConnectedToWiFi, and methods like connectToWiFi(), disconnectFromWiFi(), checkStorage(), and browseInternet().

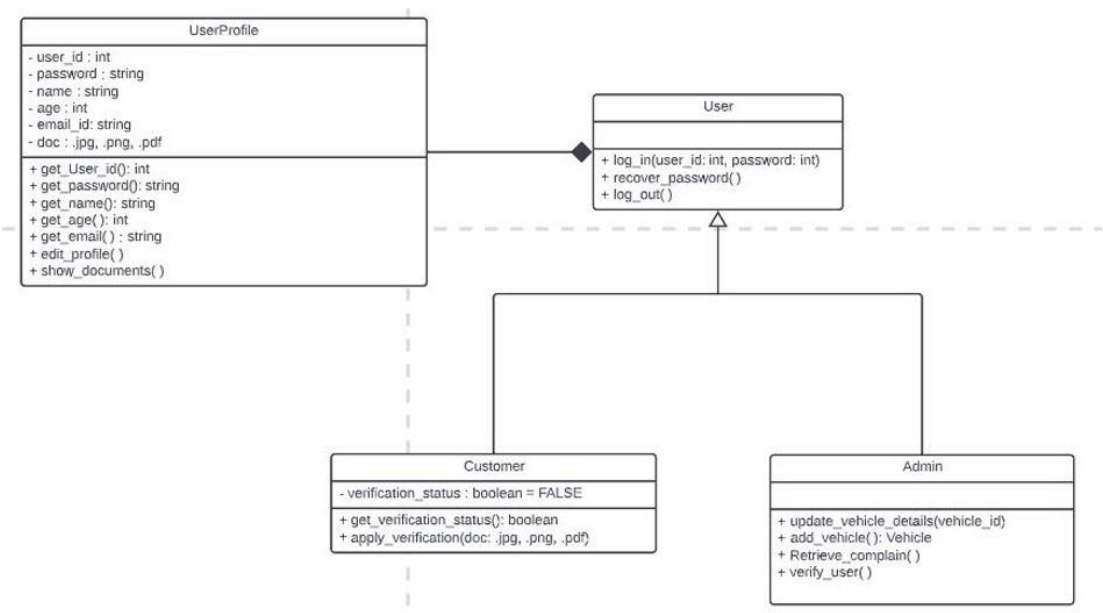
```java
public class Main {
    public static void main(String[] args) {
        // Create a Smartphone object
        Smartphone myPhone = new Smartphone("Samsung", "Galaxy S21", 75, 128);

        // Using methods from ElectronicDevice, Mobile, and Smartphone
        myPhone.powerOn();
        myPhone.showInfo();
        myPhone.checkBattery();
        myPhone.call("John Doe");
        myPhone.chargeBattery(20);
        myPhone.checkBattery();
        myPhone.connectToWiFi();
        myPhone.browseInternet();
        myPhone.checkStorage();
        myPhone.powerOff();
    }
}
```

This code shows how to use a Smartphone object and methods from the ElectronicDevice, Mobile, and Smartphone classes. It basically shows how one object can use features from all these classes.

**Output**

```
Samsung Galaxy S21 is now ON.
Brand: Samsung, Model: Galaxy S21
Battery level is 75%.
Calling John Doe...
Battery charged. Current level: 95%.
Battery level is 95%.
Connected to WiFi.
Browsing the internet...
Storage capacity: 128GB.
Samsung Galaxy S21 is now OFF.
```

3. 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!




**Concept of Inheritance**

Inheritance is a concept where one class (child class) inherits attributes and methods from another class (parent class). This allows the child class to use the properties and methods defined in the parent class and add new properties or methods.

In this diagram, the User class is the parent class of two child classes: Customer and Admin. This means Customer and Admin inherit attributes and methods from the User class. Additionally, User has an aggregation relationship with UserProfile.

**Relationships between the classes in this diagram can be explained like this:**

1. UserProfile is the class that stores user profile data like user_id, password, name, etc. It also has methods to access user profile information.
2. User has methods for login, logout, and managing passwords. User has an aggregation relationship with UserProfile, meaning each User object has a UserProfile.
3. Customer and Admin are subclasses of User. They inherit methods like log_in, log_out, and recover_password from User and have their own specific methods.
    - Customer has methods for user verification.
    - Admin has methods for managing vehicles and user complaints.

**System flow:**

1. User logs into the system using the log_in method.
2. If the user is a Customer, they can verify their identity using the apply_verification method.
3. If the user is an Admin, they can manage vehicle data, add vehicles, handle complaints, and verify users.
4. All users can log out of the system using the log_out method.

**Program code**

```java
public class UserProfile {
    private int userId;
    private String password;
    private String name;
    private String email;
    private String doc; // can be .jpg, .png, .pdf

    public UserProfile(int userId, String password, String name, String email) {
        this.userId = userId;
        this.password = password;
        this.name = name;
        this.email = email;
    }

    public int getUserId() {
        return userId;
    }

    public String getPassword() {
        return password;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }

    public void editProfile(String name, String email) {
        this.name = name;
        this.email = email;
    }

    public void showDocuments() {
        System.out.println("Showing documents: " + doc);
    }
}
```

```java
public class User {
    protected UserProfile profile;

    public User(UserProfile profile) {
        this.profile = profile;
    }

    public void logIn(int userId, String password) {
        if (this.profile.getUserId() == userId && this.profile.getPassword().equals(password)) {
            System.out.println("Login successful!");
        } else {
            System.out.println("Invalid credentials.");
        }
    }

    public void recoverPassword() {
        System.out.println("Recovering password...");
    }

    public void logOut() {
        System.out.println("User logged out.");
    }
}
```

```java
public class Customer extends User {
    private boolean verificationStatus = false;

    public Customer(UserProfile profile) {
        super(profile);
    }

    public boolean getVerificationStatus() {
        return verificationStatus;
    }

    public void applyVerification(String doc) {
        if (doc.equals(".jpg") || doc.equals(".png") || doc.equals(".pdf")) {
            verificationStatus = true;
            System.out.println("Verification applied successfully.");
        } else {
            System.out.println("Invalid document format.");
        }
    }
}
```

```java
public class Admin extends User {

    public Admin(UserProfile profile) {
        super(profile);
    }

    public void updateVehicleDetails(int vehicleId) {
        System.out.println("Updating vehicle details for vehicle ID: " + vehicleId);
    }

    public void addVehicle(String vehicle) {
        System.out.println("Vehicle added: " + vehicle);
    }

    public void retrieveComplaint() {
        System.out.println("Retrieving complaints...");
    }

    public void verifyUser() {
        System.out.println("Verifying user...");
    }
}
```

```java
public class MainProgram {
    public static void main(String[] args) {
        // Create UserProfile object for Hanin Mariam
        UserProfile profile = new UserProfile(101, "password123", "Hanin Mariam", "hanin.mariam@example.com");

        // Create Customer object and demonstrate functionality
        Customer customer = new Customer(profile);
        customer.logIn(101, "password123");
        customer.applyVerification(".jpg");
        System.out.println("Customer verification status: " + customer.getVerificationStatus());
        customer.logOut();

        // Create Admin object and demonstrate functionality
        Admin admin = new Admin(profile);
        admin.logIn(101, "password123");
        admin.addVehicle("Chevrolet Chevelle SS");
        admin.updateVehicleDetails(1234);
        admin.retrieveComplaint();
        admin.verifyUser();
        admin.logOut();
    }
}
```

## Output

```
Login successful!
Verification applied successfully.
Customer verification status: true
User logged out.
Login successful!
Vehicle added: Chevrolet Chevelle SS
Updating vehicle details for vehicle ID: 1234
Retrieving complaints...
Verifying user...
User logged out.
```

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