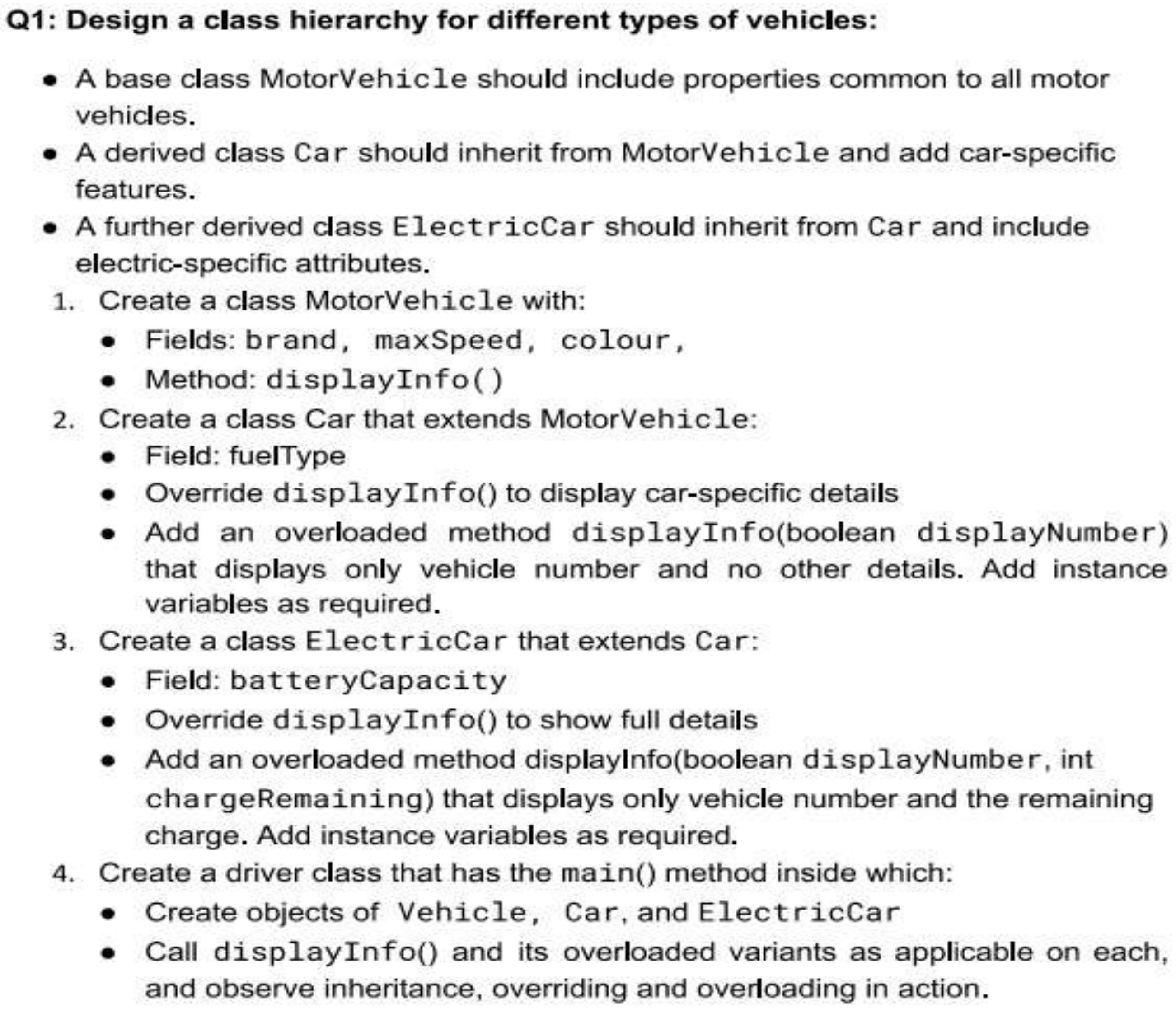
ICS1312 – JAVA PROGRAMMING LABORATORY

DATE	**: 30.7.2025**
ASSIGNMENT	**: 2**
TITLE	**: Inheritence and polymorphism**
ROLLL NO	**: 3122247001017**

LEARNING OBJECTIVE:

- **To work with interface in java**
- **To create abstract class in java**
- **To use super keyword to call parent class**

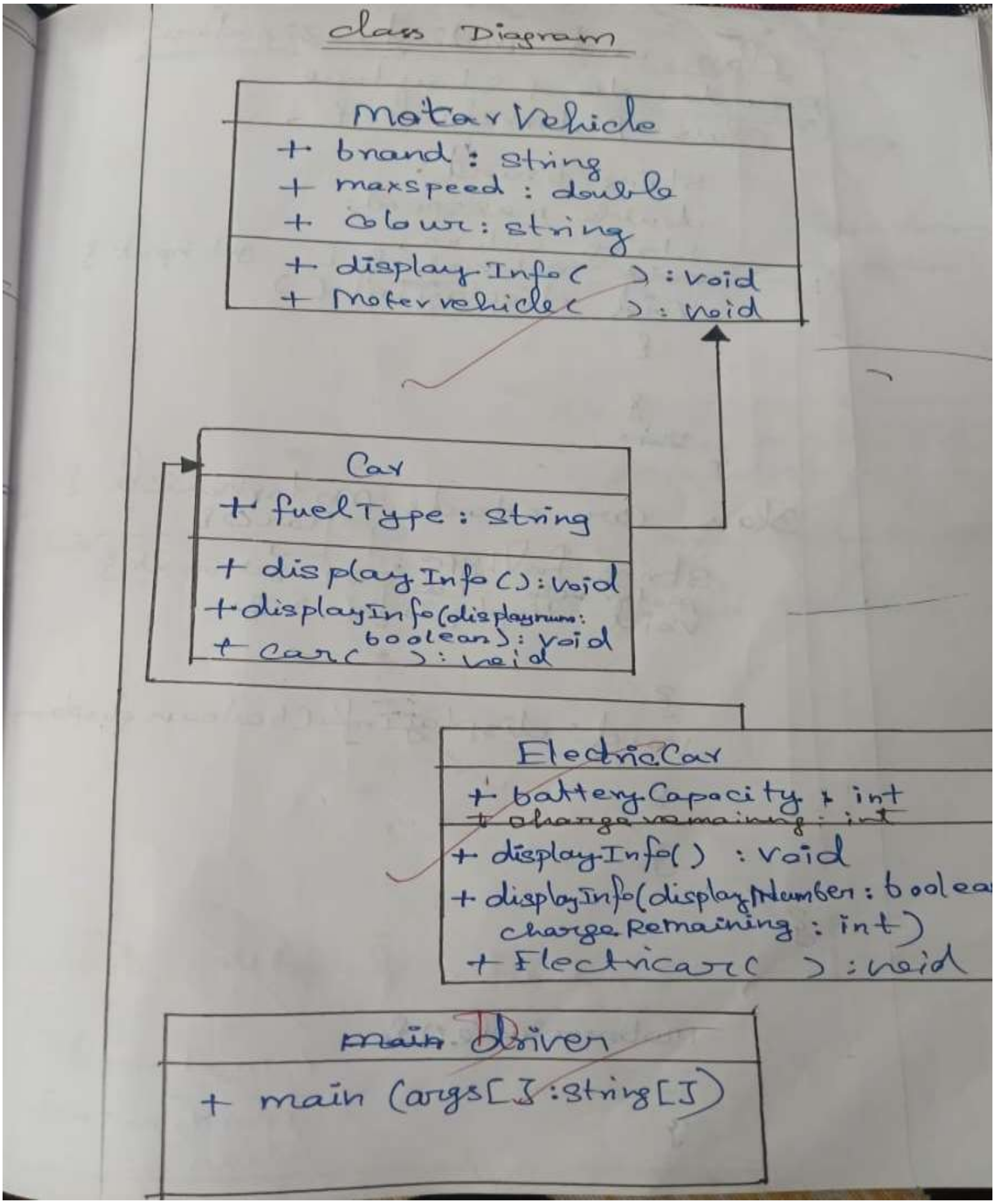


Q1: Design a class hierarchy for different types of vehicles:

- A base class MotorVehicle should include properties common to all motor vehicles.
- A derived class Car should inherit from MotorVehicle and add car-specific features.
- A further derived class ElectricCar should inherit from Car and include electric-specific attributes.

1. Create a class MotorVehicle with:
   - Fields: `brand, maxSpeed, colour,`
   - Method: `displayInfo()`
2. Create a class Car that extends MotorVehicle:
   - Field: fuelType
   - Override `displayInfo()` to display car-specific details
   - Add an overloaded method `displayInfo(boolean displayNumber)` that displays only vehicle number and no other details. Add instance variables as required.
3. Create a class `ElectricCar` that extends `Car`:
   - Field: `batteryCapacity`
   - Override `displayInfo()` to show full details
   - Add an overloaded method displayInfo(boolean `displayNumber`, int `chargeRemaining`) that displays only vehicle number and the remaining charge. Add instance variables as required.
4. Create a driver class that has the `main()` method inside which:
   - Create objects of `Vehicle, Car`, and `ElectricCar`
   - Call `displayInfo()` and its overloaded variants as applicable on each, and observe inheritance, overriding and overloading in action.

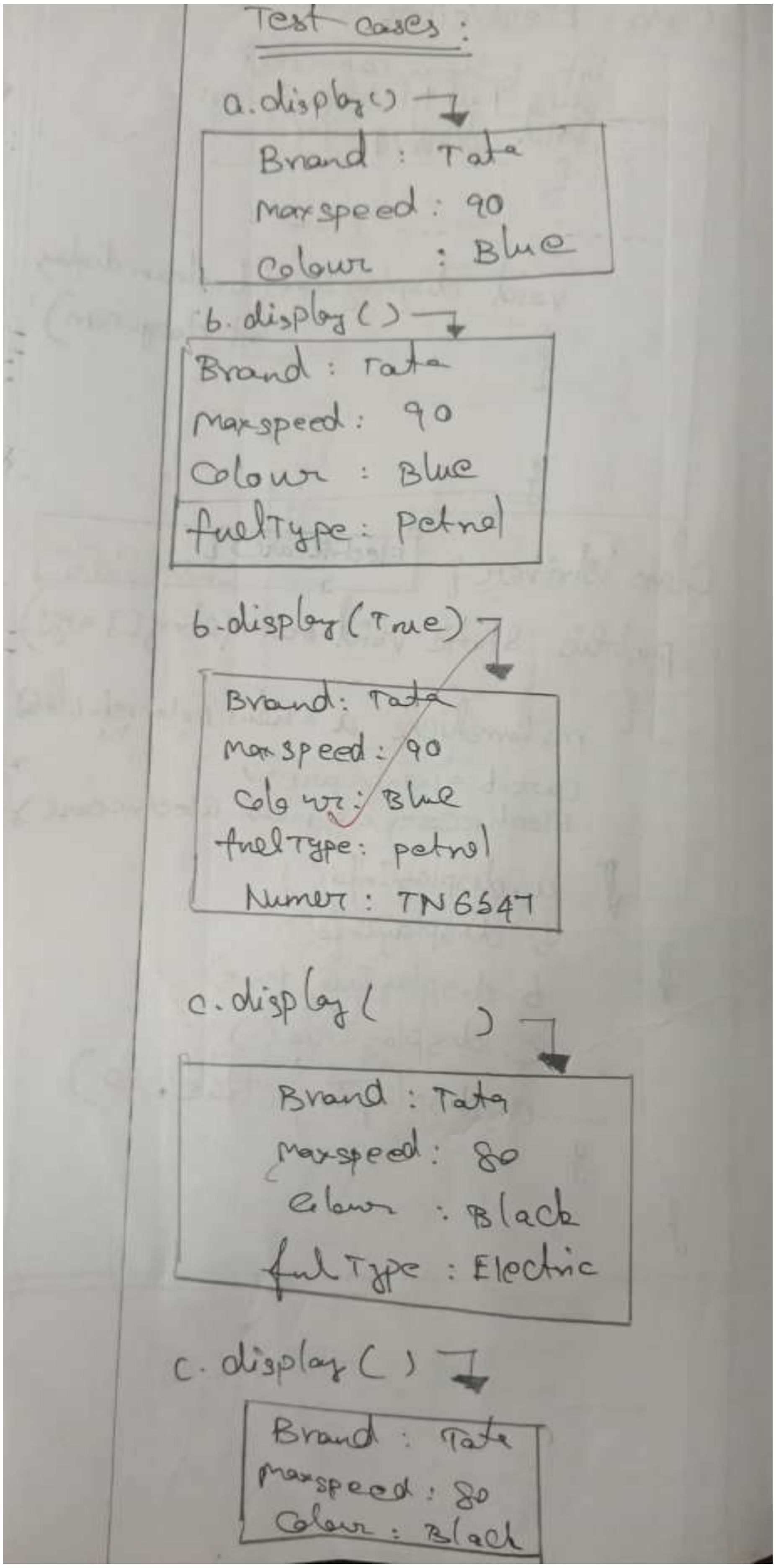
CLASS DIAGRAM :



CODE:

```java
import java.util.Scanner;

class MotorVehicle {
    String brand;
    double maxSpeed;
    String colour;
    String vehicleNumber;

    public MotorVehicle(String brand, double maxSpeed, String colour, String
vehicleNumber) {
        this.brand = brand;
        this.maxSpeed = maxSpeed;
        this.colour = colour;
        this.vehicleNumber = vehicleNumber;
    }

    void displayInfo() {
        System.out.println("Brand: " + brand);
        System.out.println("Max Speed: " + maxSpeed + " km/h");
        System.out.println("Colour: " + colour);
        System.out.println("Vehicle Number: " + vehicleNumber);
    }
}

class Car extends MotorVehicle {
    String fuelType;

    public Car(String brand, double maxSpeed, String colour, String
vehicleNumber, String fuelType) {
        super(brand, maxSpeed, colour, vehicleNumber);
        this.fuelType = fuelType;
    }


    void displayInfo() {
        System.out.println("\nCar Details:");
        super.displayInfo();
        System.out.println("Fuel Type: " + fuelType);
    }

    void displayInfo(boolean displayNumber) {
        if (displayNumber) {
            System.out.println("Vehicle Number: " + vehicleNumber);
        } else {
            System.out.println("Vehicle Number not displayed.");
        }
    }
}

class ElectricCar extends Car {
    int batteryCapacity;
    int chargeRemaining;

    public ElectricCar(String brand, double maxSpeed, String colour, String
vehicleNumber, String fuelType,
                       int batteryCapacity, int chargeRemaining) {
        super(brand, maxSpeed, colour, vehicleNumber, fuelType);
        this.batteryCapacity = batteryCapacity;
        this.chargeRemaining = chargeRemaining;
    }


    void displayInfo() {
        System.out.println("\nElectric Car Details:");
        super.displayInfo();
        System.out.println("Battery Capacity: " + batteryCapacity + " kWh");
        System.out.println("Charge Remaining: " + chargeRemaining + "%");
    }

    void displayInfo(boolean displayNumber, int chargeRemaining) {
        if (displayNumber) {
            System.out.println("Vehicle Number: " + vehicleNumber);
        }
        System.out.println("Charge Remaining: " + chargeRemaining + "%");
    }
}

public class Driver{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        System.out.println("Enter MotorVehicle Details:");
        System.out.print("Brand: ");
        String mvBrand = sc.nextLine();
        System.out.print("Max Speed: ");
        double mvSpeed = sc.nextDouble();
        sc.nextLine();
        System.out.print("Colour: ");
        String mvColour = sc.nextLine();
        System.out.print("Vehicle Number: ");
        String mvNumber = sc.nextLine();
        MotorVehicle mv = new MotorVehicle(mvBrand, mvSpeed, mvColour,
mvNumber);


        System.out.println("\nEnter Car Details:");
        System.out.print("Brand: ");
        String carBrand = sc.nextLine();
        System.out.print("Max Speed: ");
        double carSpeed = sc.nextDouble();
        sc.nextLine();
        System.out.print("Colour: ");
        String carColour = sc.nextLine();
        System.out.print("Vehicle Number: ");
        String carNumber = sc.nextLine();
        System.out.print("Fuel Type: ");
        String fuelType = sc.nextLine();
        Car car = new Car(carBrand, carSpeed, carColour, carNumber, fuelType);


        System.out.println("\nEnter Electric Car Details:");
        System.out.print("Brand: ");
        String ecBrand = sc.nextLine();
        System.out.print("Max Speed: ");
        double ecSpeed = sc.nextDouble();
        sc.nextLine();
        System.out.print("Colour: ");
        String ecColour = sc.nextLine();
        System.out.print("Vehicle Number: ");
        String ecNumber = sc.nextLine();
        System.out.print("Fuel Type: ");
        String ecFuel = sc.nextLine();
        System.out.print("Battery Capacity (kWh): ");
        int batteryCap = sc.nextInt();
        System.out.print("Charge Remaining (%): ");
        int chargeRemain = sc.nextInt();

        ElectricCar eCar = new ElectricCar(ecBrand, ecSpeed, ecColour,
ecNumber, ecFuel, batteryCap, chargeRemain);


        System.out.println("\n==================== OUTPUT
====================");

        System.out.println("\n--- MotorVehicle ---");
        mv.displayInfo();
```

```java
        System.out.println("\n--- Car ---");
        car.displayInfo();
        car.displayInfo(true);

        System.out.println("\n--- ElectricCar ---");
        eCar.displayInfo();
        eCar.displayInfo(true, chargeRemain);

        sc.close();
    }
}
```



Test cases :

a.display() →

```
Brand : Tata
maxspeed : 90
Colour : Blue
```

b.display() →

```
Brand : Tata
maxspeed : 90
Colour : Blue
fuelType : petrol
```

b.display(True) →

```
Brand: Tata
maxspeed: 90
Colour: Blue
fuelType: petrol
Numer : TN6547
```

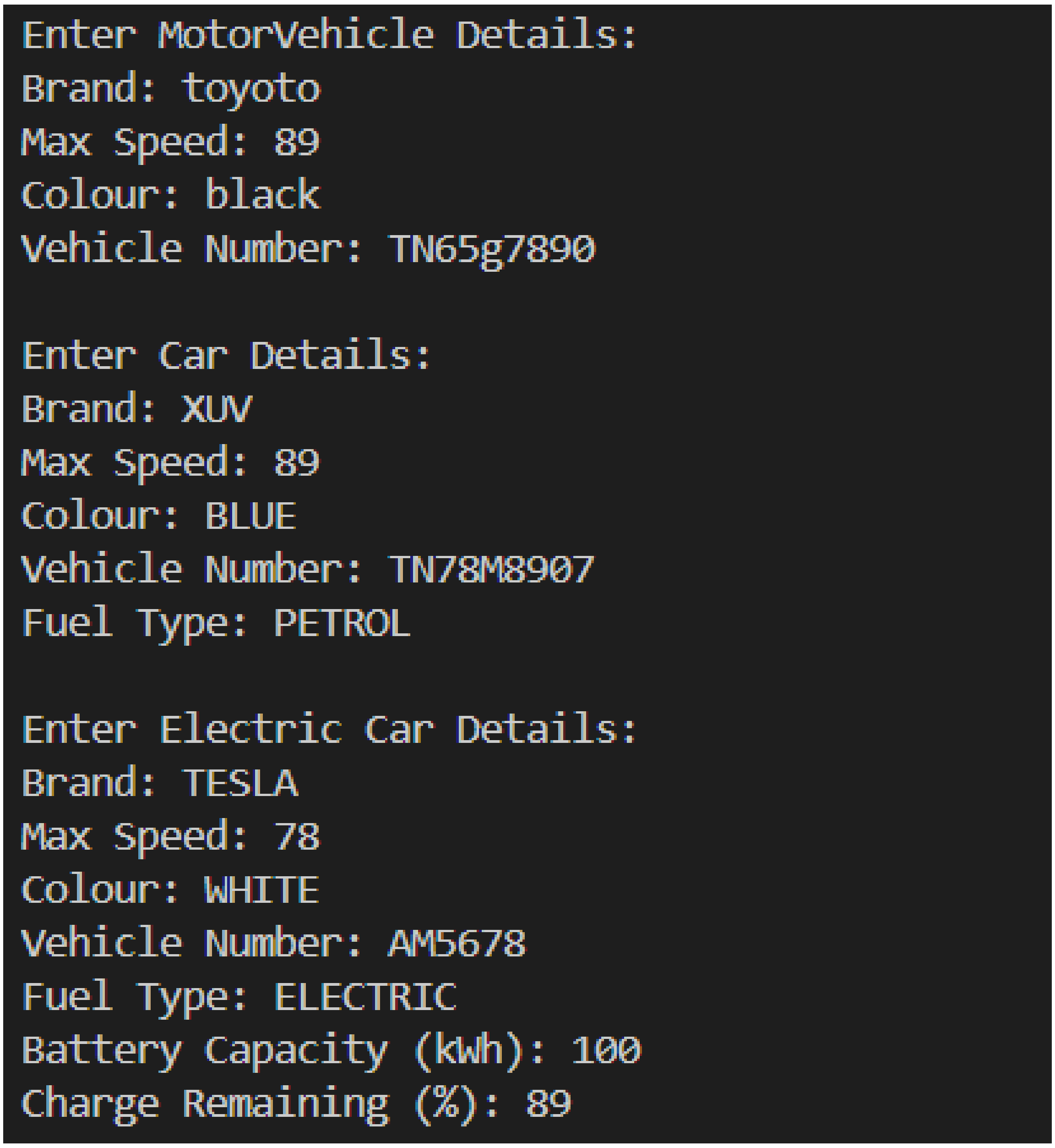
c.display(   ) →

```
Brand : Tata
maxspeed : 80
Colour : Black
fuelType : Electric
```

c.display ( ) →

```
Brand : Tata
maxspeed : 80
Colour : Black
```

OUTPUT:



```
Enter MotorVehicle Details:
Brand: toyoto
Max Speed: 89
Colour: black
Vehicle Number: TN65g7890

Enter Car Details:
Brand: XUV
Max Speed: 89
Colour: BLUE
Vehicle Number: TN78M8907
Fuel Type: PETROL

Enter Electric Car Details:
Brand: TESLA
Max Speed: 78
Colour: WHITE
Vehicle Number: AM5678
Fuel Type: ELECTRIC
Battery Capacity (kWh): 100
Charge Remaining (%): 89
```

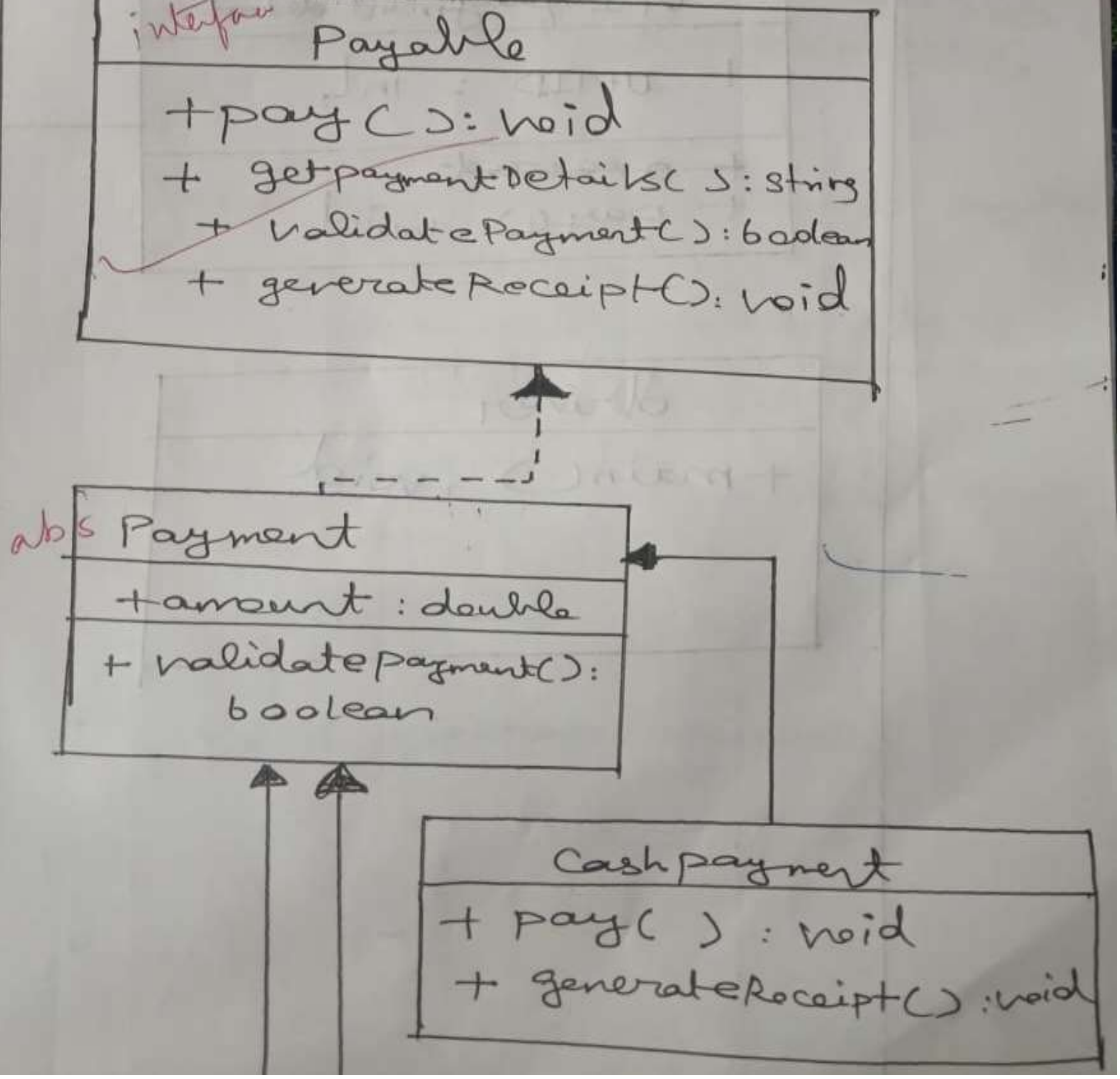
Q2: Design an interface Payable and an abstract class Payment that implements Payable. Classes CashPayment, CreditCardPayment, and UPIPayment should extend Payment. Each payment type should override a method to show its unique payment details.

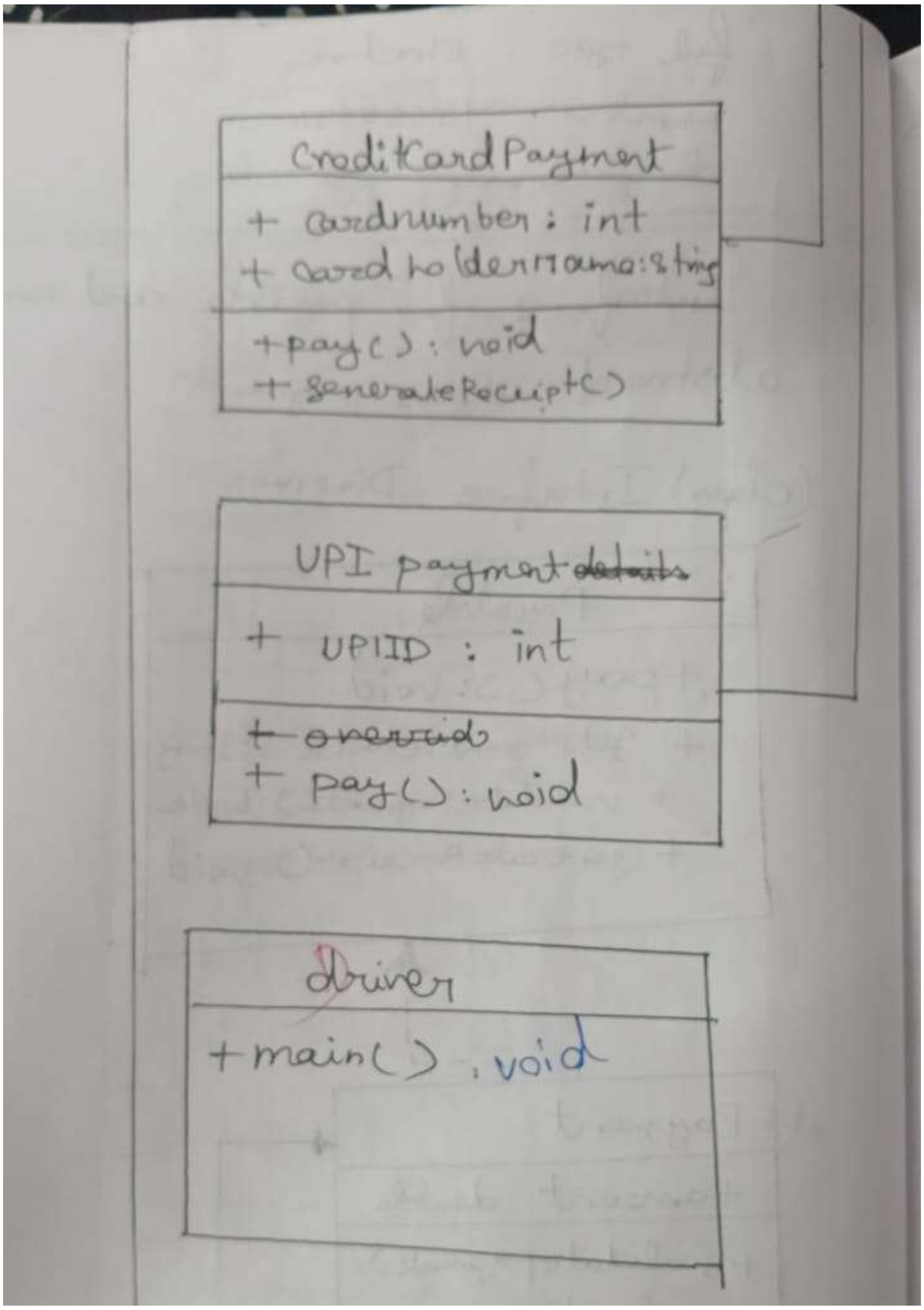
1. Create an interface `Payable` with methods:
   - void `pay()`;
   - String `getPaymentDetails()`;
   - boolean `validatePayment()`;
   - void `generateReceipt()`;
2. Create an abstract class Payment that:
   - Has a field `amount`
   - Implements `Payable`
   - Defines `validatePayment()` to return `true` if amount > 0.
3. Create a class `CashPayment` that:
   - Inherits `Payment`
   - Overrides `pay()` and `generateReceipt()` to display a cash payment specific message
4. Create a class `CreditCardPayment` that:
   - Adds fields for card number and cardholder name
   - Overrides pay() to display a credit card payment specific message
   - Overrides generateReceipt() to ask if the payer needs a receipt and displays appropriate messages based on their input.
5. Create a class `UPIPayment` that:
   - Adds UPI ID
   - Overrides pay() to show UPI payment details
6. Suggest a method where you can apply overloading, and demonstrate its use.
7. Create a driver class that has the `main()` method inside which:
   - Create objects of each subclass and call their `pay()` methods
   - Use an array of `Payment` references to demonstrate runtime polymorphism using all three types of payment
   - Demonstrate the application of overloading by invoking the overloaded methods as appropriate.

Number : TN658532
chargeremaing : 50

Interface for payable and an abstract class payment

(class) Interface Diagram

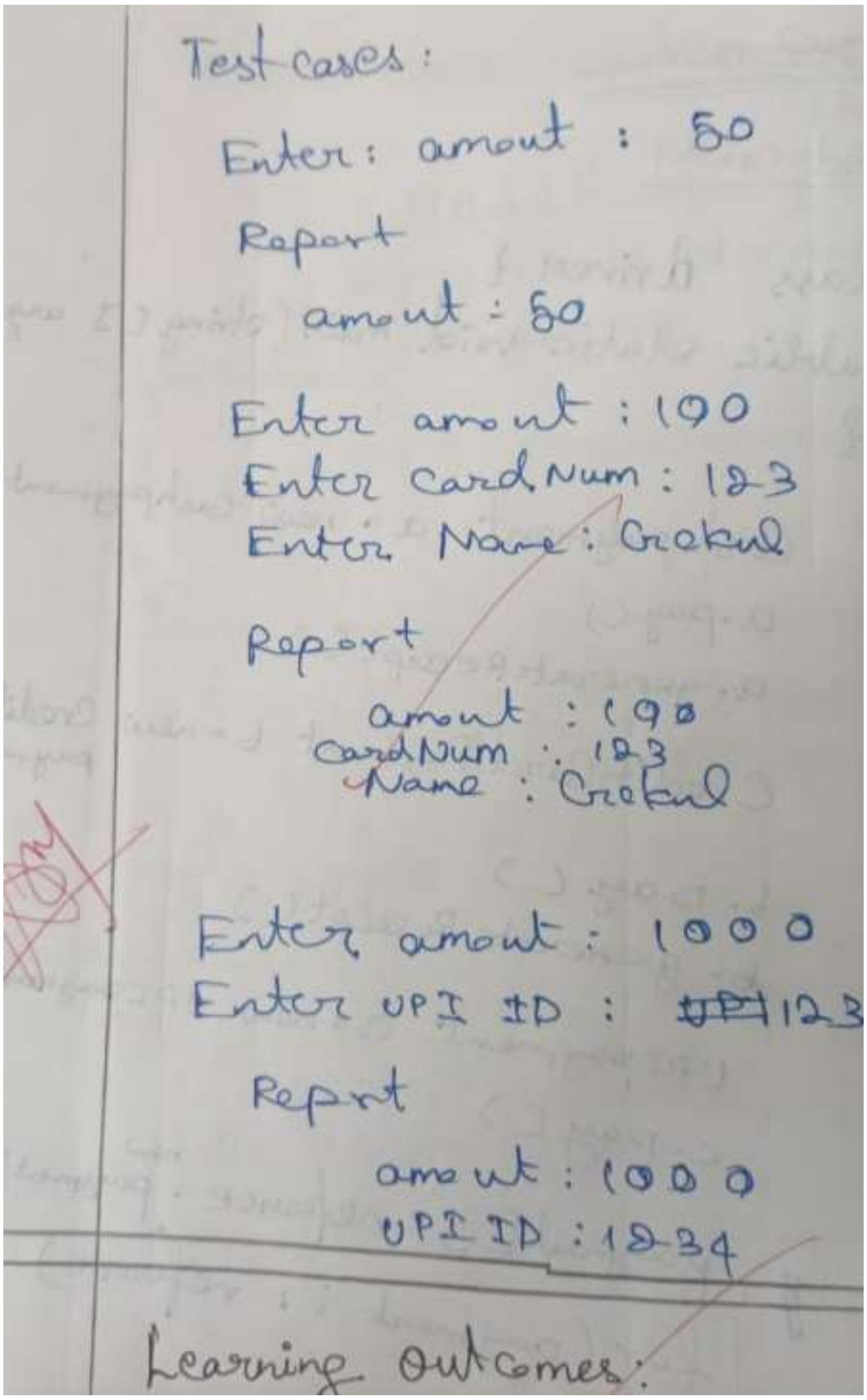

interface Payable
+pay ( ): void
+ getpaymentDetails( ): string
+ validatePayment( ): boolean
+ generateReceipt( ): void
abs Payment
+amount : double
+ validatepayment( ): boolean
Cashpayment
+ pay( ) : void
+ generateReceipt( ) : void



CODE:

```
import java.util.Scanner;

interface Payable {
    void pay();
    String getPaymentDetails();
    boolean validatePayment();
    void generateReceipt();
```

```java
}

abstract class Payment implements Payable {
    protected double amount;

    public Payment(double amount) {
        this.amount = amount;
    }

    public boolean validatePayment() {
        return amount > 0;
    }
}

class CashPayment extends Payment {
    public CashPayment(double amount) {
        super(amount);
    }

    public void pay() {
        System.out.println("Cash Payment of ₹" + amount + " received.");
    }

    public String getPaymentDetails() {
        return "Payment Type: Cash, Amount: ₹" + amount;
    }

    public void generateReceipt() {
        System.out.println("\n\n=== Receipt ===");
        System.out.println("Payment Method: " +
this.getClass().getSimpleName());
        System.out.println("Amount Paid: ₹" + amount);
        System.out.println("================");
    }
}

class CreditCardPayment extends Payment {
    private String cardNumber;
    private String cardHolder;

    public CreditCardPayment(double amount, String cardNumber, String
cardHolder) {
        super(amount);
        this.cardNumber = cardNumber;
        this.cardHolder = cardHolder;
    }

    public void pay() {
        System.out.println("Credit Card Payment of ₹" + amount + " received 
from " + cardHolder);
    }

    public String getPaymentDetails() {
        return "Payment Type: Credit Card, Card Holder: " + cardHolder + ", 
Card No: " + cardNumber;
    }

    public void generateReceipt() {
        System.out.println("=== Receipt ===");
        System.out.println("Payment Method: " + 
this.getClass().getSimpleName());
        System.out.println("Amount Paid: ₹" + amount);
        System.out.println("================");
    }
}

class UPIPayment extends Payment {
    private String upiID;

    public UPIPayment(double amount, String upiID) {
        super(amount);
        this.upiID = upiID;
    }

    public void pay() {
        System.out.println("UPI Payment of ₹" + amount + " sent to " + upiID);
    }

    public String getPaymentDetails() {
        return "Payment Type: UPI, UPI ID: " + upiID + ", Amount: ₹" + amount;
    }

    public void generateReceipt() {
        System.out.println("\n\n=== Receipt ===");
        System.out.println("Payment Method: " + 
this.getClass().getSimpleName());
        System.out.println("Amount Paid: ₹" + amount);
        System.out.println("================");
    }
}

class OverloadDemo {
    public void printDetails(String details) {
        System.out.println(details);
    }
    public void printDetails(String details, double amount) {
        System.out.println(details + " | Amount: ₹" + amount);
    }

    public void printDetails(Payment payment) {
        System.out.println("Details: " + payment.getPaymentDetails());
    }
}

public class PaymentSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        CashPayment cash = new CashPayment(500);
        CreditCardPayment card = new CreditCardPayment(1200,
"1234567890123456", "Alice");
        UPIPayment upi = new UPIPayment(750, "alice@upi");

        Payment[] payments = {cash, card, upi};

        System.out.println("\n--- Payment Processing ---");
        for (Payment p : payments) {
            if (p.validatePayment()) {
                p.pay();
                p.generateReceipt();
                System.out.println(p.getPaymentDetails());
                System.out.println("----------------------");
            } else {
                System.out.println("Invalid Payment");
            }
        }

        System.out.println("\n--- Method Overloading Demo ---");
        OverloadDemo od = new OverloadDemo();
        od.printDetails("Simple message");
        od.printDetails("Payment Done", 1500);
        od.printDetails(card);
    }
}
```



Test cases:

Enter: amout : 50

Report

amout : 50

Enter amout : 190
Enter card Num : 123
Enter Name : Gokul

Report

amout : 190
CardNum : 123
Name : Gokul

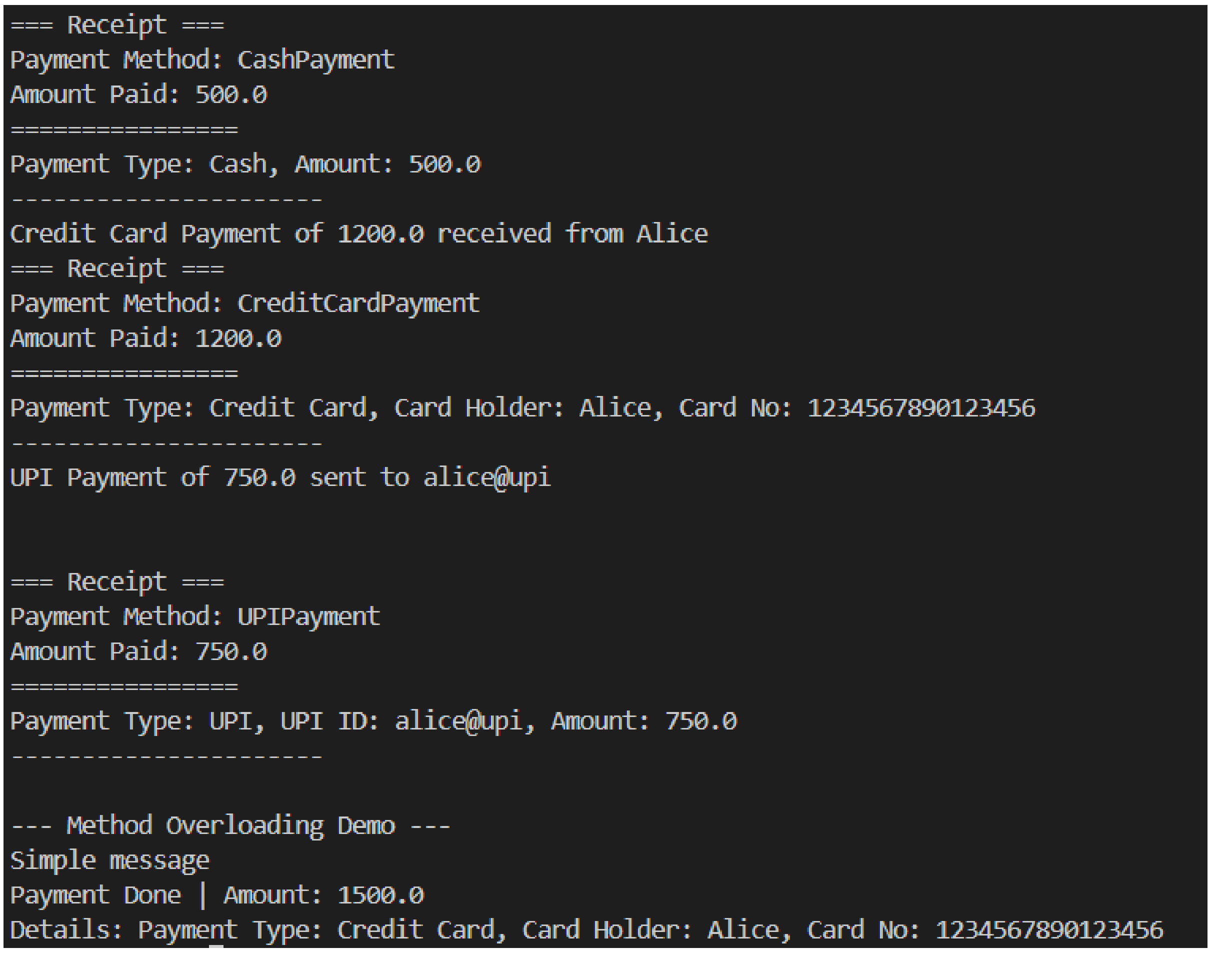
Enter amout : 1000
Enter UPI ID : ~~UPI~~123

Repart

amout : 1000
UPI ID : 1234

Learning outcomes:

OUTPUT:



```
=== Receipt ===
Payment Method: CashPayment
Amount Paid: 500.0
================
Payment Type: Cash, Amount: 500.0
----------------------
Credit Card Payment of 1200.0 received from Alice
=== Receipt ===
Payment Method: CreditCardPayment
Amount Paid: 1200.0
================
Payment Type: Credit Card, Card Holder: Alice, Card No: 1234567890123456
----------------------
UPI Payment of 750.0 sent to alice@upi


=== Receipt ===
Payment Method: UPIPayment
Amount Paid: 750.0
================
Payment Type: UPI, UPI ID: alice@upi, Amount: 750.0
----------------------

--- Method Overloading Demo ---
Simple message
Payment Done | Amount: 1500.0
Details: Payment Type: Credit Card, Card Holder: Alice, Card No: 1234567890123456
```

LEARNING OUTCOMES:

- LEARNED TO IMPLEMENT INTERFACE
- LEARNED TO USE ABSTRACT CLASS
- LEARNED TO USE SUPER() IN RIGHT PLACE