



Programming Test Results (With Test Cases)

Result Summary

Field	Value
Test ID	41163
Student ID	29195
Programs (with test cases)	1
Total Test Cases	6
Test Cases Passed	6
Fully Passed Programs	1
Partially Passed Programs	0
Failed Programs	0
Overall % (with test cases)	100.00%
Grade	Outstanding

Programs With Test Cases

#	Program Name	Total TC	Passed	Success Rate	Score /10	Submitted At	Attempts
1	Payment Processing Using Method Overloading	6	6	100.0%	10	28/11/2025, 12:38:17	0

Program Details (With Test Cases)

Program 1: Payment Processing Using Method Overloading

Languages: Java

Score (010):	10 / 10
Test Case Summary:	Total: 6 Passed: 6
	Failed: 0 Success: 100.0%
Attempts:	0
Submitted At:	28/11/2025, 12:38:17
Description:	Develop a scenario based program by using Method Overloading for processing the payment using different available options like Cash Payment, Credit Card and Debit Card Payment.

Create one BLC class called Payment.

Write 3 overloaded methods makePayment()

1) Method Name : makePayment()

Argument : 1 argument of type double amount

Return Type : void

Access modifier : public

In this method validate the parameter variable by using one Helper method (private method) validateAmount() to take only positive amount [Description is given below]

2) Method Name : makePayment()

Argument : 3 arguments [String cardHolderName, String creditCardNumber,double amount]

Return Type : void

Access modifier : public

In this method validate the parameter variable amount and creditCardNumber through Helper methods, amount must be positive integer and creditCardNumber must be of 16 digits exactly.

3) Method Name : makePayment()

Argument : 2 arguments [String debitCardNumber, double amount]

Return Type : void

Access modifier : public

In this method validate the parameter variable amount and debitCardNumber though Helper methods, amount must be positive integer and debitCardNumber must be of 16 digits exactly.

HELPER METHODS (Must be private so accessible from inside Payment class only)
[For more about helper method, read the notes, Access modifier topic, public access modifier]

1) Method Name : validateAmount()
Argument : 1 argument double amount

Return Type : boolean
Access modifier : private

In this method validate the amount, if amount is 0 or less than 0, print an error message and return false otherwise return true.

2) Method Name : validateCardNumber()
Argument : 1 argument String cardNumber.

Return Type : boolean
Access modifier : private

In this method validate the card number for its length, if length is exactly 16 digits then return true otherwise return false.

3) Method Name : maskCardNumber()
Argument : 1 argument String cardNumber.

Return Type : String
Access modifier : private

By using this method we should display only last 4 digit of card (Credit OR Debit both) only.[See the Test cases for more details in the below of this question]

Create an ELC class called PaymentProcess.

Inside main method display the following details as a Menu.

```
System.out.println("Payment Menu");
System.out.println("Please select any one Payment Method from the Menu :");
System.out.println("\t\t1) Payment by using Cash ");
System.out.println("\t\t2) Payment by using Credit Card ");
System.out.println("\t\t3) Payment by using Debit Card ");
```

Write Switch case with Scanner class to make the payment through different Options :

Sample Input Format

For Cash Payment

Choice: 1

Amount: <positive amount>

For Credit Card Payment

Choice: 2

Card Holder Name: <name>

Credit Card Number: <16-digit number>

Amount: <positive amount>

For Debit Card Payment

Choice: 3

Debit Card Number: <16-digit number>

Amount: <positive amount>

Processing payment via Credit Card...

Card Holder: <name>

Card Number: *-*-**-<last 4 digits>

Amount Paid RS :<amount>

Payment Successful!

Constraints:

-

Sample Input:

1 1200

Sample Output:

Processing payment via Cash... Amount Paid RS :12000.0 Payment Successful!

Explanation:

Payment Menu Please select any one Payment Method from the Menu : 1) Payment by using Cash 2) Payment by using Credit Card 3) Payment by using Debit Card Please enter your Payment choice [1/2/3] 1 Enter the amount you want to pay through cash : 12000 Processing payment via Cash... Amount Paid RS :12000.0 Payment Successful!

Solution Code

```
void main(){
    int choice = Integer.parseInt(IO.readln());
    Payment payment = new Payment();
    switch(choice){
        case 1 ->{
            double amount =Double.parseDouble(IO.readln());
            payment.makePayment(amount);
        }
        case 2 ->{
            String cardHolderName = IO.readln();
            String cardNumber = IO.readln();
            double amount = Double.parseDouble(IO.readln());
            payment.makePayment(cardHolderName,cardNumber,amount);
        }
        case 3 ->{
            String cardNumber = IO.readln();
            double amount = Double.parseDouble(IO.readln());
            payment.makePayment(cardNumber,amount);
        }
    }
}
```

```

}

class Payment{

    private boolean validateAmount(double amount){
        if(amount>0){
            return true;
        }
        else{
            return false;
        }
    }

    public void makePayment(double amount){
        if(validateAmount(amount)){
            IO.println("Processing payment via Cash....");
            IO.println("Amount Paid RS :" +amount);
            IO.println("Payment Successful!");
        }
        else{
            IO.println("Error: Amount must be greater than zero.");
        }
    }

    private boolean validateCardNumber(String cardNumber){
        //if(cardNumber.is){
        if(cardNumber.length() == 16){
            return true;
        }
        else{
            return false;
            //System.exit(0);
        }
    }

    public void makePayment(String cardHolderName, String creditCardNumber, double amount){
        if(validateAmount(amount) && validateCardNumber(creditCardNumber)){
            int count = 1;
            IO.println("Processing payment via Credit Card....");
            IO.println("Card Holder: " +cardHolderName);
            char arr [] = cardHolderName.toCharArray();

```

```

        IO.print("Card Number: ****_****_****_"+creditCardNumber.substring(12));

        IO.println();
        IO.println("Amount Paid RS :" + amount);
        IO.println("Payment Successful!");
    }
    else{
        IO.println("Error: Invalid card number. It must be 16 digits.");
        System.exit(0);
    }
}

private String maskCardNumber(String cardNumber){
    return "****_****_****_"+cardNumber.substring(12);
}

public void makePayment(String cardNumber,double amount){
    if(validateCardNumber(cardNumber)&&validateCardNumber(cardNumber)){
        IO.println("Processing payment via Debit Card...");
        IO.println("Card Number: "+maskCardNumber(cardNumber));
        IO.println("Amount Paid RS :" + amount);
        IO.println("Payment Successful!");
    }
    else{
        IO.println("Error: Invalid card number. It must be 16 digits.");
        System.exit(0);
    }
}
}

```