Programming Test Results (With Test Cases)

Result Summary

Field	Value
Test ID	41240
Student ID	29227
Programs (with test cases)	3
Total Test Cases	12
Test Cases Passed	12
Fully Passed Programs	3
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	Test-LooseCouplingNotification Demo	3	3	100.0%	10	13/12/2025, 10:12:42	0
2	TestEmployeeUsingFunction	4	4	100.0%	10	13/12/2025, 09:57:21	0
3	Combine Two Predicates Using and() Method	5	5	100.0%	10	13/12/2025, 09:44:15	0

Program Details (With Test Cases)

Program 1: Test-LooseCouplingNotificationDemo

Languages:	Java	
Score (010):	10 / 10	
Test Case Summary:	Total: 3	Passed: 3
	Failed: 0	Success: 100.0%
Attempts:	0	
Submitted At:	13/12/2025, 10:12:42	

Description:

Create a Java program to demonstrate loose coupling and polymorphism using an interface and its implementing classes.

You must define an interface NotificationHub that contains two abstract methods: sendAlert() and sendReport().

Create two classes that implement this interface:

EmailNotification

SMSNotification

Each class must provide its own version of alert and report messages.

In the ELC class, write a static polymorphic method named:

"processNotification"

This method should call the two interface methods (sendAlert() and sendReport()) based on the object passed.
This proves loose coupling because the method works with any class implementing the interface.

In the main method:

Take user input (email or sms)

Create the corresponding object

Pass it to the static polymorphic method

The correct classs methods must execute based on the object provided

This demonstrates how polymorphism removes direct dependency on specific classes and enables loose coupling.

Constraints:	User must enter either "email" or "sms" All printing must happen from inside implemented methods, not main Static method must accept interface reference only No direct object-specific method should be called from main
Sample Input:	email
Sample Output:	Email Alert Sent Email Report Generated
Explanation:	Polymorphism allows the static method to work with any object that implements NotificationHub. calls SMSs alert & report.

Solution Code

```
void main()
{
    String s= IO.readln();
    if(s.startsWith("s"))
    {
        SmsN sms= new SmsN();
        sms.sendAlert("");
        sms.sendReport("");

    }
    else if(s.startsWith("e"))
    {
        EmailN email = new EmailN();
        email.sendAlert("");
        email.sendReport("");
    }
    else{
        IO.println("Invalid notification type!");
    }

}
interface NotificationHub
{
    void sendAlert(String str);
    void sendReport(String str);
}
class EmailN implements NotificationHub
{
    public void sendAlert(String str)
    {
        IO.println("Email Alert Sent");
    }
    public void sendReport(String str)
    {
        IO.println("Email Report Generated");
    }
}
class SmsN implements NotificationHub
{

    public void sendAlert(String str)
    {
        IO.println("SMS Alert Sent");

    }
    public void sendReport(String str)
    {
        IO.println("SMS Report Generated");

    }
}
```

Program 2: TestEmployeeUsingFunction

Languages:	Java	
Score (010):	10 / 10	
Test Case Summary:	Total: 4	Passed: 4
	Failed: 0	Success: 100.0%
Attempts:	0	
Submitted At:	13/12/2025, 09:57:21	

Description:

You are required to write a Java program that demonstrates the use of the java.util.function.Function<T, R> functional interface with either:

A lambda expression, OR

A method reference

Your task is to:

Create an Emp class with:

int empNo

}

String name

float salary

Read employee details from user input.

Create a Function<Emp, Float> to:

Add 5000.00 bonus to the employee's salary.

Return the updated salary.

Apply the function and print the latest salary.

You may use either a lambda expression or a method reference.

Constraints:	Salary must be positive. Use Function<Emp, Float> only. Bonus is fixed: 5000.00 Input must be taken from the user.
Sample Input:	101 Rahul 45000
Sample Output:	Latest Salary: 50000.0
Explanation:	-

Solution Code

```
import java.util.function.*;
public class Emp{
void main()
{
    Integer id=Integer.parseInt(IO.readln());
    String name= IO.readln();
    Double salary=Double.parseDouble(IO.readln());
    Function<Emp,Double> sal= (emp)->{
        final Double usal=salary+5000.0;
        return usal;
    };
    Emp em = new Emp();
    IO.println("Latest Salary: "+sal.apply(em));

}
}
```

Program 3: Combine Two Predicates Using and() Method

Languages:	Java	
Score (010):	10 / 10	
Test Case Summary:	Total: 5	Passed: 5
	Failed: 0	Success: 100.0%
Attempts:	0	
Submitted At:	13/12/2025, 09:44:15	

Description: ou are required to write a Java program that demonstrates how to combine two Predicate conditions using the and() method from the java.util.function.Predicate interface.

Your task:

Accept an integer input N.

Create two predicates:

Predicate 1: Check if the number is even.

Predicate 2: Check if the number is greater than 10.

Combine both predicates using the and() method.

Evaluate the input number using the combined predicate.

Print whether the number satisfies both conditions.

Constraints: Input must be an integer. 1 N 109 You must use: Predicate<Integer> test() and() method

Sample Input: 12

Sample Output: Number satisfies both predicates: true

Explanation: Predicate 1 returns true if the number is even. Predicate 2 returns true if the number is greater than 10. The and() method combines them so the number must satisfy both conditions. Example: Input: 12 Even? Greater than 10? Combined result:

Solution Code

```
import java.util.function.*;

void main()
{
    Integer n= Integer.parseInt(IO.readln());
    //IO.print(n);
    boolean isValid=false;
    Predicate<Integer> even=(num)->num%2==0;
    Predicate<Integer> greater=(num)->num>10;
    boolean b1=even.test(n);
    boolean b2 = greater.test(n);
    if(b1 && b2)
    {
        isValid=true;
        IO.print("Number satisfies both predicates: "+isValid);
    }
    else{
        IO.print("Number satisfies both predicates: "+isValid);
    }
}
```