DAY 24:

ASSIGNMNET 3:

Task 5: Functional Interfaces

Create a method that accepts functions as parameters using Predicate, Function, Consumer, and Supplier interfaces to operate on a Person object.

ANSWER:

```
import java.util.function.Consumer;

import java.util.function.Function;

import java.util.function.Predicate;

import java.util.function.Supplier;


public class Main {

  public static void main(String[] args) {

    Product product = new Product("Phone", 500);


    // Example usage of the method with various functions

    processProduct(product,

        p -> p.getPrice() > 100, // Predicate: Check if price is greater than 100

        p -> p.getPrice() * 0.9, // Function: Apply 10% discount to the price

        discountPrice -> System.out.println("Discounted Price: $" + discountPrice), // Consumer:
Print the discounted price

        () -> new Product("Laptop", 1000)); // Supplier: Provide a default Product if the predicate
fails

  }


  static void processProduct(Product product,

                Predicate<Product> predicate,

                Function<Product, Double> function,

                Consumer<Double> consumer,

                Supplier<Product> supplier) {
```

```
        if (predicate.test(product)) {
            double discountPrice = function.apply(product);
            consumer.accept(discountPrice);
        } else {
            Product defaultProduct = supplier.get();
            double defaultPrice = defaultProduct.getPrice();
            double discountPrice = function.apply(defaultProduct);
            consumer.accept(discountPrice);
        }
    }
}

class Product {
    private String name;
    private double price;

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }
}
```

In this example:

- We define a Product class with name and price fields.

- We define a method processProduct that accepts a Product object and functions of type Predicate<Product>, Function<Product, Double>, Consumer<Double>, and Supplier<Product>.

- Inside the processProduct method, we use these functional interfaces to perform operations on the Product object based on the provided functions.

- In the main method, we demonstrate how to use the processProduct method with various functions to operate on a Product object, such as applying a discount to the price.