DAY 24:
ASSIGNMENT 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.

```
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) {
        Person person = new Person("Alice", 30);

        // Example usage of the method with various functions
        processPerson(person,
                p -> p.getAge() > 25, // Predicate: Check if age is greater than 25
                p -> "Hello, " + p.getName(), // Function: Transform Person to a greeting message
                System.out::println, // Consumer: Print the result
                () -> new Person("Bob", 25)); // Supplier: Provide a default Person if the predicate fails
    }

    static void processPerson(Person person,
                              Predicate<Person> predicate,
                              Function<Person, String> function,
```

```
                        Consumer<String> consumer,
                        Supplier<Person> supplier) {
        if (predicate.test(person)) {
            String result = function.apply(person);
            consumer.accept(result);
        } else {
            Person defaultPerson = supplier.get();
            String result = function.apply(defaultPerson);
            consumer.accept(result);
        }
    }
}

class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}
```

In this example:

- We define a Person class with name and age fields.

- We define a method processPerson that accepts a Person object and functions of type Predicate<Person>, Function<Person, String>, Consumer<String>, and Supplier<Person>.

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

- In the main method, we demonstrate how to use the processPerson method with various functions to operate on a Person object.