Predicate

Programa Predicate

Las expresiones lambda son una característica que permite definir de manera concisa y funcional métodos anónimos. Los predicados son una interfaz funcional (`java.util.function`) que se utiliza comúnmente con expresiones lambda para expresar condiciones booleanas. La interfaz funcional `Predicate<T>` toma un tipo de dato genérico `T` y tiene un único método llamado `test` que recibe un argumento de tipo `T` y devuelve un valor booleano.

En este caso se toma de base el programa anterior (el del patrón Observer) y se hacen unas modificaciones en el código de las clases, donde se determina si el resultado es true o false por medio de un algoritmo. En este caso al usar `Predicate` quedan de la siguiente manera las clases (Los otros archivos se mantienen igual).

```
//-----------------------Numero Primo-------------------------

import java.util.function.Predicate;

public class Primo extends Observer {

    int num;

    public Primo(int num, Operaciones op){
        super(op);
        this.num = num;
    }

    Predicate<Integer> esPrimo = (numero) -> {
        for (int i = 2; i < numero; i++) {
            if (numero%i==0) {
                return false;
            }
        }
        return true;
    };

    @Override
    void update(){
        boolean resultado = esPrimo.test(this.num);
        System.out.println("El numero "+ this.num+ " es primo? : "
        + resultado);
```

```
    }

}
```

```
//----------------------Numero multiplo de otro-------------------------

import java.util.function.Predicate;

public class MultiplodeOtro extends Observer{
    int a;
    int b;

    public MultiplodeOtro(int a, int b, Operaciones op){
        super(op);
        this.a = a;
        this.b = b;
    }

    Predicate<Integer> esMultiplo = (numero) -> b % a == 0;

    @Override
    void update(){
        boolean resultado = esMultiplo.test(this.a);
        System.out.println("El numero " + this.a + " es multiplo de " + this.b
        + "? : " + resultado);
    }
}
```

```
//----------------------Factorial de un numero-------------------------

import java.util.function.Predicate;

public class Factorial extends Observer{

    int base;
    int fact;

    public Factorial(int base, int fact, Operaciones op){
        super(op);
        this.base = base;
        this.fact = fact;
    }

    Predicate<Integer> esFactorial = (numero) -> {
        if(base > fact){
            return false;
        }
```

```
        int aux=1;

        for (int i = 1; i <= base; i++) {
            aux *= i;
        }

        return fact == aux;

    };

    @Override
    void update(){
        boolean resultado = esFactorial.test(this.fact);
        System.out.println("El numero " + this.fact + " es el factorial de " + this.base
        + "? : " + resultado);
    }

}
```

En todos los casos solamente recibimos como parámetro a numero y adaptamos el algoritmo a su forma funcional. Después podemos usar el método test() y guardamos lo que devuelve en la variable resultado

Con estos cambios usando la expresión lambda y la interfaz funcional `Predicate` hace que el código sea más funcional y flexible. Ademas el funcionamiento del programa no presenta cambios

```
El numero 10 es primo? : false
El numero 107 es primo? : true
----------------------
El numero 10 es primo? : false
El numero 120 es el factorial de 5? : true
----------------------
El numero 10 es primo? : false
El numero 120 es el factorial de 5? : true
El numero 3 es multiplo de 10? : false
PS C:\Users\LENOVO\Desktop\patronObserver>
```

Resultado completo de la terminal