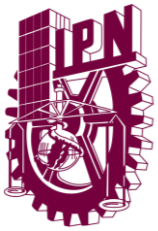

INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

INGENIERIA EN INTELIGENCIA ARTIFICIAL

PARADIGMAS DE PROGRAMACION

PRACTICA NUMERO 1

Funciones Puras, de primer orden y orden superior

VAZQUEZ PEREZ ANGEL MARTIN

GRUPO 3BV2

INTRODUCCION

En esta práctica estaremos empezando a utilizar la programación funcional, dentro de ellos primeros pasos que conlleva esta programación funcional son reconocer sobre los tipos de funciones que podemos programar, dentro de estas están las puras, las de primer orden que conllevan recursividad, y las de segundo orden.

Describiremos el comportamiento que este tipo de funciones tiene para después poder implementar funciones con estos métodos.

Funciones puras.- Estas funciones no tienen algún efecto secundario o algún cambio a los argumentos que le están llegando para este utilizarlos

Funciones de primer orden.- estas funciones pueden recibir otras funciones como argumento, para después usarlo y devolver un resultado, un caso muy común de este tipo es la recursividad directa, la cual nos indica que la función puede llamarse a sí mismo dentro de la misma función.

Funciones de segundo orden.-Estas funciones pueden operar dentro de otras funciones, nos permite una modularidad muy clara y entendible dentro del funcionamiento de todas las funciones presentes en el código.

Al terminar esta práctica vamos a poder dominar muy bien este tipo de funciones y esta forma de programar nos puede beneficiar mucho durante nuestra trayectoria en la programación, quizá desde ahorita no podemos ver del todo la función de la programación funcional, pero en un futuro al estar implementando este tipo de funciones nos daremos cuenta lo importante que es.

EJERCICIO 1

Aritmética básica con funciones puras

Sumar, restar, multiplicar, dividir y obtener el módulo de la división.

```java
public static int sumapura(int x,int y){ //λx.λy. x + y
    return x+y;
}
public static int restapura(int x,int y){//λx.λy. x - y
    return x-y;
}
public static int multiplicacionpura(int x,int y){ //λx.λy. x * y
    return x*y;
}
public static int divisionpura(int x,int y){ //λx.λy. x * y
    if(y==0 && x>0){
        return Integer.MAX_VALUE;
    }
    else if(y==0 && x<0){
        return Integer.MIN_VALUE;
    }
    return x/y;
}

public static int modulopuro(int x,int y){ //λx.λy. x % y
    if (y==0){
        return x;
    }
    return x%y;
}
```

En estas funciones puras que se muestran en la imagen de arriba, podemos afirmar que estas son puras, ya que nunca se afecta a las variables x, y para conocer su resultado.
Entonces estas funciones fueron fáciles de implementar ya que solo ocupamos los operadores aritméticos que java tiene integrado, para la resta, para la suma, para la multiplicación, para la división e incluso para el módulo, entonces para usar este tipo de funciones puras solo ocupamos que se le pasen 2 números enteros, para que este devuelva un valor.

Para el caso de la división se tuvo que añadir un poco más de código ya que hay una restricción, cuando el divisor es 0, en este caso nos estaremos encontrando con una excepción del lenguaje de Java ya que no se puede dividir entre 0, pero no queremos que esto suceda entonces retornamos un valor aunque no es lo correcto retornamos el valor mas grande que puede tomar un entero en el lenguaje de Java ya sea negativo o positivo.

EJERCICIO 2

Crear otra versión empleando funciones de primer orden (Recursividad directa).

En las funciones seguimos la lógica que vimos en clase, la suma recursiva el caso donde se va a detener es cuando nuestro número x que cada incremento estará disminuyendo 1 sea igual a 0 para así se vayan retornando todos los valores hasta conocer la suma. Va a ser parecido a la resta donde se va a estar decrementando 1 los 2 valores., en nuestra multiplicación es como tener muchas sumas nunca se hace la multiplicación en esta recursividad, división tiene varios factores cuando a sea menor que b o cuando sea igual va a devolver un 1

```
public static int sumarecursiva(int x,int y){ // = (λx.λy.(if (= x 0) y || (if (= y 0)x(sumarecursiva(- x 1),(+ y 1)))))
    if(x==0){
        return y;
    }
    if(y==0){
        return x;
    }
    return sumarecursiva(x-1,y+1);
}
public static int restarecursiva(int x,int y){//= (λx.λy.(if (= y 0)x (restarecursiva(- x 1),(- y 1)))))
    if (y==0){
        return x;
    }
    return restarecursiva(x-1,y-1);
}

public static int multiplicacionrecursiva(int x,int y){//= (λx.λy.(if (= y 0)0  (+ (multiplicacionrecursiva(x,(- y 1))))))
    if(y==0){
        return 0;
    }
    return x +multiplicacionrecursiva(x,y-1);
}

public static int divisionrecursiva(int x, int y) {// λx.λy. if (x < y) 0
                                                    //else if (x == y) 1 else 1 + divisionrecursiva(x - y, y)
     if(y==0 && x>0){
        return Integer.MAX_VALUE;
    }
    else if(y==0 && x<0){
        return Integer.MIN_VALUE;
    }
        if (x<y){
            return 0;
        }
        if (x==y){
            return 1;
        }
        else
            return 1+divisionrecursiva(x - y, y);
    }

public static int modulorecursivo(int x,int y){// λx.λy. if (y == 0) x else if (x < y) x else modulorecursivo(x - y, y)
    if(y==0){
        return x;
    }
    if (x<y){
        return x;
    }
    return modulorecursivo(x-y,y);
}
```

EJERCICIO 3

3. Una versión más con funciones de segundo orden en este tipo de funciones estarán llamando a otras funciones que sean recursivas sin tener operaciones puras dentro de todas las funciones, es por eso por lo que vemos funciones por todas partes, ningún otro operador aritmético. Entonces tendrá que tener la misma funcionalidad que las otras funciones anteriormente mencionadas.

```
public static int sumasegundo(int x,int y){// λx.λy. if (x == 0) y else
                                            //sumarecursiva(restarecursiva(x, 1), sumarecursiva(y, 1))
    if(x==0){
        return y;
    }
    return sumarecursiva(restarecursiva(x,1),sumarecursiva(y,1));
}
public static int restasegundo(int x,int y){// λx.λy. if (y == 0) x else
                                             //restarecursiva(restarecursiva(x, 1), restarecursiva(y, 1))
    if (y==0){
        return x;
    }
    return restarecursiva(restarecursiva(x,1),restarecursiva(y,1));
}

public static int multiplicacionsegundo(int x,int y){// λx.λy. if (y == 0) 0
                                                      //else sumarecursiva(x, multiplicacionrecursiva(x, restarecursiva(y, 1)))
    if(y==0){
        return 0;
    }
    return sumarecursiva(x,multiplicacionrecursiva(x,restarecursiva(y,1)));
}
```

```
public static int divisionsegundo(int x, int y) {// λx.λy. if (x < y) 0 else if (x == y) 1
                                                  //else sumarecursiva(1, divisionrecursiva(restarecursiva(x, y), y))
        if (x<y){
            return 0;
        }
        if (x==y){
            return 1;
        }
        else
            return sumarecursiva(1,divisionrecursiva(restarecursiva(x,y), y));
    }

public static int modulosegundo(int x,int y){// λx.λy. if (x < y) x
                                              //else modulorecursivo(restarecursiva(x, y), y)
    if (x<y){
        return x;
    }
    return modulorecursivo(restarecursiva(x,y),y);
}
```

INTERFACES.

Yo maneje 3 interfaces una para operaciones puras, la otra para las de primer orden y otra para orden superior, aunque sean los mismos parámetros vamos a realizarlo así, quizá ahí pude ahorrarme las interfaces y solo ocupar una, pero es para conocer el manejo de estas

```java
public interface Operacionespuras {
    int sumar(int x, int y);
    int restar(int x, int y);
    int multiplicar(int x, int y);
    int dividir(int x, int y);
    int modulo(int x,int y);
}
```

ASIGNACION DE LA INTERFAZ CON LAS FUNCIONES

Aquí asignamos nuestras funciones puras por medio de la interfaz para así ya poder usarlo. Se realiza en todas las funciones que tenemos cada una con su respectiva interfaz.

```java
public class Puras implements Operacionespuras {

    @Override
    public int sumar(int x, int y) {
        return Practical.sumapura(x, y);
    }
    @Override
    public int restar(int x, int y) {
        return Practical.restapura(x, y);
    }
    @Override
    public int multiplicar(int x, int y) {
        return Practical.multiplicacionpura(x, y);
    }
    @Override
    public int dividir(int x, int y) {
        return Practical.divisionpura(x, y);
    }
    @Override
    public int modulo(int x, int y) {
        return Practical.modulopuro(x, y);
    }
}
```

EJECUCION DEL PROGRAMA

En la ejecución del programa simplemente nos va a solicitar el valor de a y el valor de b, y después se van a mostrar todos los resultados obtenidos que nos devuelve cada función, ya sea por el método puro, el método de primer orden y el método del segundo orden

```
Este programa ejecuta todas las funciones programadas: puras, primer orden y segundo orden
Introduzca el valor de a
4
Introduzca el valor de b
9
METODO PURO
el numero a es 4 y el b es 9
Suma: 13
Resta: -5
Multiplicación: 36
División: 0
Módulo: 4

METODO PRIMER ORDEN
el numero a es 4 y el b es 9
Suma: 13
Resta: -5
Multiplicación: 36
División: 0
Módulo: 4

METODO SEGUNDO ORDEN
el numero a es 4 y el b es 9
Suma: 13
Resta: -5
Multiplicación: 36
División: 0
Módulo: 4
```

CONCLUSIONES

Ya nos adentramos muy bien a lo que es la programación funcional, donde ahora implementamos unas funciones muy sencillas que simplemente sumaran restaran multiplicaran y dividirán 2 números, todo esto siguiendo el paradigma funcional.

Fue muy fácil para nosotros ya implementar el método de segundo orden porque las funciones recursivas ya las teníamos, solo era cosa de estas implementarlas y juntarlas para que no tuviera operadores aritméticos dentro de esta, sino tuviera puras funciones que funcionen mediante nuestra interface.

Las interfaces son una herramienta muy buena dentro de Java, ya que al implementar una interfaz dentro de nuestra clase no otros podemos conocer que funciones y que tipos de valores va a recibir una función definida en nuestra interfaz para que nosotros dentro de la case retornemos le valor que nosotros queramos

BIBLIOGRAFIA

Rolfo, M. (2022, 24 octubre). *¿Qué es la programación funcional? Una guía práctica*.

Codigoencasa.com. https://codigoencasa.com/programacion-funcional/