

INTERFACES FUNCIONALES



Interface funcional

Parametros

Retorno

Metodo abs.

SUPPLIER<T>

O

T

GET()

En esta línea, se crea una instancia de la interfaz funcional Supplier<String>

```
Supplier<String> muSup = () -> "Mi primer lambda";  
String txt = muSup.get();  
System.out.println(txt);
```

: En esta línea, se llama al método get() y la almacena en la variable txt

se imprime en la consola el contenido de la variable txt, que en este caso será

"Mi primer lambda".

Interface funcional

Parametros

Retorno

Metodo abs.

CONSUMER<T>

1(T)

VOID

ACCEPT()

Creamos nuestra lista

```
List<Producto> listaProductos = new ArrayList<>();
```

Agregamos nuestros obj

```
listaProductos.add(new Producto(nombre: "Calavera Nissan", stock: 20, sueldo: 1500));
listaProductos.add(new Producto(nombre: "Llanta Michelin", stock: 4, sueldo: 3000));
listaProductos.add(new Producto(nombre: "Aceite de motor", stock: 10, sueldo: 500));
listaProductos.add(new Producto(nombre: "Espejo retrovisor", stock: 5, sueldo: 800));
listaProductos.add(new Producto(nombre: "Batería de coche", stock: 2, sueldo: 2500));
```

No devuelve nada

```
listaProductos.forEach(producto-> System.out.println(producto));
```

Solicita un consumer

```
forEach(Consumer<? super Producto> ... void
```

Interface funcional

Parametros

Retorno

Metodo abs.

BINARYOPERATOR<T> 2(T,T)

T

APPLY(T,T)

Definimos nuestra lambda

```
// Definimos una expresión lambda para sumar dos números
BinaryOperator<Integer> sumar = (a, b) -> a + b;
// Utilizamos la expresión lambda para sumar dos números
int resultado = sumar.apply(t: 5, u: 3);
System.out.println("La suma es : " + resultado);
```

Ejecutamos nuestra
lambda con su metodo
esta se almacena en
resultado

Imprime nuestro
resultado

Interface funcional

Parametros

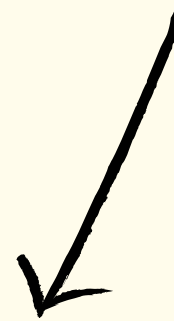
Retorno

Metodo abs.

BINARYOPERATOR<T> 2(T,T)

T

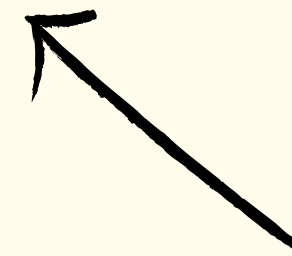
APPLY(T,T)



Definimos nuestra lambda

```
// Definimos una expresión lambda para sumar dos números
BinaryOperator<Integer> sumar = (a, b) -> a + b;
// Utilizamos la expresión lambda para sumar dos números
int resultado = sumar.apply(t: 5, u: 3);
System.out.println("La suma es : " + resultado);
```

Ejecutamos nuestra
lambda con su metodo
esta se almacena en
resultado



Imprime nuestro
resultado

Interface funcional

Parametros

Retorno

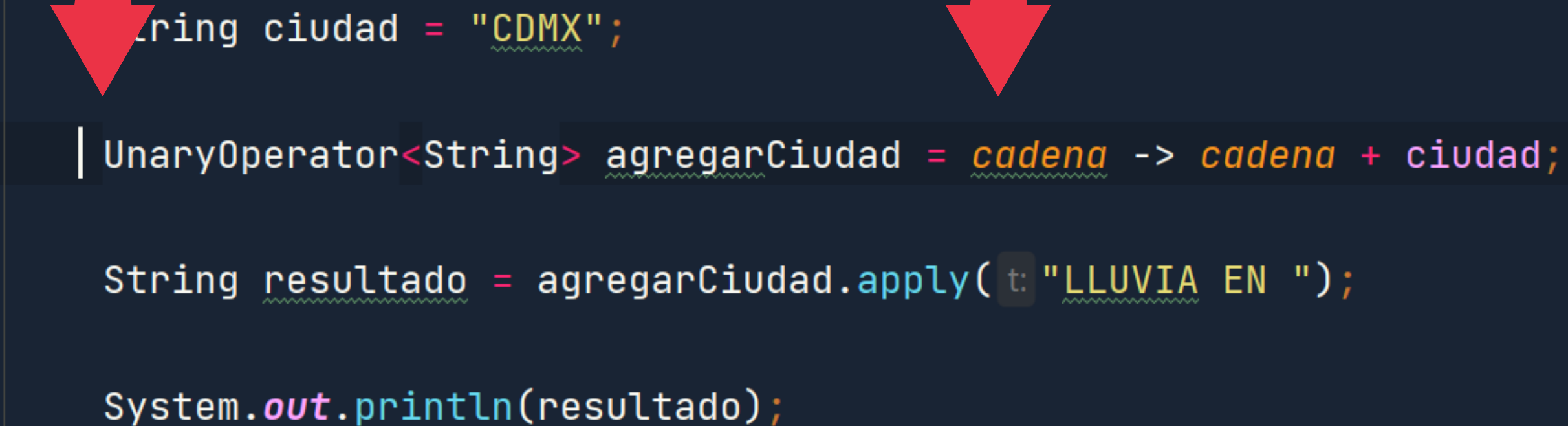
Metodo abs.

UNARYOPERATOR<T>

1(T)

T

APPLY(T,T)



```
String ciudad = "CDMX";

UnaryOperator<String> agregarCiudad = cadena -> cadena + ciudad;

String resultado = agregarCiudad.apply(t: "LLUVIA EN ");

System.out.println(resultado);
```

Definimos nuestra lambda

Ejecutamos nuestra
lambda con su metodo
esta se almacena en
resultado

Imprime nuestro
resultado

Interface funcional

Parametros

Retorno

Metodo abs.

PREDICATE<T>

1(T)

BOOLEAN

TEST(T)



```
Predicate<Integer> esMayorEdad = edad -> edad >= 18;
```

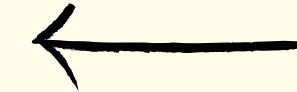
```
// Evaluamos la expresión lambda con un número  
boolean resultado = esMayorEdad.test(t: 17);
```

```
System.out.println("Puede votar : " + resultado);
```

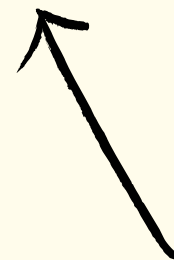
Definimos nuestra lambda



Ejecutamos nuestra
lambda con su metodo
esta se almacena en
resultado



Retorna true o false



Interface funcional

Parametros

Retorno

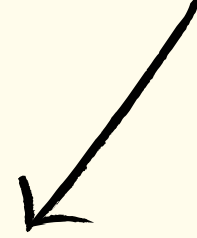
Metodo abs.

FUNCTION<T,R>

1(T)

R

APPLY(T)



```
Function<Integer, Integer> calcularCuadrado = numero -> numero * numero;
```

```
int resultado = calcularCuadrado.apply(t: 12);
```

```
System.out.println("El cuadrado del numero es: " + resultado);
```

Definimos nuestra
lambda



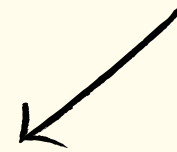
Lanzamos nuestra
lamda



Nos regresa el cuadrado
del numero



Definimos una lista de cadenas.



```
List<String> nombres = Arrays.asList("Alan", "Andrea", "Pedro");  
  
nombres.forEach(nombre -> System.out.println(nombre));
```



nombre de nuestra lista



Utilizamos una expresión lambda para
imprimir cada nombre en la lista.

Interface funcional

Parametros

Retorno

Metodo abs.

FUNCTION<T,R>

1(T)

R

APPLY(T)

```
Function<String, String> convertirAMayusculas = cadena -> cadena.toUpperCase();  
  
String resultadoMayusculas = convertirAMayusculas.apply(t: "Aprendo java 8");  
  
System.out.println("Conversion mayúsculas: " + resultadoMayusculas);  
  
}
```

Definimos nuestra
lambda

Lanzamos nuestra
lambda

Imprime nuestro
resultado convertido a
MAYUSCULAS

Interface funcional

Parametros

Retorno

Metodo abs.

PREDICATE<T>

1(T)

BOOLEAN

TEST(T)



```
Predicate<Integer> esPar = numero -> numero % 2 == 0;  
  
boolean esParResultado = esPar.test(t: 4);  
  
System.out.println("¿El número es par? " + esParResultado);
```

Se define lambda y se aplica la logica

Lanzamos nuestra lambda

Retornamos nuestro resultado en este caso
BOOLEAN