



Daniel Ivan Anaya Alvarez

### Interface Functional Activity

The objective of this activity is to demonstrate the knowledge we have gained during the course by explaining and providing examples of functional interfaces available in Java 17. In the next figure we can see the interface functional and the way it is meant to use it.

<i>Interface Funcional</i>	<i>Parámetros</i>	<i>Tipo de Retorno</i>	<i>Método Abstracto</i>
Supplier<T>	0	T	get()
Consumer<T>	1(T)	void	accept(T)
BiConsumer<T,U>	2(T,U)	void	accept(T,U)
Predicate<T>	1(T)	boolean	test(T)
BiPredicate<T,U>	2(T,U)	boolean	test(T,U)
Function<T,R>	1(T)	R	apply(T)
BiFunction<T,U,R>	2(T,U)	R	apply(T,U)
UnaryOperator<T>	1(T)	T	apply(T)
BinaryOperator<T>	2(T,T)	T	apply(T,T)

*Illustration 1 Interface Functional chart*

### Supplier

A Supplier is defined without taking any input and instead provides an output. When called, it returns the value specified as the output, as shown in the following example:

```
package Version2;

import java.util.function.Supplier;

public class Principal2 {
```

```

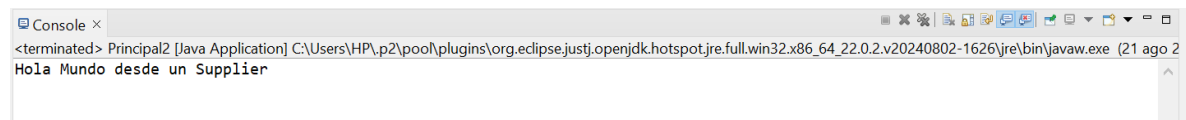
    public static void main(String[] args) {
        // Usando una expresion lambda para crear un Supplier que devuelve un
String
        Supplier<String> supplier = () -> "Hola Mundo desde un Supplier";

        // Llamamos al método get() para obtener el valor
        String mensaje = supplier.get();

        // Imprimimos el valor retornado por el Supplier
        System.out.println(mensaje);
    }
}

```

## Supplier print:



The screenshot shows a console window titled "Console x" with the following text:
   
<terminated> Principal2 [Java Application] C:\Users\HP\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (21 ago 2
   
Hola Mundo desde un Supplier

## Consumer

The Consumer interface accepts an input and performs an operation on it without returning any result. In this exercise, it was used to print a message.

```

package Version2;

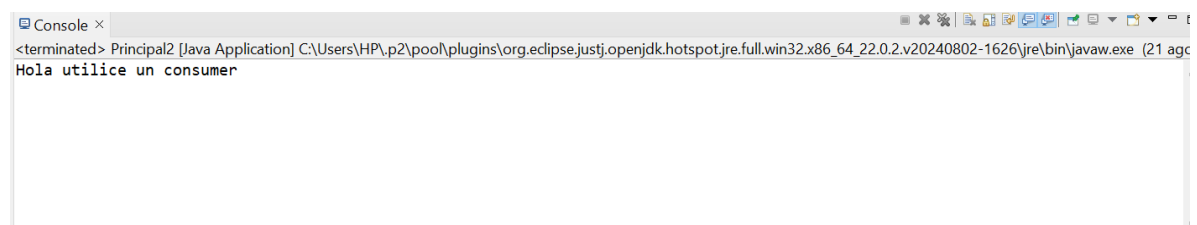
import java.util.function.Consumer;

public class Principal2 {
    public static void main(String[] args) {
        // Generamos la referencia del consumer
        Consumer<String> cons = cor -> System.out.println(cor);

        // Se utiliza un consumer
        cons.accept("Hola utilice un consumer");
    }
}

```

## Consumer Print



The screenshot shows a console window titled "Console x" with the following text:
   
<terminated> Principal2 [Java Application] C:\Users\HP\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (21 ago
   
Hola utilice un consumer

## Predicate

The Predicate interface takes an input and returns a boolean value. In this exercise, it was used to check if a given integer equals 1

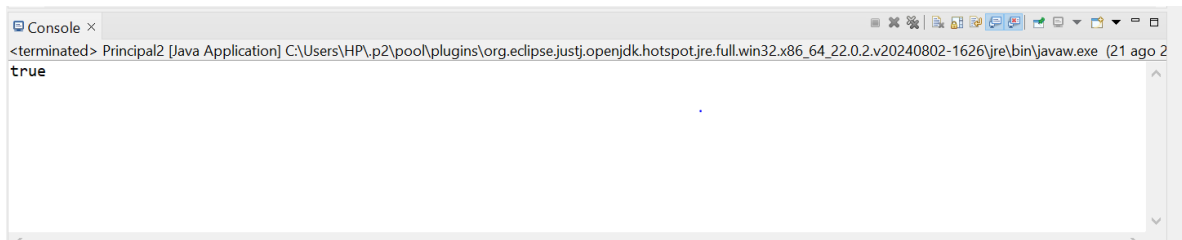
```
package Version2;

import java.util.function.Predicate;

public class Principal2 {
    public static void main(String[] args) {
        //Generamos nuestro predicate le mandamos un integer y nos regresa un
        //booleano
        Predicate<Integer> predicate = x -> (x==1);

        //Checa si se cumple la condición
        boolean x2 = predicate.test(1);
        System.out.println(x2);
    }
}
```

## Predicate Print



## UnaryOperator

The UnaryOperator interface is a specialized Function that accepts one input and returns a result of the same type. In this exercise, it was used to double an integer value.

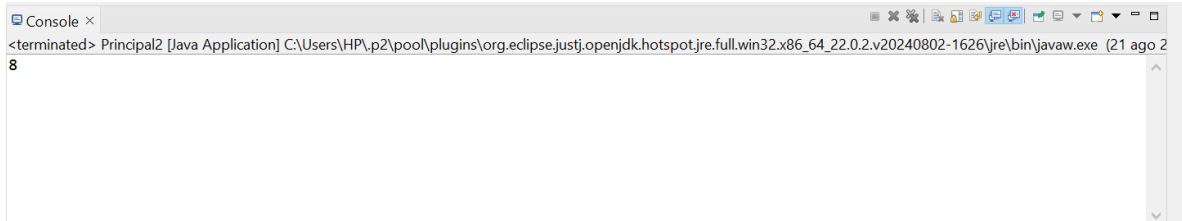
```
package Version2;

import java.util.function.UnaryOperator;

public class Principal2 {
    public static void main(String[] args) {
        //Generamos la referencia
        UnaryOperator<Integer> uo = x -> x*2;
        int uor = uo.apply(4);
        System.out.println(uor);
    }
}
```

```
}  
}
```

## UnaryOperator Print

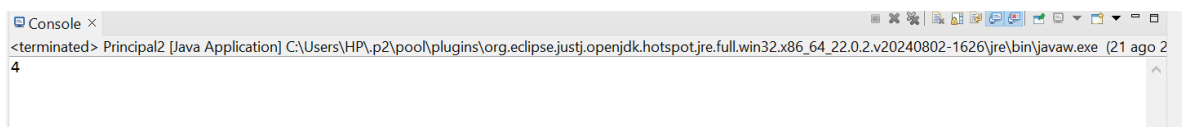


## Function

The Function interface takes an input and returns a result of a different type. In this exercise, it was used to calculate the length of a string.

```
package Version2;  
  
import java.util.function.Function;  
  
public class Principal2 {  
    public static void main(String[] args) {  
        //En function en el generic declaramos que recibe un string y va  
        //devolver un integer  
        Function<String,Integer> fun =x->x.length();  
        //Se pone el string y cuenta la longitud de las palabras  
        int fun1 = fun.apply("Hola");  
        System.out.println(fun1);  
    }  
}
```

## FunctionPrint



## BiConsumer

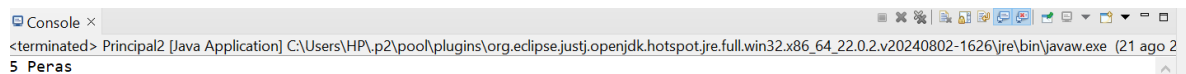
The BiConsumer interface accepts two inputs and performs an operation without returning any result. In this exercise, it was used to print an integer combined with a string.

```
package Version2;
```

```
import java.util.function.BiConsumer;

public class Principal2 {
    public static void main(String[] args) {
        //BiConsumer agarra integer y string y no regresa nada
        BiConsumer<Integer,String> bicons = (intb, strb) ->
        System.out.println(intb + strb);
        //Se ingresa el integer y el string
        bicons.accept(5, " Peras");
    }
}
```

## BiConsumerPrint



Console ×  
 <terminated> Principal2 [Java Application] C:\Users\HP\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (21 ago 2  
 5 Peras

## BinaryOperator

```
package Version2;

import java.util.function.BinaryOperator;

public class Principal2 {
    public static void main(String[] args) {
        //BinaryOperator introducimos dos valores del mismo tipo en este caso int
        //y hace la operación
        BinaryOperator<Integer> Biop = (i1,i2) -> i1+i2*4;
        int Biopr = Biop.apply(4,8);
        System.out.println(Biopr);
    }
}
```

## BinaryOperator Print



Input  
 36  
 ...Program finished with exit code 0  
 Press ENTER to exit console.

## **Conclusion**

In this activity we were able to see how does Interface functional works, according to the results we can say that those functions works alike lambdas, each of these instructions have their own behavior they are versatile, some ask for inputs and you expect an output and some doesn't ask you an input and it returns you something.