RESUMEN FUNCIONES LAMBDA

Sintaxis general:

(parámetros) → { cuerpo - lambda }

Parámetros:

- Si tiene un parámetro NO es obligatorio el ()
- Si no tienen parámetros, () son obligatorios.
- Si tiene más de un parámetro también poner ()

Cuerpo:

- Con una sola línea, las {} NO son obligatorios y no es necesario return.
- Con más de una línea, {} obligatorias y return también.

Definición:

Una función lambda son funciones anónimas y se pueden utilizar donde haya una interface funcional.

Una interface funcional tiene un único método y **opcionalmente** se puede anotar con **@FunctionalInterface**

CLASIFICACIÓN DE LAS EXPRESIONES LAMBDA:

1) Consumidores: (se dividen en consumidoras y biconsumidoras)

Consumidoras:

Aceptan un valor y no devuelven nada: Se pueden utilizar para modificar una propiedad de un objeto, para imprimir, etc.

String m -> System.out.println(m);

Consumer<T>	void accept(T t)

Biconsumidoras:

Reciben dos valores y no devuelven resultados. Uso similar a las anteriores. Modificar datos, imprimir, etc.

(String key, String value) -> System.out.println("Key: %s, value: %s%n", key, value);

BiConsumer<T,U>	void accept(T t, U u)

2) Proveedores:

No tienen parámetros pero devuelven un resultado.

() → return createRandomInteger()

Supplier<T>	T get()

3) Funciones: (pueden ser funciones o bifunciones):

Reciben un argumento y devuelven un valor como resultado y cuyos tipos no tienen porqué ser iguales.

Function<T,R>	R apply(T t)

Order persistedOrder -> persistedOrder.getIdientifier();

BiFunction<T,U,R>	R apply(T t, U u)

(Address address, String name) -> new Person(name, address);

4) Operadores: Unarios y Binarios

UnaryOperator<T>	T apply(T t)
BinaryOperator<T>	T apply(T t1, T t2)

5) Predicados: Unarios y Binarios

Predicate<T>	boolean test(T t)
BiPredicate<T,U>	boolean test(T t, U u)