EXPRESIONES LAMBDA Y REFERENCIA A METODOS

Interfaces funcionales Propias

- Runnable: No tiene parámetros, retorna void, sintaxis: ( ) -> { ... }, para tareas simples sin retorno
- Function<T,R>: Tiene 1 parámetro (T), retorna R, sintaxis: (t) -> { return r; }, para transformar un valor

- BiFunction<T,U,R>: Tiene 2 parámetros (T, U), retorna R, sintaxis: (t, u) -> { return r; }, para combinar dos valores

METODOS PRINCIPALES

- Runnable: usa .run()
- Function: usa .apply(t)
- BiFunction: usa .apply(t, u)

Referencias a métodos

Forma abreviada de escribir lambdas cuando ya existe un método que hace lo que necesitas.

Tipos

- Método estático: Usa un método de una clase (Clase::metodo).
- Método de un objeto: Usa un método de un objeto ya creado (objeto::metodo).
- Método de un objeto arbitrario: Usa un método de cualquier objeto de una clase (Clase::metodoInstancia).
- Constructor: Crea un objeto nuevo (Clase::new).

Que es?

EXPRESIONES LAMBDA

Son una forma corta de escribir funciones y se usan para pasar código como si fuera un valor (por ejemplo, a un metodo).
También se llaman funciones anónimas, porque no tienen nombre ni clase y para usarlas se necesita una interfaz funcional.

Sintaxis

Sin parámetros	() -> sentencia
Con un parámetro	param -> sentencia
Con varios parámetros	(param1, param2) -> sentencia
Con más de una sentencia	(param) -> { sentencia1; sentencia2; }

Interfaces Funcionales

Son interfaces con un solo método abstracto y se usan junto con las expresiones lambda. La anotación @FunctionalInterface no es obligatoria, pero se recomienda usarla.

Cuando si usar y no

Se usan lambdas cuando el código es corto y simple. No se usan cuando el código es largo o complejo, se necesitan varios métodos, o la interfaz tiene más de un método abstracto.