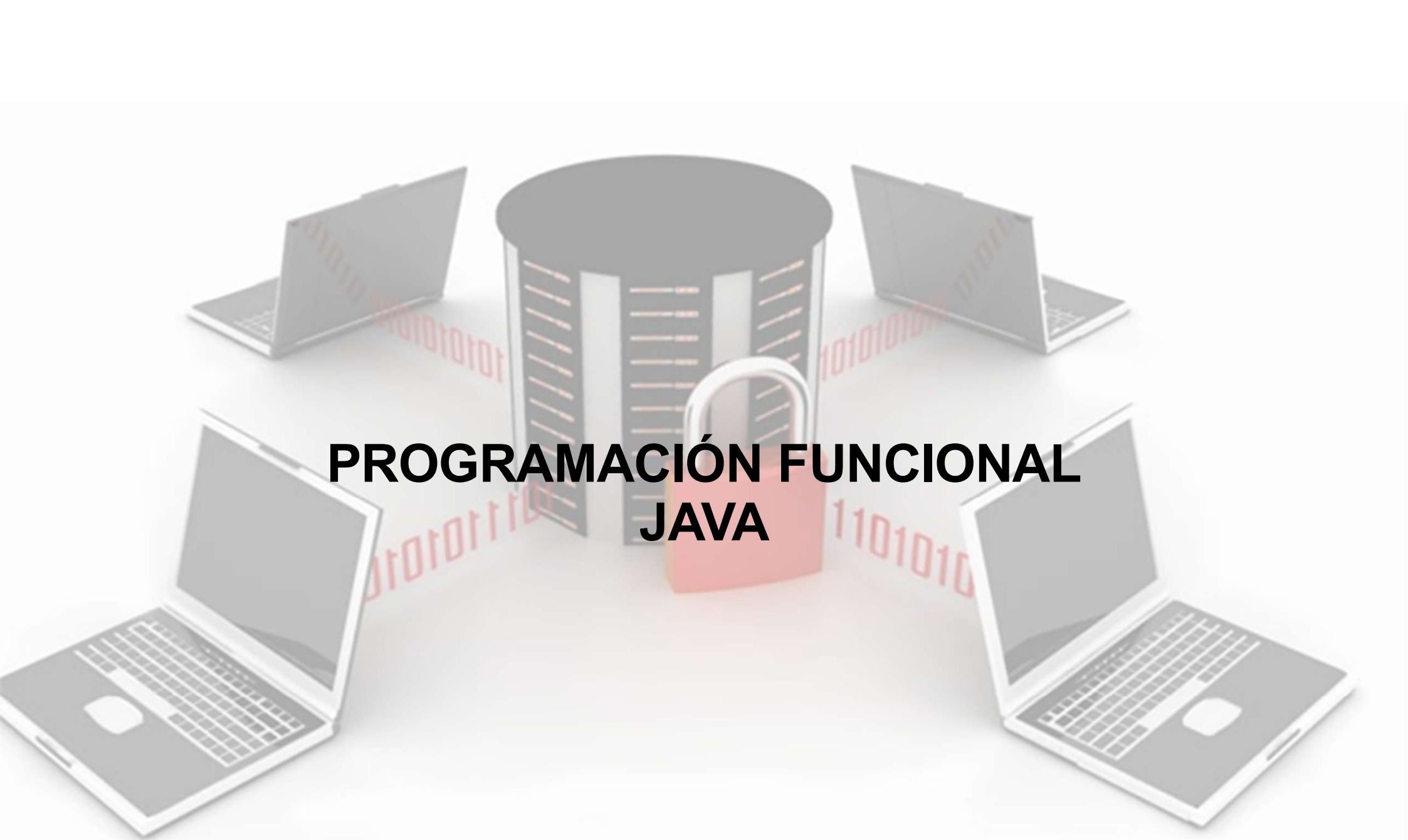

# PROGRAMACIÓN FUNCIONAL JAVA

# INTERFACES FUNCIONALES

- Es una interfaz con un y sólo un método abstracto.
- Puede tener todos los métodos default, static o private que quiera.
- Se recomienda anotarla con @FunctionalInterface

```
@FunctionalInterface
public interface MiInterfazFuncional {

    void procesa(Cliente c);

}
```

# java.util.function

| | | |
|---|---|---|
| Predicados | Predicate<T> | boolean test(T) |
| Representa un predicado, devuelve un booleano a partir del parámetro | | |
| Consumidores | Consumer<T> | void accept(T) |
| Representa una operación que acepta un parámetro y no retorna nada | | |
| Funciones | Function<T, R> | R apply(T) |
| Representa una función, acepta un argumento y produce un resultado | | |
| Proveedores | Supplier<R> | R get() |
| Representa un proveedor de un resultado | | |
| Operadores | UnaryOperator<T> | T apply(T) |
| Representa una función, acepta un argumento y produce un resultado del mismo tipo | | |

# Expresiones lambda

- Una característica única de las interfaces funcionales es que pueden ser instanciadas usando “expresiones lambdas”
- Permiten crear código más conciso y significativo, además de abrir la puerta hacia la programación funcional en Java, en donde las funciones juegan un papel fundamental
- Funciones como entidades de primer nivel
  - Las expresiones lambda permiten utilizar las funciones (métodos) como se utilizan los tipos primitivos o los objetos. Es decir, poder pasar funciones, en tiempo de ejecución como valores de variables, valores de retorno o parámetros de otras funciones.

# Expresiones lambda - sintaxis

- Sintaxis general

  (lista parámetros) -> {cuerpo del lambda}

Lista de parámetros

Si hay un solo parámetro, no son obligatorios los paréntesis

Si no hay parámetro o hay más de uno, son obligatorios

Cuerpo del lambda

Si tiene una sólo línea, no es necesario utilizar las llaves y en el caso que deba devolver algo no es necesario el return

Cuando tiene más de una línea, las llaves son obligatorias y en el caso que deba devolver un valor debe incluir return

# Expresiones lambda - ejemplos

- x -> x + 2
- () -> System.out.println(“mensaje”)
- (lado1, lado2) -> lado1 * lado2
- (int lado1, int lado2) -> {return lado1 * lado2;}

# Referencia a métodos

- Permiten utilizar un método como expresión lambda, simplificando aún más la sintaxis

  **System.out :: println** en lugar de **s -> System.out.println(s)**

- Tipos
  - Referencia a métodos estáticos
  - Referencia a métodos de instancia de un objeto existente
  - Referencia a métodos de instancia de un tipo
  - Referencia a métodos constructores