**PATRON OPTION**

Es un “envoltorio” donde este patrón, optional, sirve para evitar errores de tipo NullPointerException

public final class Optional<T>

Es un contenedor de un objeto, que puede ser nulo, y puede no ser nulo.

Como obtener un Optional<T>, se usan 3 clases de métodos:

Optional.**empty**() //Devuelve un optional vacio

Optional.**of**(Object value)//Devuelve un optional en base al valor pasado por parámetro

Optional.**ofNullable**(Object value) // devuelve un optional con un valor si no es nulo, en otro caso, devuelve *empty();*

Ejemplo:

Optional<String> opt = Optional.**of**(“Juan Soy Optional”);

Entonces con la variable opt podemos acceder a métodos tales como:

**isPresent**() //devuelve true si hay valor

**get**() //devuelve valor si lo hay, sino devuelve NoSuchElementException

**ifPresent**(Consumer cns) // si hay valor, invoca al consumer, expresión lambda

**orElse**(T other) //si hay valor, lo devuelve, sino devuelve other

**orElseGet**(Supplier) // si hay valor, lo devuelve, sino lo toma del supplier

**orElseThrow**(Supplier) // si hay valor, lo devuelve, sino lanza una excepción.

Buenas practicas

<https://refactorizando.com/java-optional-buenas-practicas/>

No asignar null a un optional

No usar optional en constructores

No usar optional con listas

No usar optional como parámetro de un método

Usar mejor orElse() & orElseThrow() que isPresent() & get()

Evitar usar optional como operador ternario

No usar optional al devolver valor de un método