BAAZIZ WISSAL

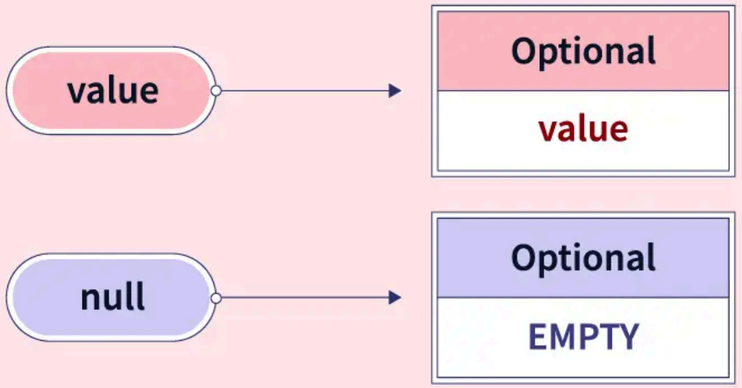
Optional in java

09/09/2024

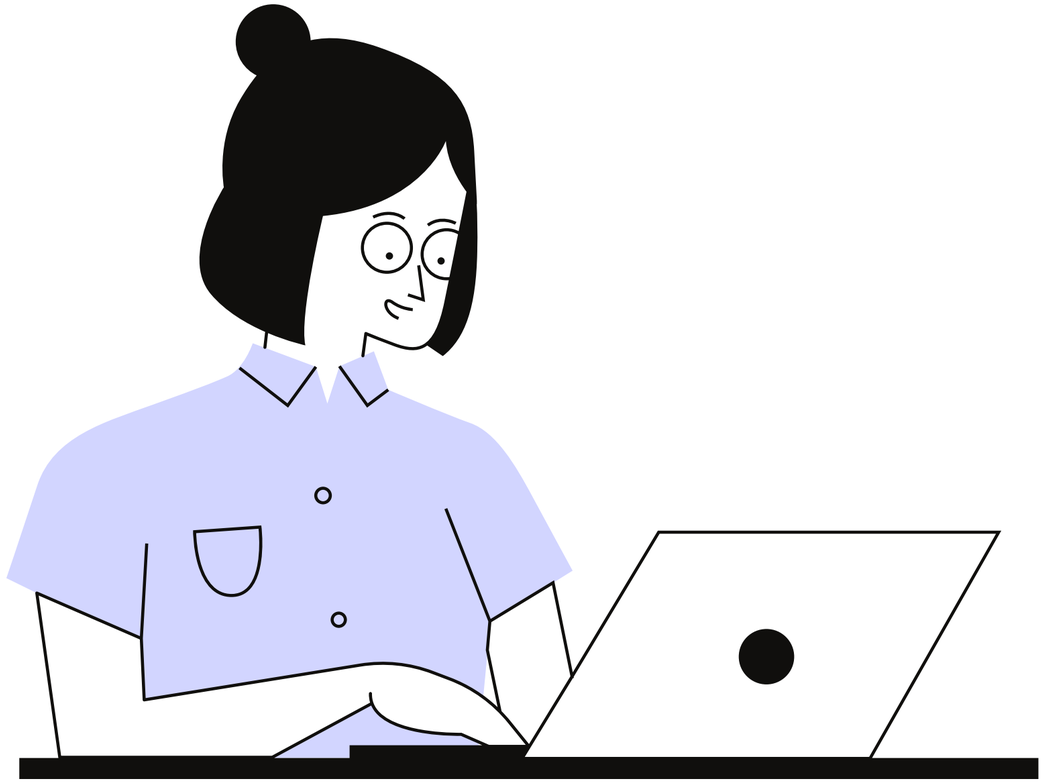
Introduction to Java Optionals

The Optional class was introduced in Java 8 to handle cases where a value may or may not be present. It's designed to prevent NullPointerException by explicitly handling optional (or nullable) values.

← RETOURNER À LA VUE D'ENSEMBLE

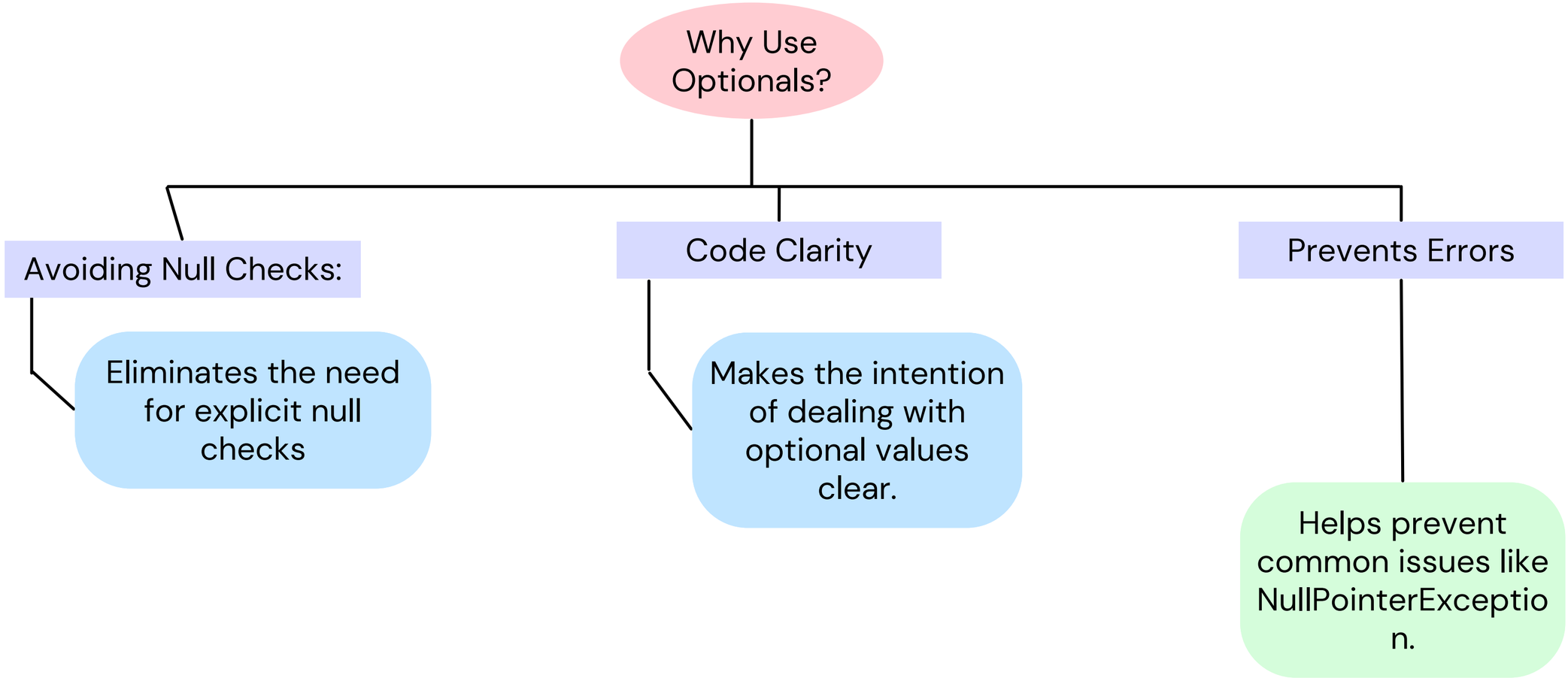

value
Optional
value
null
Optional
EMPTY

Key Concepts



- **Optional**: A container object that may or may not contain a non-null value.
- **Null Safety:** Optional helps avoid null checks and reduces the risk of NullPointerException.
- **Functional Operations:** Provides methods to handle values in a functional style, such as map, flatMap, filter, and ifPresent.

← RETOURNER À LA VUE D'ENSEMBLE



← RETOURNER À LA VUE D'ENSEMBLE

Creating and Using Optionals

CREATING AN OPTIONAL
Optional.of(value): Creates an Optional containing a non-null value.
Optional.ofNullable(value): Creates an Optional that can hold either a value or null.
Optional.empty(): Creates an empty Optional (i.e., no value present).

← RETOURNER À LA VUE D'ENSEMBLE

COMMON METHODS

isPresent(): Checks if a value is present.

ifPresent(Consumer): Executes the given action if a value is present.

orElse(T other): Returns the value if present, otherwise returns the provided default value.

orElseGet(Supplier): Returns the value if present, otherwise invokes a Supplier function to provide a value.

map(Function): Applies a function to the value if present and returns a new Optional.

flatMap(Function): Similar to map, but avoids nested Optionals.

orElseThrow(): Returns the value if present or throws an exception.

Example

```java
import java.util.Optional;

public class OptionalExample {
    public static void main(String[] args) {
        // Creating an Optional containing a value
        Optional<String> optionalValue = Optional.of("Hello, World!");

        // Creating an Optional that can be null
        Optional<String> nullableValue = Optional.ofNullable(null);

        // Example usage of ifPresent and orElse
        optionalValue.ifPresent(value -> System.out.println("Value is present: " + value));

        String defaultValue = nullableValue.orElse("Default Value");
        System.out.println("Nullable value: " + defaultValue); // Output: Default Value

        // Using map to transform the value
        optionalValue.map(String::toUpperCase).ifPresent(System.out::println); // Output: HI
    }
}
```