TELUSKO

Optional Class

👉 What is Optional in Java?

- Optional is a container object introduced in Java 8.
- It is used to represent the presence or absence of a value.
- Its main goal is to avoid the classic runtime error: NullPointerException.

👉 Why was Optional introduced?

- To handle null values in a safe and clean way without writing too many null checks manually.

👉 Key Points of Optional in Stream API

- It is returned by stream terminal operations like:
  - findFirst()
  - findAny()
  - max() and min()

- Optional helps you:
  - Avoid null pointer exceptions
  - Write cleaner and safer code
  - Provide default values using orElse()
  - Chain fallback logic using orElseGet()
  - Throw custom exception using orElseThrow()

👉 Example of the Optional class:

- Suppose we have a list of names

```java
import java.util.Arrays;
import java.util.List;

public class OptionalEx {
    public static void main(String[] args) {

        // Creating a list of names
        List<String> names = Arrays.asList("Navin", "Laxmi", "John", "Kishor");

    }
}
```

TELUSKO

- We want to find the first name that contains "x"

```java
import java.util.Arrays;
import java.util.List;
import java.util.Optional;

public class OptionalEx {
    public static void main(String[] args) {

        // Creating a list of names
        List<String> names = Arrays.asList("Navin", "Laxmi", "John", "Kishor");

        // Using Optional to find the first name containing 'x'
        Optional<String> name = names.stream()
                                .filter(str -> str.contains("x"))
                                .findFirst();

        //Printing the found name
        System.out.println(name.get());
    }
}
```

Output:

```
Laxmi
```

- Suppose no name in the list contains the letter “x”, then stream returns an empty Optional, and calling .get() on an empty Optional causes NoSuchElementException.

```java
import java.util.Arrays;
import java.util.List;
import java.util.Optional;

public class OptionalEx {
    public static void main(String[] args) {

        // Creating a list of names (Updated: "Laxmi" changed to "Lakshmi")
        List<String> names = Arrays.asList("Navin", "Lakshmi", "John", "Kishor");

        // Using Optional to find the first name containing 'x'
        Optional<String> name = names.stream()
                                .filter(str -> str.contains("x"))
                                .findFirst();

        // Printing the found name, if present (May throw NoSuchElementException if not found)
        System.out.println(name.get());
    }
}
```

Output:

```
Exception in thread "main" java.util.NoSuchElementException: No value present
        at java.base/java.util.Optional.get(Optional.java:143)
        at OptionalEx.main(OptionalEx.java:14)
```

- Avoiding NullPointerException using Optional.

```java
import java.util.Arrays;
import java.util.List;

public class OptionalEx {
    public static void main(String[] args) {

        // Creating a list of names
        List<String> names = Arrays.asList("Navin", "Lakshmi", "John", "Kishor");

        // Using Optional to find the first name containing 'x'
        // Optional<String> name = names.stream()
        //                      .filter(str -> str.contains("x"))
        //                      .findFirst();

        // Printing the result using Optional's orElse() to handle absence of match
        // System.out.println(name.orElse("Not Found"));

        // Alternative approach: Directly assigning the result to a String variable
        // Using orElse() to return "Not Found" if no match is found
        String name2 = names.stream()
                            .filter(str -> str.contains("x"))
                            .findFirst()
                            .orElse("Not Found");

        // Printing the result
        System.out.println(name2);
    }
}
```

Output:

```
Not Found
```

👉 Remember

- Optional is a powerful and safe way to handle possibly null results from Stream operations.
- Avoids NullPointerException and makes your code cleaner and safer.
- Always prefer using .orElse() or .orElseGet() instead of .get() directly.