In Java 8, the Optional class was introduced as a container that may or may not contain a non-null value. It's designed to help handle situations where a value might be missing, thus reducing the chances of null pointer exceptions. Optional provides a way to explicitly indicate the possibility of a value being absent, encouraging better handling of such cases.

Here's a basic overview of the Optional class and how you might use it, along with an example:

1. \*\*Creating an Optional instance:\*\*

You can create an Optional instance using static methods like `empty()` and `of()`.

```java

import java.util.Optional;

public class OptionalExample {

public static void main(String[] args) {

Optional<String> emptyOptional = Optional.empty();

Optional<String> nonEmptyOptional = Optional.of("Hello, world!");

}

}

```

2. \*\*Using Optional to handle possible null values:\*\*

Instead of directly storing a null value, you can use Optional to wrap the value and clearly indicate its absence.

```java

import java.util.Optional;

public class OptionalExample {

public static void main(String[] args) {

String nullableValue = null;

Optional<String> optionalValue = Optional.ofNullable(nullableValue);

if (optionalValue.isPresent()) {

System.out.println("Value present: " + optionalValue.get());

} else {

System.out.println("Value is absent.");

}

}

}

```

3. \*\*Using Optional with functional methods:\*\*

Optional provides several methods to work with the contained value in a functional manner, allowing you to avoid null checks.

```java

import java.util.Optional;

public class OptionalExample {

public static void main(String[] args) {

Optional<String> optionalName = Optional.of("Alice");

// Using map to transform the value if present

Optional<String> upperCaseName = optionalName.map(String::toUpperCase);

// Using orElse to provide a default value if absent

String defaultName = optionalName.orElse("Unknown");

System.out.println("Original name: " + optionalName.get());

System.out.println("Uppercase name: " + upperCaseName.get());

System.out.println("Default name: " + defaultName);

}

}

```

Remember that using Optional is generally recommended in cases where null values are a concern and proper handling of absent values is desired. It can improve code readability and help prevent unexpected null pointer exceptions. However, overusing Optional can also lead to less readable code, so use it judiciously.