ROHIT DOSHI

EXCEPTION PROPAGATION

JAVA

EXCEPTION PROPAGATION

In Java, this refers to the process by which an exception that is thrown in one method can be passed up the call stack to higher-level methods that have the ability to handle or propagate the exception further.

This mechanism allows for better error handling and separation of concerns, as exceptions can be caught and dealt with at appropriate levels of the program.

@rohitdoshi9

Rohit Doshi

LET'S UNDERSTAND THIS CONCEPT WITH EXAMPLES

Rohit Doshi

@rohitdoshi9

BASIC EXCEPTION PROPAGATION

```
public class ExceptionPropagationExample {

    public static void main(String[] args) {
        try {
            method1();
        } catch (Exception e) {
            System.out.println("Exception caught in main: " + e.getMessage());
        }
    }

    public static void method1() throws Exception {
        method2();
    }

    public static void method2() throws Exception {
        throw new Exception("Exception in method2");
    }
}
```

Let's see the explanation in the next slide.

BASIC EXCEPTION PROPAGATION

In this example:

- method2() throws an exception.
- method1() calls method2() and doesn't handle the exception itself, but it declares that it throws an exception.
- main() calls method1() and catches the exception. The exception is propagated up the call stack from method2() to method1() and then to main().

Rohit Doshi

CHECKED VS. UNCHECKED EXCEPTION PROPAGATION

```
public class ExceptionPropagationExample {

    public static void main(String[] args) {
        method1();
    }

    public static void method1() {
        try {
            method2();
        } catch (ArithmeticException e) {
            System.out.println("ArithmeticException caught in method1: " + e.getMessage());
        }
    }

    public static void method2() {
        int result = 10 / 0; // ArithmeticException (unchecked exception)
    }
}
```

Let's see the explanation in the next slide.

Rohit Doshi

@rohitdoshi9

CHECKED VS. UNCHECKED EXCEPTION PROPAGATION

In this example :

- method2() throws an unchecked exception (ArithmeticException), which doesn't require explicit declaration or handling.
- method1() calls method2() but only catches the specific exception it expects (ArithmeticException).

CUSTOM EXCEPTION PROPAGATION

```
class CustomException extends Exception {
    public CustomException(String message) {
        super(message);
    }
}

public class ExceptionPropagationExample {

    public static void main(String[] args) {
        try {
            method1();
        } catch (CustomException e) {
            System.out.println("CustomException caught in main: " + e.getMessage());
        }
    }

    public static void method1() throws CustomException {
        method2();
    }

    public static void method2() throws CustomException {
        throw new CustomException("Custom exception in method2");
    }
}
```

Let's see the explanation in the next slide.

Rohit Doshi

CUSTOM EXCEPTION PROPAGATION

In this example :

- A custom exception class CustomException is defined, which is a subclass of Exception.
- Both method2() and method1() declare that they can throw CustomException.
- The exception is propagated up from method2() to method1() and then caught in the main() method.

WAS THIS HELPFUL?

Be sure to save it so you
can come back to it later!

Like ! Comment ! Repost !

It helps someone curios out there!