Handling Exceptions in Java

HANDLING EXCEPTIONS

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim jwhh.com

Overview

The role of exceptions

Working with try/catch blocks

Implementing cleanup with finally

Automating cleanup

Dealing with Errors

Programs will encounter errors

Need an effective mechanism for handling and recovery

Exceptions

Non-intrusive way to signal errors

Allows errors to be handled in a structured manner

Dealing with Errors

Exception handling relies on try/catch blocks

Try block

Contains “normal” code to execute

Runs to completion when no exceptions

Exits block if exception thrown

Catch block

Contains error handling code

Runs only if matching exception is thrown

Receives exception information

Main.java

```java
int i = 12;

int j = 5;

try {

int result = i / (j - 2);

System.out.println(result);

}
```

Main.java

```
int i = 12;
int j = 5;
try {
    int result = i / (j - 2);
    System.out.println(result);
} catch (Exception ex) {
    System.out.println("Error: " + ex.getMessage());
    ex.printStackTrace();
}
doMoreWork();
```

$\overbrace{(j - 2)}^{3}$

Main.java

```
int i = 12;
int j = 2;
                          0
try {
    int result = i / (j - 2);
    System.out.println(result);
} catch (Exception ex) {
    System.out.println("Error: " + ex.getMessage());
    ex.printStackTrace(); // Helpful during app development
}
doMoreWork();
```

Handling Cleanup

Tasks often require cleanup

Close file, database, etc.

May be needed even if exception occurs

Finally block

Can be added at end of try/catch

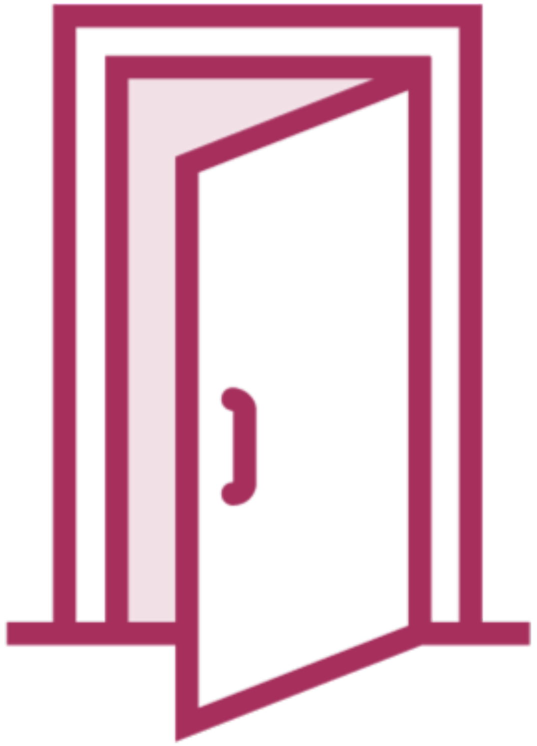
Runs in all cases following try or catch

Automating Cleanup

Manual cleanup can be cumbersome

- Often requires null checks
- Often requires additional exception handling within finally block

Automating Cleanup



AutoCloseable interface

- Indicates automated cleanup support
- Has 1 method: Close

Closeable interface

- Inherits from AutoClosable
- Has 1 method: Close

Automating Cleanup

Try-with-resources automates resource cleanup

Utilizes AutoCloseable

Automatically calls close

Verifies non-null before calling close

Syntax

Similar to traditional try

AutoCloseable resource must be created as part of try statement

Exception handling

Can optionally include catch block(s)

Same catch block(s) handle try body and automatic closing

Summary

Exceptions

- Serve as a signal for errors
- Allow for structured error handling

Handing exceptions

- Use try/catch blocks

Summary

Try block

- Contains “normal” code to execute
- Runs to completion if no exception
- Exits immediately if exception thrown

Catch block

- Contains error handling code
- Runs if matching exception thrown
- Receives exception information

Summary

Finally block

- Allows for manual cleanup
- Runs in all cases following try or catch

Automating cleanup

- Try-with-resources
- Can be used with any type that implements AutoCloseable interface