



LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

Error Handling (Exception handling)

Object-Oriented Programming with Java

Antonius Camara

Types of errors

- ▶ Syntax errors
 - ▶ The IDE (ex: Eclipse) will notify you about those while you are editing the code
- ▶ Runtime errors (Exceptions)
 - ▶ The code syntax is correct and you can compile/run a program
 - ▶ However, during program execution some unexpected error occur. For example:
 - ▶ Arithmetical operation (dividing a number by zero)
 - ▶ Wrong input is processed (the program is expecting an integer, but the user entered a text)
 - ▶ It is not possible to connect to the database
 - ▶ NULL value at some variable
 - ▶ These unexpected errors are called “Exceptions”
- ▶ Logical errors
 - ▶ The syntax is correct, there are no exception errors, but the programming logic is producing results that should not occur
 - ▶ For example, your program is printing:
 - ▶ `Hello 25! Your age is Mary.`

How to deal with these errors

- ▶ Syntax errors
 - ▶ Fix the syntax using the hints provided by the IDE
- ▶ Logical errors
 - ▶ Create test cases where you can verify that the application works as expected
- ▶ Runtime errors (Exceptions)
 - ▶ Programming languages offer “Exception Handling” mechanisms that you will need to code in your own program

“Exception handling”

- ▶ Programming techniques that allows a program to gracefully handle runtime errors
 - ▶ The program will NOT crash when an exception is found
 - ▶ More robust and fault-tolerant programs
- ▶ Programming languages offer exception handling constructs to help you easily implement error-handling operations. In Java:
 - ▶ Exception Classes
 - ▶ Methods that “throw” exceptions
 - ▶ Try/Catch/Finally blocks

Common exceptions in Java

- ▶ **Null pointer** exceptions (`class NullPointerException`)
 - ▶ Your program tries to access an object that does not exist yet
- ▶ **Arithmetic** exceptions (`class ArithmeticException`)
 - ▶ The program tries to perform an illegal arithmetic operation (ex: divide by zero)
- ▶ **Input mismatch** exceptions (`class InputMismatchException`)
 - ▶ The program is processing a value that has a different data type than expected
- ▶ **Index out of bounds** (`class IndexOutOfBoundsException`)
 - ▶ The program tries to access an indexed object using an index that does not exist. Ex: accessing the “fourth” item in an array that contains only three items

Handling exceptions in Java → “Catching” exceptions

```
try {  
    // Your programming logic. Program statements  
    // This is the section where exceptions can occur  
}  
catch (AnExceptionType exception) {  
    // Statements to handle the exception type  
}  
catch (AnotherExceptionType exception) {  
    // Statements to handle the exception type  
}  
catch (...) {  
    // Statements to handle the exception type  
}  
finally {  
    // This section is used for statements you want to execute  
    // regardless if the try section was successful or if an  
    // exception has happened. The finally statements will always  
    // be executed. This section is OPTIONAL  
}
```

Handling exceptions in Java → “Catching” exceptions

```
try {  
    // Your programming logic. Program statements  
    // This is the section where exceptions can occur  
}  
catch (Exception exception) {  
    // You can also have a Catch block that processes all types  
    // of exceptions (Exception class).  
    // This way you don't need to process different exception types  
    // separately  
}  
finally {  
    // Finally statements. Optional  
}
```

The Java API contains information of what exceptions a method may “throw”

Ex: Class Scanner, method `nextInt()`

[https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html#nextInt\(int\)](https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html#nextInt(int))

nextInt

```
public int nextInt()
```

Scans the next token of the input as an `int`.

An invocation of this method of the form `nextInt()` behaves in exactly the same way as the invocation `nextInt(radix)`, where `radix` is the default radix of this scanner.

Returns:

the `int` scanned from the input

Throws:

`InputMismatchException` - if the next token does not match the *Integer* regular expression, or is out of range

`NoSuchElementException` - if input is exhausted

`IllegalStateException` - if this scanner is closed

Here you can see what exceptions this method may throw. It is up to you to decide if you want to implement a try/catch block when calling this method

Some methods explicitly throw exceptions

Ex: Class DriverManager, method getConnection()

[https://docs.oracle.com/javase/7/docs/api/java/sql/DriverManager.html#getConnection\(java.lang.String,%20java.util.Properties\)](https://docs.oracle.com/javase/7/docs/api/java/sql/DriverManager.html#getConnection(java.lang.String,%20java.util.Properties))

getConnection

```
public static Connection getConnection(String url,  
                                     String user,  
                                     String password)
```

throws SQLException

Attempts to establish a connection to the given database URL. The DriverManager attempts to select an appropriate driver from the set of registered JDBC drivers.

Parameters:

`url` - a database url of the form `jdbc:subprotocol:subname`

`user` - the database user on whose behalf the connection is being made

`password` - the user's password

Returns:

a connection to the URL

Throws:

`SQLException` - if a database access error occurs

In these cases, you will HAVE to code a Try/Catch block when calling the method, otherwise the IDE (Eclipse) will display a syntax error

Hands on...

- ▶ Test and Inspect the program `errorHandling/ErrorHandling.java` (download from github) to see how the program crashes when different exceptions occur. Check the crash error messages displayed by Java
- ▶ Test and Inspect the program `errorHandling/ErrorHandlingGraceful.java` to see how the exceptions were gracefully handled by the program
- ▶ Start to plan how to incorporate exception handling in your own programs (check the evaluation criteria)