Exceptions

Module 20

Exceptions

- An Exception is thrown if an error occures that can’t be handled
- If the Exception is not caught, the program will crash
- Exceptions can be thrown from the standard Java library
- You can also define and throw exceptions yourself

Exceptions

Example of an exception

- ```
  int test = Integer.parseInt("not an int");
  ```

- Trying to parse a String to an int will cause a NumberFormatException like this:

```
Run  Main
/Library/Java/JavaVirtualMachines/jdk1.8.0_112.jdk/Contents/Home/bin/java ...
objc[71635]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachine
Exception in thread "main" java.lang.NumberFormatException: For input string: "not an int"
    at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    at java.lang.Integer.parseInt(Integer.java:580)
    at java.lang.Integer.parseInt(Integer.java:615)
    at com.company.Main.main(Main.java:7)

Process finished with exit code 1
```
```

Catch an exception

```
try {
    int test = Integer.parseInt("not an int");
} catch (Exception e) {
    e.printStackTrace();
}
```

- This time the exception is handled by catching it and printing out the stack trace from the exception
- The program will continue to execute after this statement

Finally

```
try {
    int test = Integer.parseInt("not an int");
} catch (Exception e) {
    e.printStackTrace();
} finally {
    System.out.println("this will always run");
}
```

- A finally block is always run, regardless of wheather there was an exception thrown or not

Catch different types of exceptions

```java
try {
    int test = Integer.parseInt("not an int");
}catch (NumberFormatException e) {
    System.out.println("NumberFormatException");
}catch (Exception e) {
    e.printStackTrace();
}
```

- Different types of exceptions can be handled separately

Throw an exception

```
if (number == null) {
    throw new Exception("number cannot be null");
}
```

- Throw your own exceptions when an error occurs
- Provides an opportunity to catch them somewhere else and handle the exception there

Checked exceptions

- Checked exceptions must be handled
- Checked exceptions are checked by the compiler, it will only compile the code if the exception is handled
- Unchecked exceptions are not checked by the compiler, also called Runtime exceptions

Creating your own exceptions

- An exception must be a subclass of the Throwable class
- A checked exception must be a subclass of the Exception class
- An exception is just like any other class and can have any variables, methods and constructors

```java
public class MyOwnException extends Exception { // example of Exception
    public MyOwnException(String message) {
        super(message);
    }
}

throw new MyOwnException("this is an error!"); // throwing the exception
```

AW Academy