Exceptions

Overview

What to catch when try catch is empty

```
    void catchError() {
      try {

      }catch (Error e) {}
      }


    void catchException() {
      try {

      }catch (Exception e) {}
      }


    void catchThrowable() {
      try {

      }catch (Throwable e) {}
      }


    void catchRTE() {
      try {

      }catch (RuntimeException e) {}
      }


    //Exception 'java.io.IOException' is never thrown in the correspo
    void catchCustomCheckedException() {
      try {

      }catch (IOException e) {}  //DOES NOT COMPILE!
    }
```

CatchExceptions

Throws

A method can declare an exception even if it's not thrown within the method itself.

```
void throwsRuntimeException() throws RuntimeException {
  throwingNothing();
}
void throwingNothing() {}
```

throws Exceptions

Error Classes

Error	Description
ExceptionInInitializerError	Thrown when static initializer throws exception and doesn't handle it
StackOverflowError	Thrown when method calls itself too many times (called infinite recursion because method typically calls itself without end)
NoClassDefFoundError	Thrown when class that code uses is available at compile time but not runtime

ExceptionInInitializerError

```
static int[] array = new int[0];
static {
array[0]= 20;
}
```

ExceptionInInitializerError

Some Runtime Exception

```
public class NumberFormatException extends IllegalArgumentException
```

try with resources

Variables declared in the try block should be either **final** or **effectively final**.

```
var bf = Files.newBufferedWriter(path);
try (bf) { // DOES NOT COMPILE
    bf.append("Welcome to the zoo!");
 }
bf = null;  //here I re-assign the variable used in the try block
```

Closable & AutoClosable

```
package java.io;
public interface Closeable extends AutoCloseable {
void close() throws IOException;
}
```

```
package java.lang;
public interface AutoCloseable {
  void close() throws Exception;
}
```

Note that `AutoCloseable` is in the `java.lang` package, import it's not needed.

`AutoCloseable` is a more general-purpose interface and can be used for any resource that needs to be closed, not necessarily related to I/O.

-g:vars

NullPointerException with -g:vars