# Exceptions

* Are used for handling error / problem situations in programs.
* Java has readymade classes for handling these situations.
* Java has three kind of exceptions:
  1. Compile time (checked) exceptions. The compiler “knows” of these happening, so it generates an error at compiling time. You have to deal with it, or the program won’t compile.
  2. Runtime (unchecked) exceptions. You don’t have to deal with these because the compiler ignores them. However, you should not code in such a way, that these happens. And you can deal with these exceptions too, but usually it is unnecessary.
  3. Errors. Cannot be handled in any way. Signals for some serious problem, f.ex out of memory. They cannot be handled because the whole system can crash.
* The head class of all exceptions is java.lang.Throwable. It is extended by Error (no need to handle (almost) ever) and Exception. The Exception class includes the RuntimeException class and all extenders of that class need not to be handled. They can be handled if so wished, but I recommend checking the possible errors with code.
* You can handle the exceptions in two ways: 1) handle it immediately where it happens or 2) make the method throw an exception. The second option means that you have to handle the exception in the caller (or make the caller throw an exception, and so on).
* The basic form of an exception handler:

try {

Trying to do something that raises an exception.

The code in this block is run until the exception happens!

If no exception happens all the code is run.

}

catch ( SomeExceptionClass nameOfVariable ) {

Handle the exception the best you can.

}

more catch blocks

...

finally {

Run regardless of the exception happening.

Used for cleaning up, if needed.

}

* Instructions:
  1. Do not handle the RuntimeExceptions with try-catch-finally, because exception handling is slow. Code the checks.
  2. Try to minimize the number of try blocks.
  3. Do not leave the exception unhandled.
* Example: ExceptionTest