Exceptions

TODO(This is a stub)

An *exception* type declaration is any type declaration that meets the following criteria:

- It is a [class or object declaration][Classifier declaration];
- It has `kotlin.Throwable` as supertype;
- It has no type parameters.

Any object of an exception type may be *thrown* or *catched*.

Catching exceptions

A [`try`-expression][Try-expression] becomes *active* once the execution of the program enters it and stops being active once the execution of the program leaves it. If there are several active `try`-expressions, the one that became active last is *currently active*.

If an exception is thrown while a try-expression is currently active and this try-expression has any `catch`-blocks, those `catch`-blocks are checked for applicability for this exception. A `catch`-block is applicable for an exception object if the runtime type of this expression object is a subtype of the bound exception parameter of this `catch`-block. Note that this is subject to Kotlin [runtime type information][Runtime type information] limitations and may be dependent on the platform implementation of runtime type information, as well as the implementation of exception classes.

If a `catch`-block is applicable for the exception thrown, the code inside the block is evaluated and the value of the block is returned as the value of a `try`-expression. If this `try`-expression contains a `finally`-block, the body of this block is evaluated after the body of the selected `catch` block. The `try`-expression itself is not considered active inside its own `catch` and `finally` blocks. If this results in throwing other exceptions (including the one caught by the `catch`-block), they are propagated as normal.

If none of the `catch`-blocks of the currently active `try`-expression are applicable for the exception, the `finally` block (if any) is still evaluated and the exception is propagated, meaning that the next active `try`-expression becomes currently active and is checked for applicability.

If there is not a single active `try`-block, the execution of the program finishes, signaling that the exception has reached top level.

Throwing exceptions

Throwing an exception object is performed using `throw`-expression. A valid throw expression `throw e` requires that:

- `e` is a value of [a runtime-available type][Runtime-available types];
- `e` is a value of an exception type (see above).

Throwing an exception results in checking active `try`-blocks as described above.

> Note: Kotlin does not specify whether throwing exceptions involves construction of a program stack trace and how the actual exception handling is performed internally. This is a platform-dependent mechanism.

TODO: control flow?

TODO: concurrency?

TODO: write it better