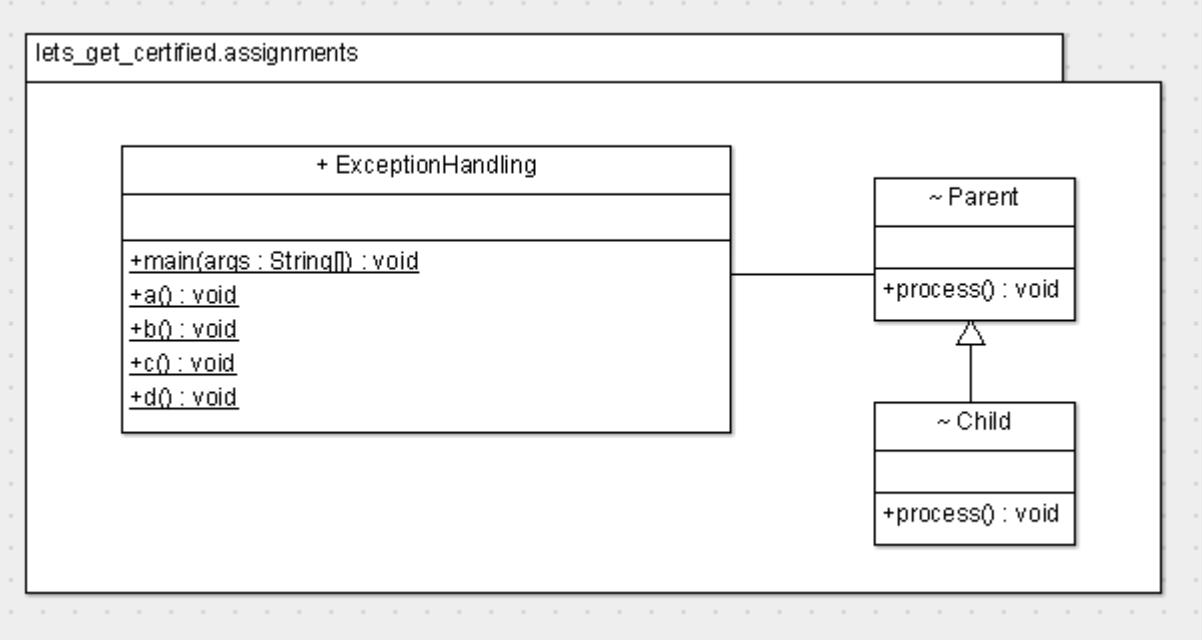
Exceptions



1. To demonstrate that "finally" **<u>is always executed</u>** regardless of whether or not an exception occurs in the *try* block.
    a. Code a method named *a()* with a return type of *String*.
        i. In the method, code an empty *try* block (does not throw any exception).
        ii. Code an exception handler for *Exception* that outputs a tracer message "Exception occurred" and returns "exception".
        iii. Code a *finally* block that outputs a tracer message "finally section…" and returns "finally".
    b. Call *a()* from *main()* and output the string returned.
    c. Run the program and observe the output.

2. To demonstrate that "finally" is executed even when you "return" from *try* block.
    a. Code a method named *b*; return type *String*.
        i. In the method, code a *try* block that returns "ok".
        ii. Code an exception handler for *Exception* that outputs a tracer message "Exception occurred" and returns "exception"
        iii. Code a *finally* block that outputs "finally section…" (**note: no *return* statement in *finally* this time**).
    b. Call *a* from *main()* and output the string returned.
    c. Run the program and observe the output.
    d. Now, revisit the *finally* block in *b()* and insert *return "finally";* Re-run the program – notice how the return from *finally* REPLACES the return from the *try* block.

3. To demonstrate that "finally" is executed even when you "return" from exception handler.
    a. Code a method named *c()*; return type *String*.
        i. In the method, code a *try* block that simply throws an *Exception* (only for demonstration purposes – you should never throw an *Exception* when you don't actually have an error).
        ii. Code an exception handler for *Exception* that outputs a tracer message "Exception occurred" and returns "exception".
        iii. Code a *finally* block that outputs "finally section…" (**note: no *return* statement in *finally* this time**).
    b. Call c from *main()* and output the string returned.
    c. Run the program and observe the output.
    d. Now, revisit the *finally* block in *c()* and insert *return "finally";* Re-run the program – notice how the return from *finally* REPLACES the return from the *catch* block.

4. To demonstrate that the (checked) exceptions you are trying to *catch* must be thrown in the *try* block. This step is about compilation and not runtime.
    a. Code a method named *d()*; return type *String*.
        i. In the method, code a *try* block that simply returns "ok". However, *try* to *catch* the following (checked) exceptions: *IOException*, *NoSuchMethodException* and *ClassNotFoundException*.
        ii. Will they compile? Why or why not?
        iii. Catch *RuntimeException* – will it compile? Why or why not?
        iv. Catch *Exception* – will it compile? Why or why not?

5. This example is all about compilation and not runtime.
    a. Code a class *Parent* that has a method *process*() that throws an *IOException.*
    b. Code a class *Child* that inherits from *Parent*. The class *Child* overrides the *process*() method inherited from *Parent* and it throws *IOException* and *NoSuchMethodException* (both are **checked** exceptions). Do you get an error? Why or why not? If you get an error, what are your options to fix this error?
    c. In *main*() create a new reference called '*p*' of type *Parent* that refers to an object of type *Child*. Invoke *p.process*(). Note that, while the method executed will be in class *Child* (via polymorphism), the signature is determined at compile-time using the type of the reference executing the method i.e. *Parent*. Thus, the exceptions to catch are determined by the (static) reference type i.e. *Parent*.