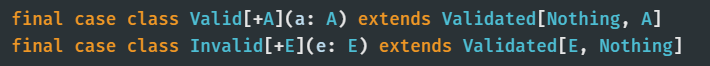
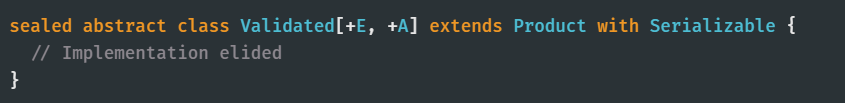
**Validated:**

A way to handle errors, instead of throwing exceptions (which don’t carry as much information as we might want).

It also stores all / multiple errors, unlike the fail-fast method of Either (which returns only the 1st error encountered)

Validated contains 2 types (similar to Either):  
- Errors on the left side  
- Successful computations on the right side

The signature and projections of Validated (a Validated type can be a Valid or Invalid):



**…**

**Validated vs Either**

Both are used for error-handling, but:  
- Either is ‘fails-fast’ / ‘short-circuiting’, so only returns the 1st error that it encounters & doesn’t carry on  
 with the rest of the system / validating.  
- Validated is error-accumulating, so use when want to retrieve all errors in the system.

They also have different data structures:  
- Either is a Monad, so has the flatMap method which can be used in for-comprehensions (used for  
 sequential validation).  
- Validated can't have a valid Monad instance, so can’t be used in for-comprehensions

(Note, Validated can be used in sequential validation by using its *andThen* & *withEither* methods

**How Invalid stores errors it has accumulated:**

* It stores errors in a NonEmptyChain or NonEmptyList.
* As the type is Invalid, it already contains at least 1 error, so we can use the *NonEmpty-* containers for simplicity.