Imagine 4 types:

* Nibble
* Byte
* Integer
* Array

Imagine types are parameterizable, such as Array<T, *optional* Integer *size*>:

* Array<Byte>
* Array<Nibble, 2>

Imagine a simple interface for Array<T, S>:

* T get(Integer n)
* void set(Integer n, T value)
* *readonly* property Integer size
* Array<T, S=*count*> slice(Integer offset, Integer count)

Now extend the Array interface for specific types:

Array<Byte> {

Array<Nibble> as<Array<Nibble>>()

}

Array<Nibble> {

Array<Byte> as<Array<Byte>>()

}

Now extend it for specific types *and sizes*:

Array<Nibble, 2> {

Byte as<Byte>()

}