
#GEEKSTONE
CONFERENCE 2.0

Speak the Language
The Domain Language!

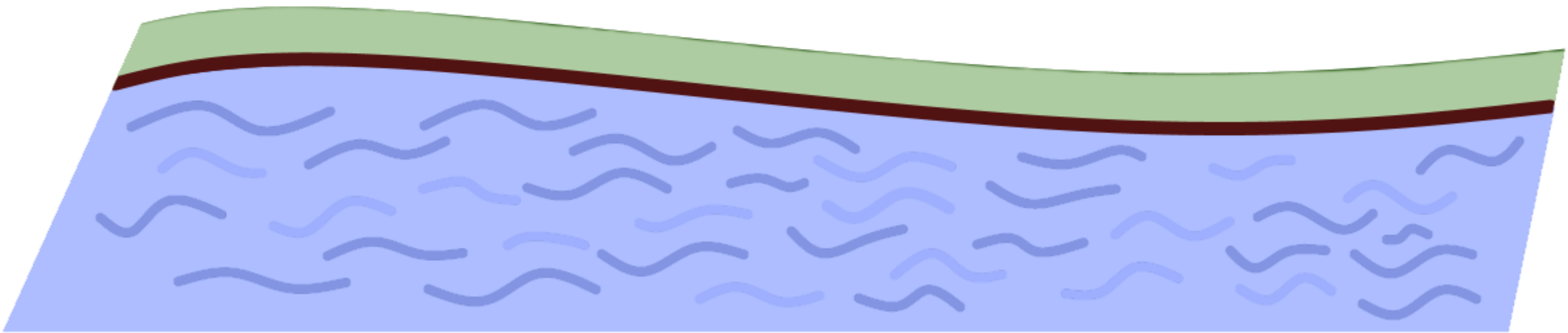
Author at Pluralsight

Publishing tutorials at YouTube

https://youtube.com/@zoran-horvat

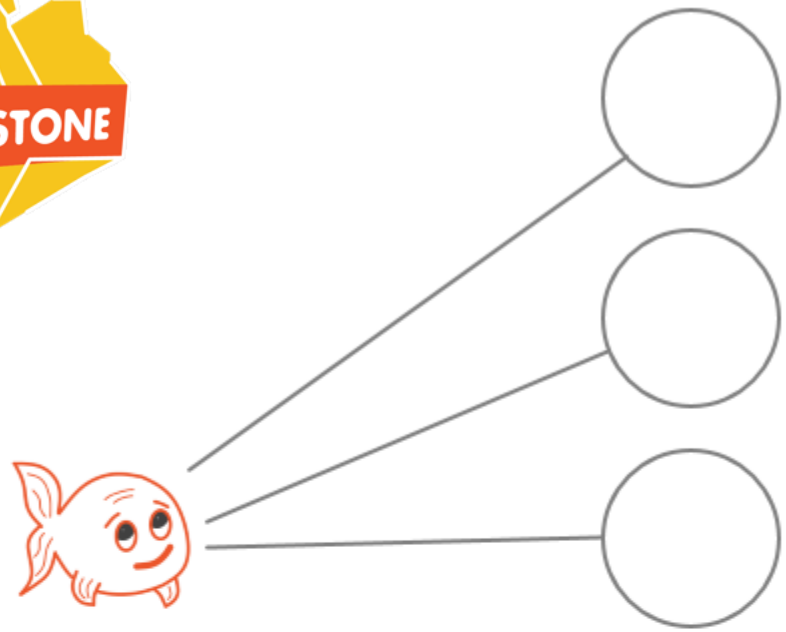
#GEEKSTONE

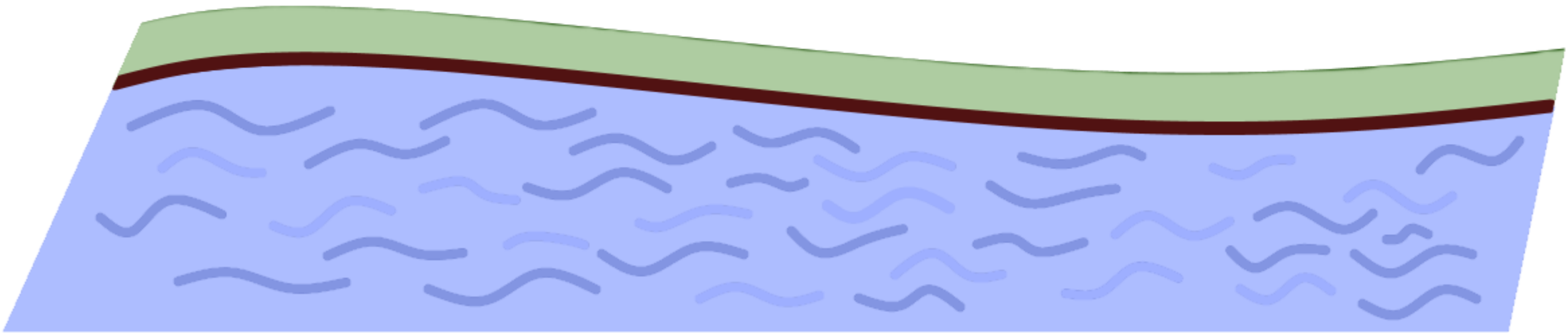
2022



#GEEKSTONE

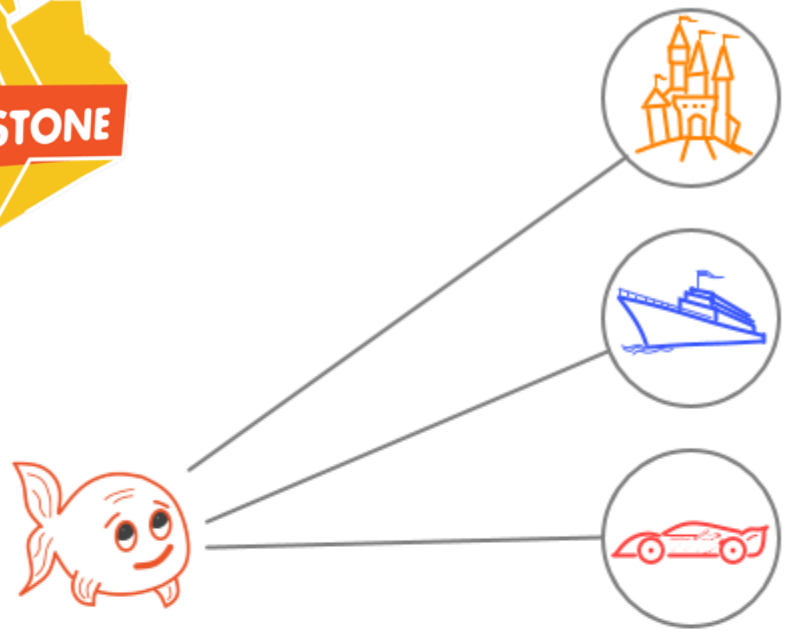
2022

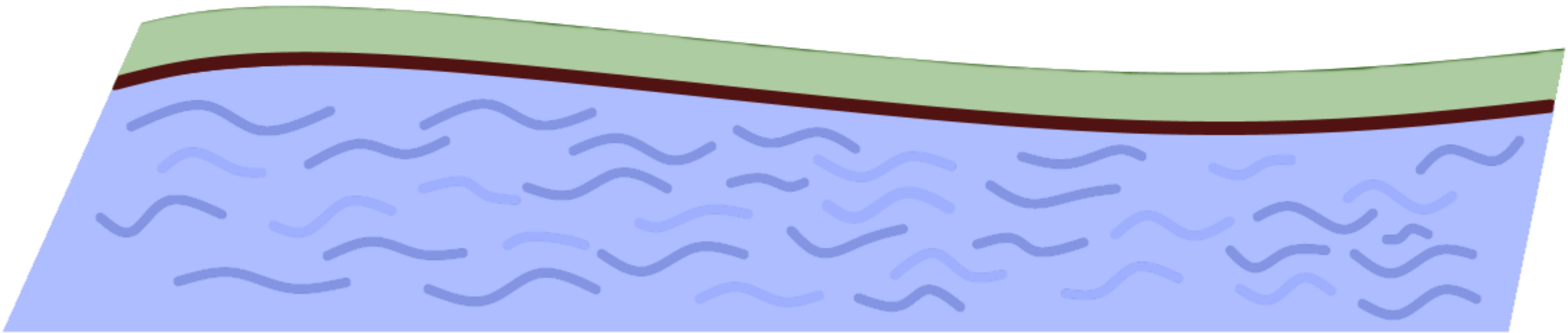




#GEEKSTONE

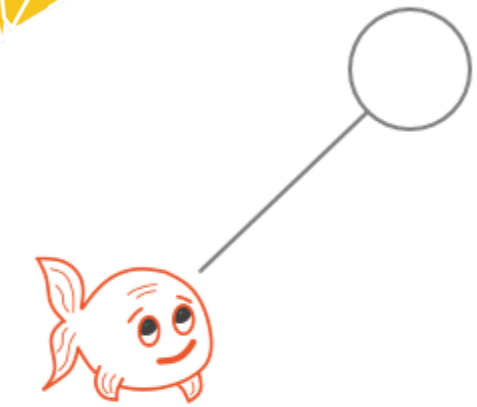
2022





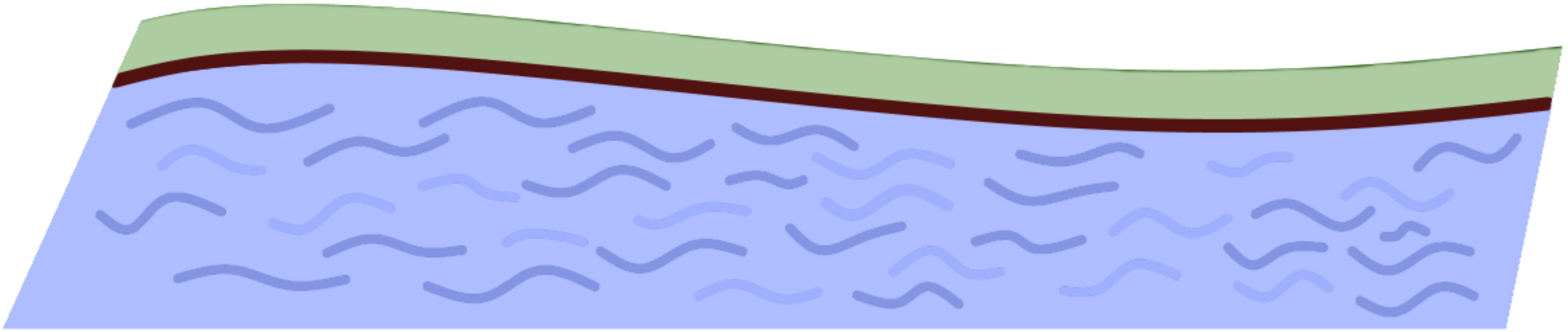
#GEEKSTONE

2022



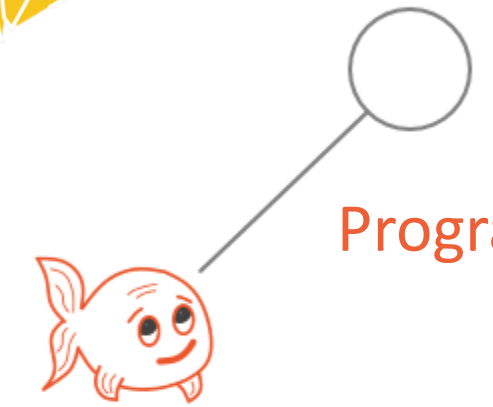
Fish: Make **one** wish

☐ I AGREE WITH THE TERMS AND CONDITIONS



#GEEKSTONE

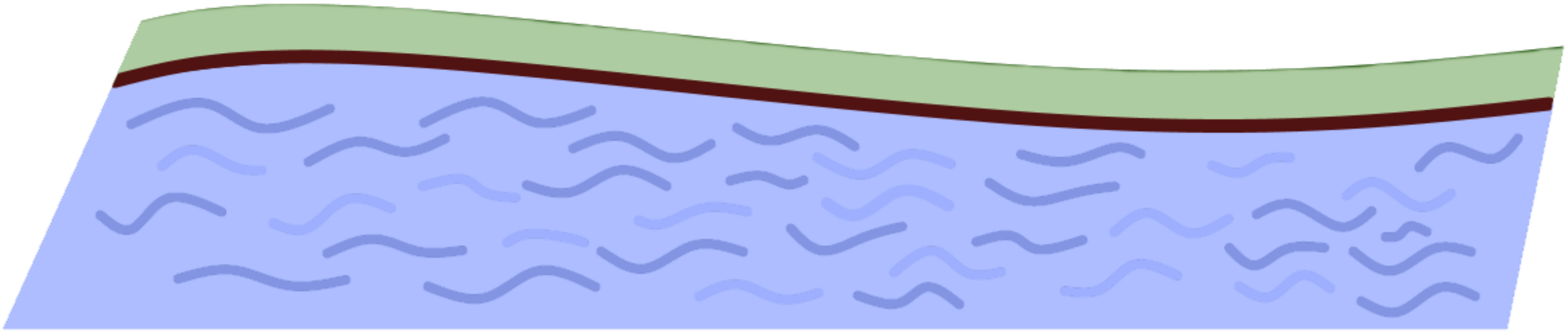
2022



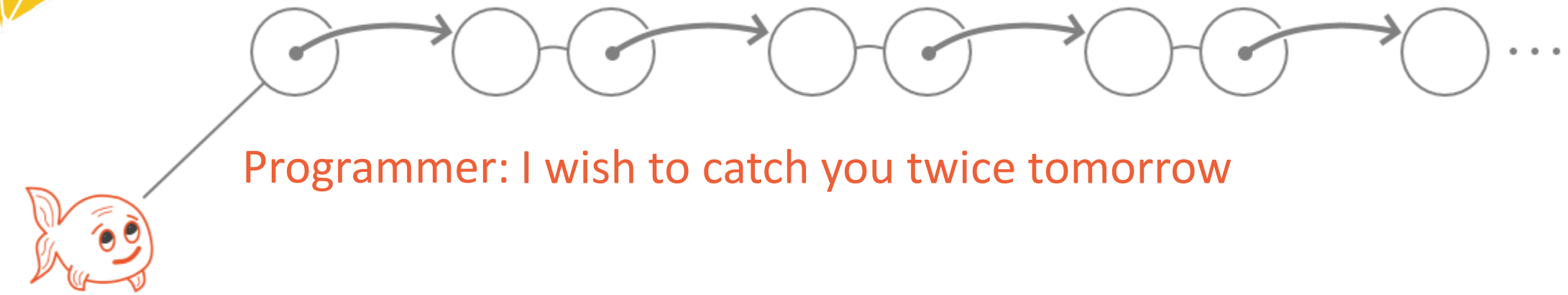
Fish: Make **one** wish

☑ I AGREE WITH THE TERMS AND CONDITIONS

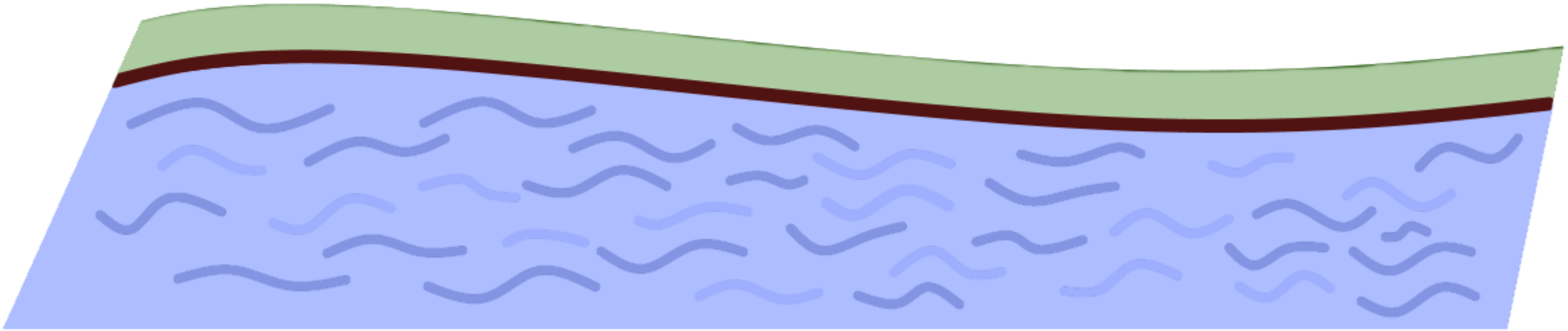
Programmer: I wish to catch you twice tomorrow

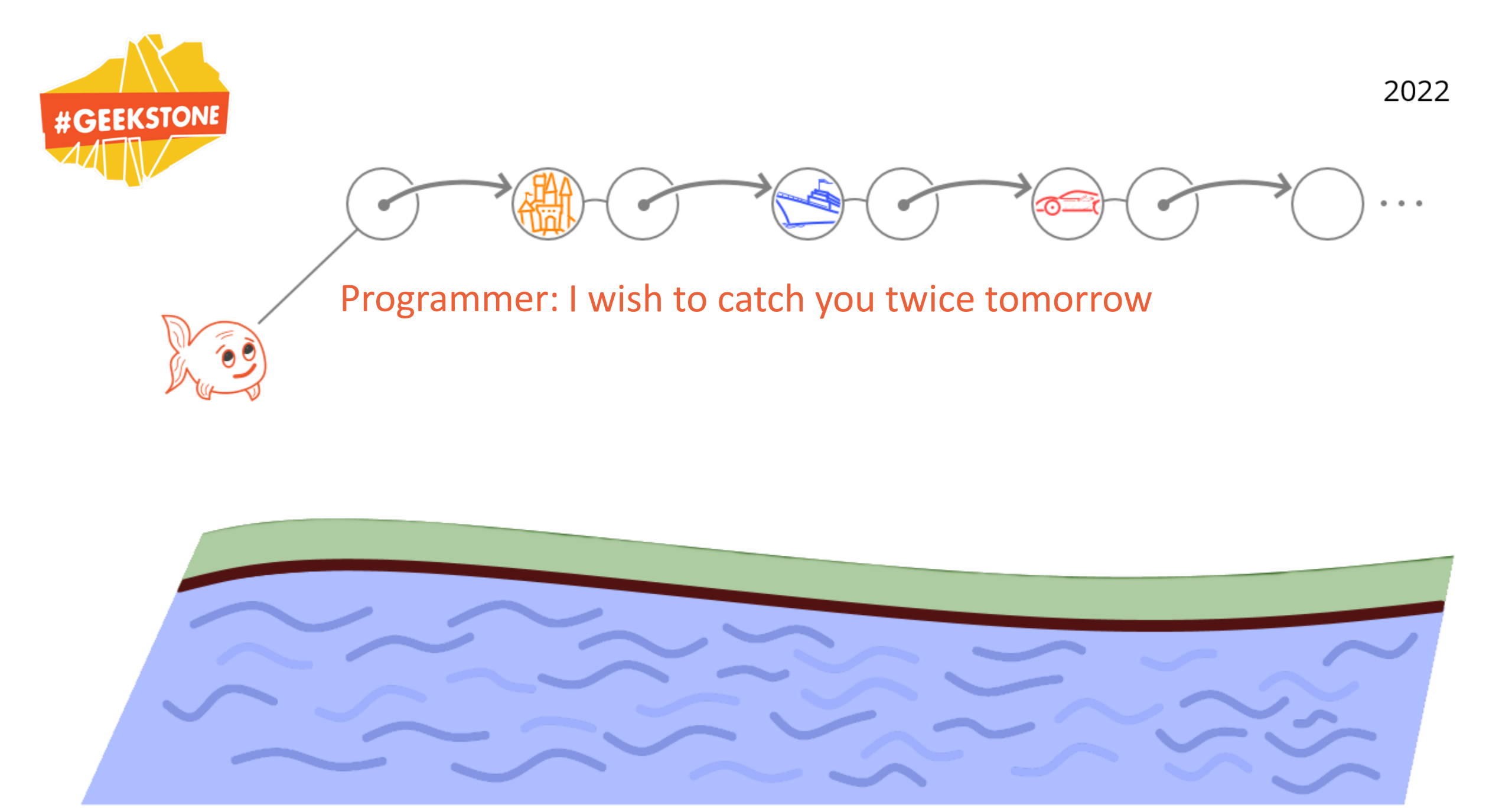


2022



Programmer: I wish to catch you twice tomorrow

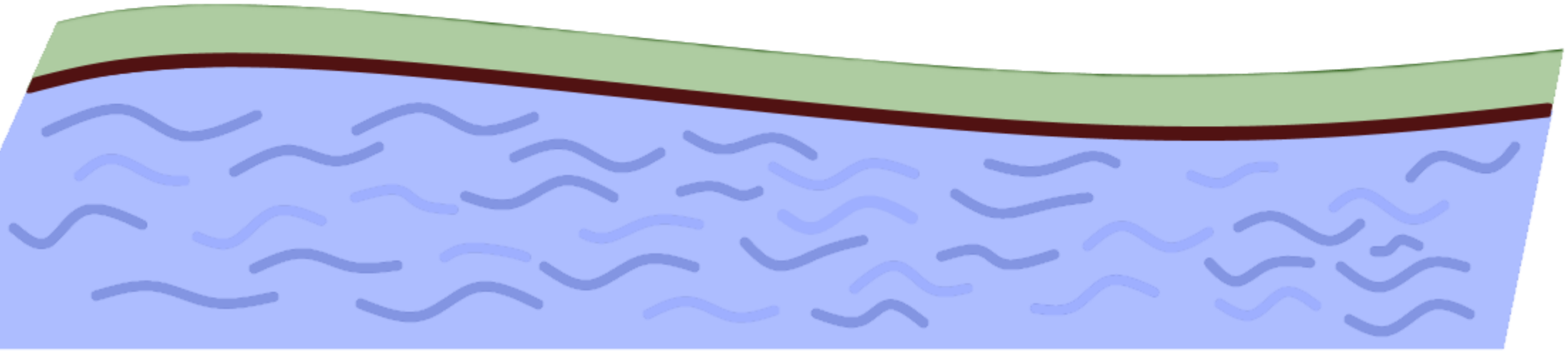



#GEEKSTONE
2022
Programmer: I wish to catch you twice tomorrow

A LISP Fish

2022

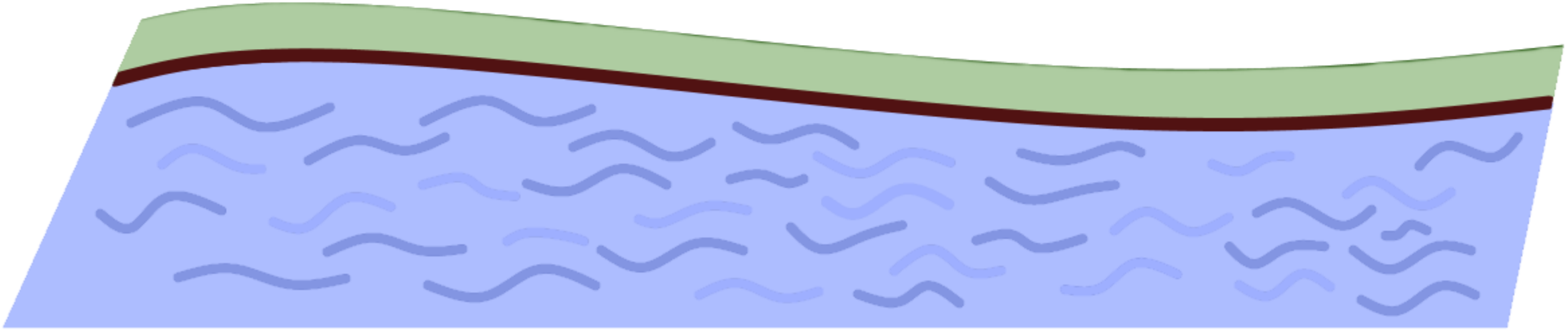
Programmer: I wish to catch you twice tomorrow

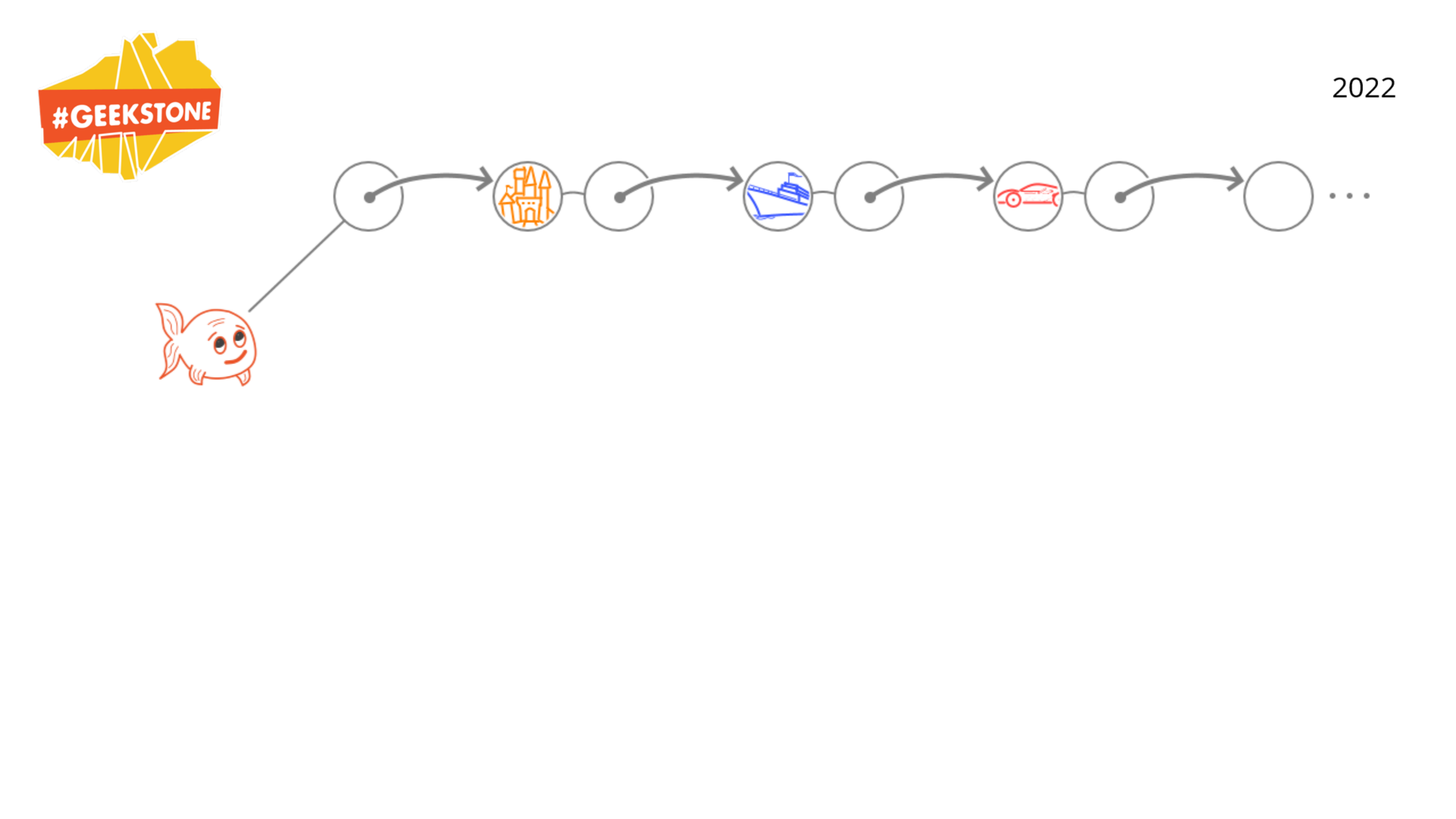


2022

Programmer: I wish to catch you twice tomorrow

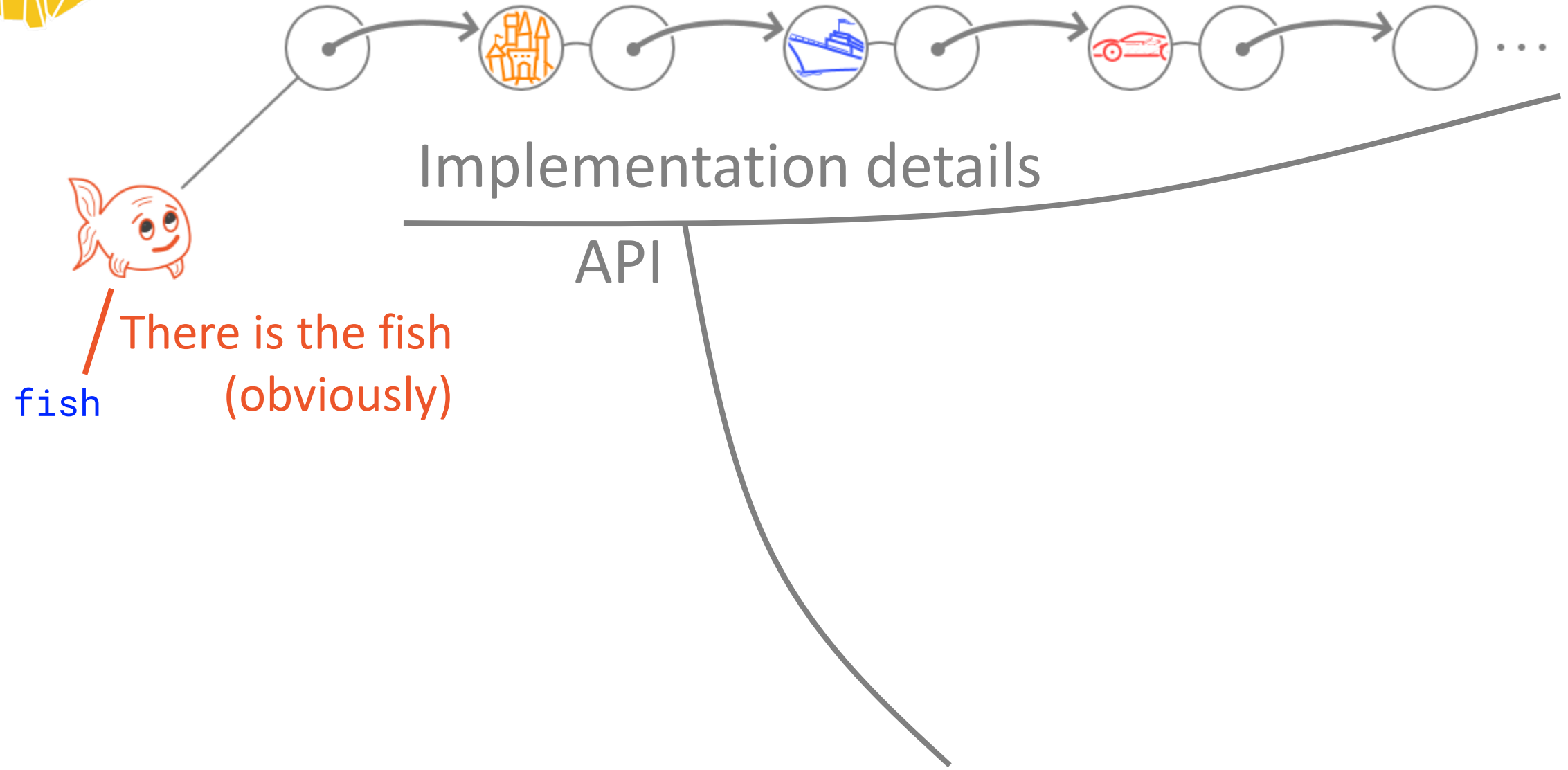
Business guy: Did I just see you release the critter!? @#?&!%




#GEEKSTONE
2022

#GEEKSTONE

2022



#GEEKSTONE

2022

Implementation details

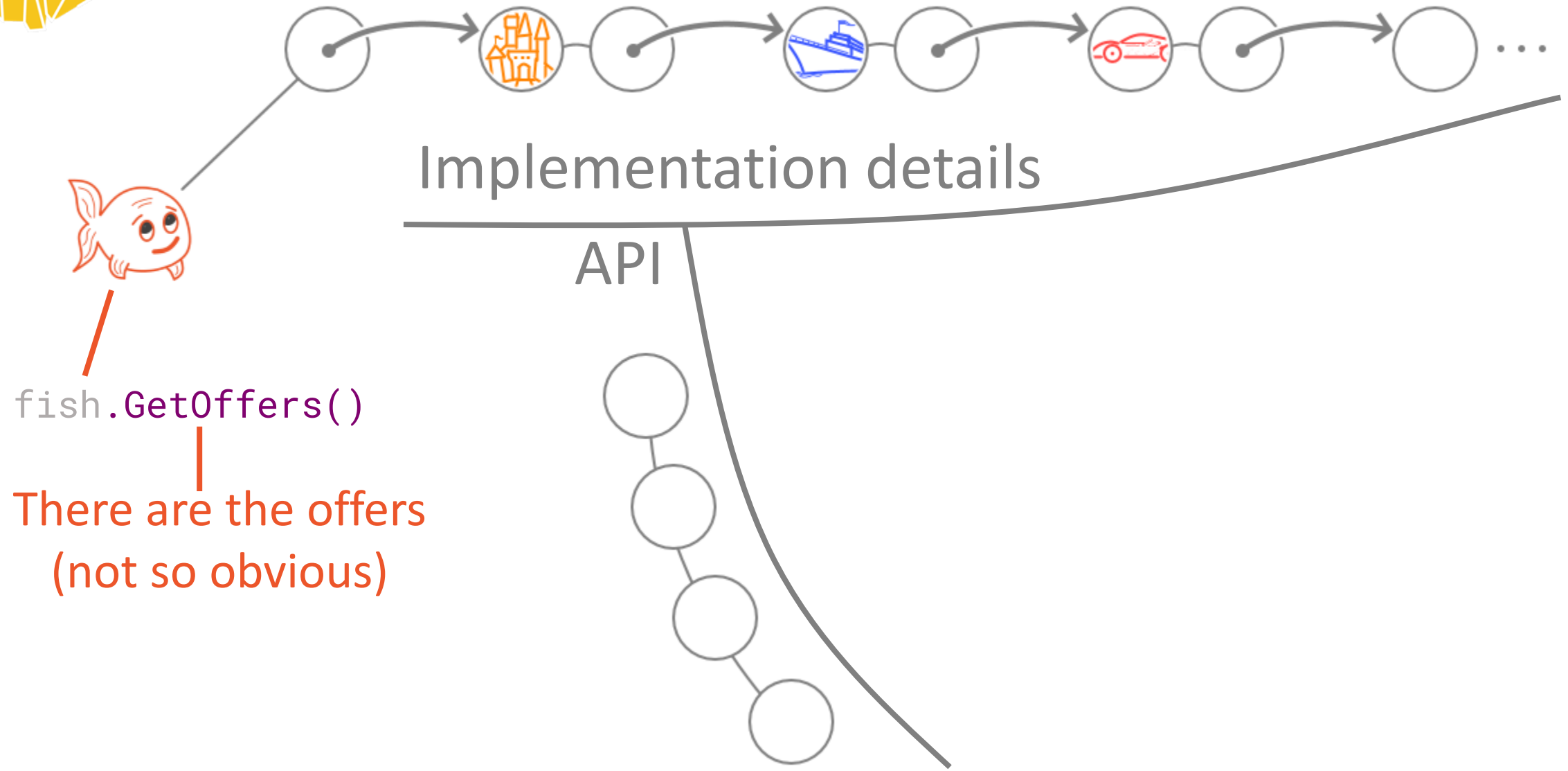
API

```
fish.GetOffers()
```

There are the offers
(not so obvious)

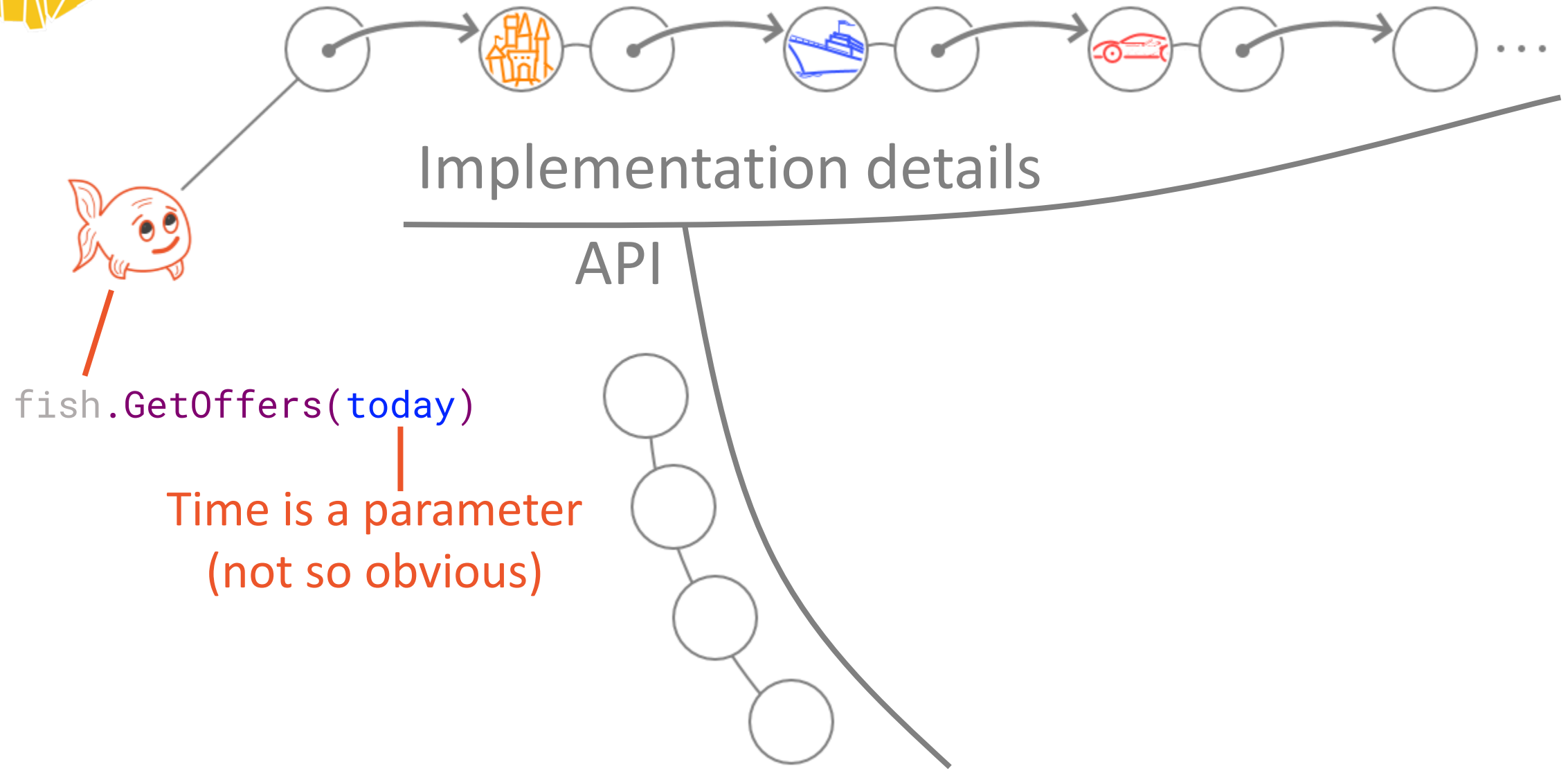
#GEEKSTONE

2022



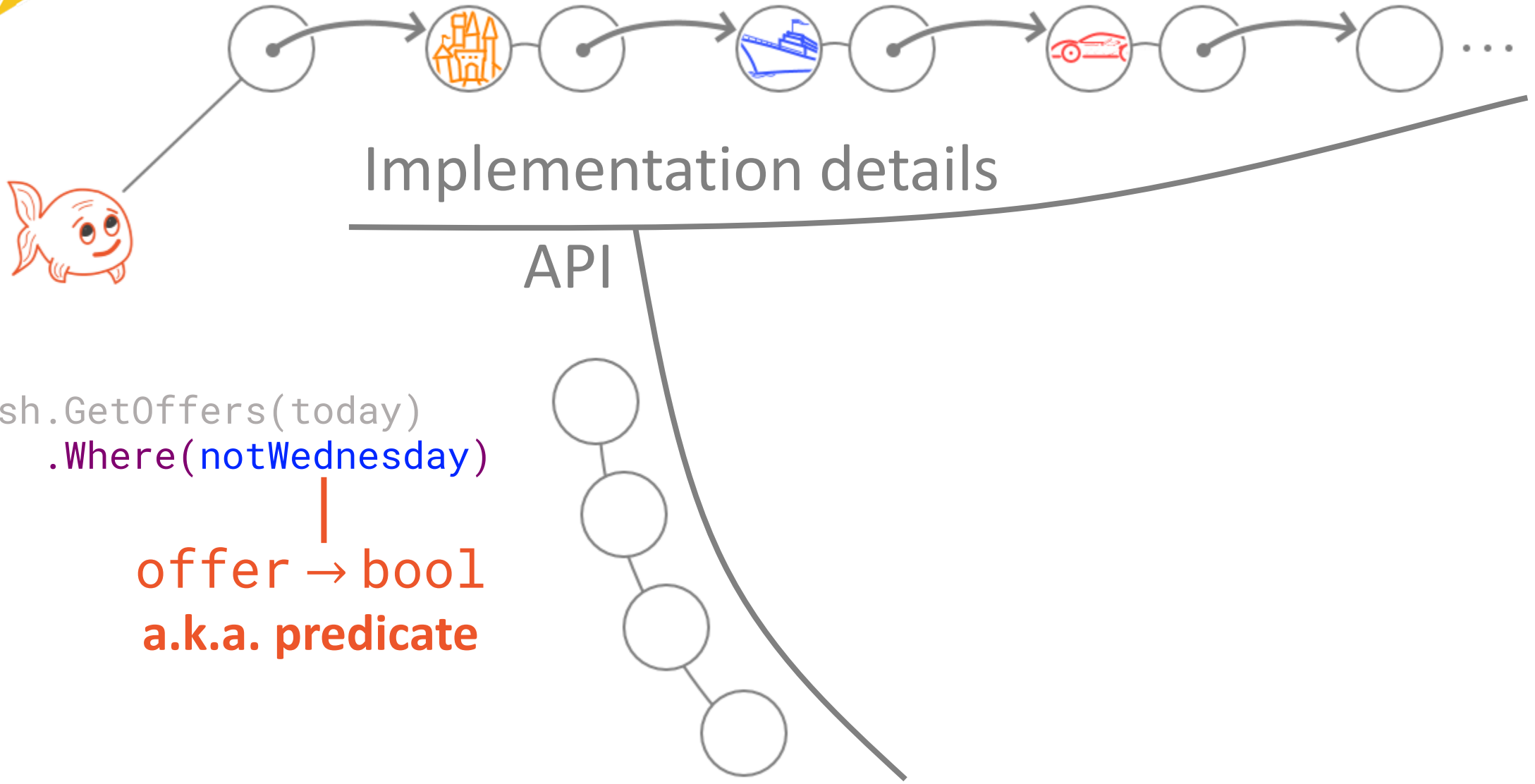
#GEEKSTONE

2022



#GEEKSTONE

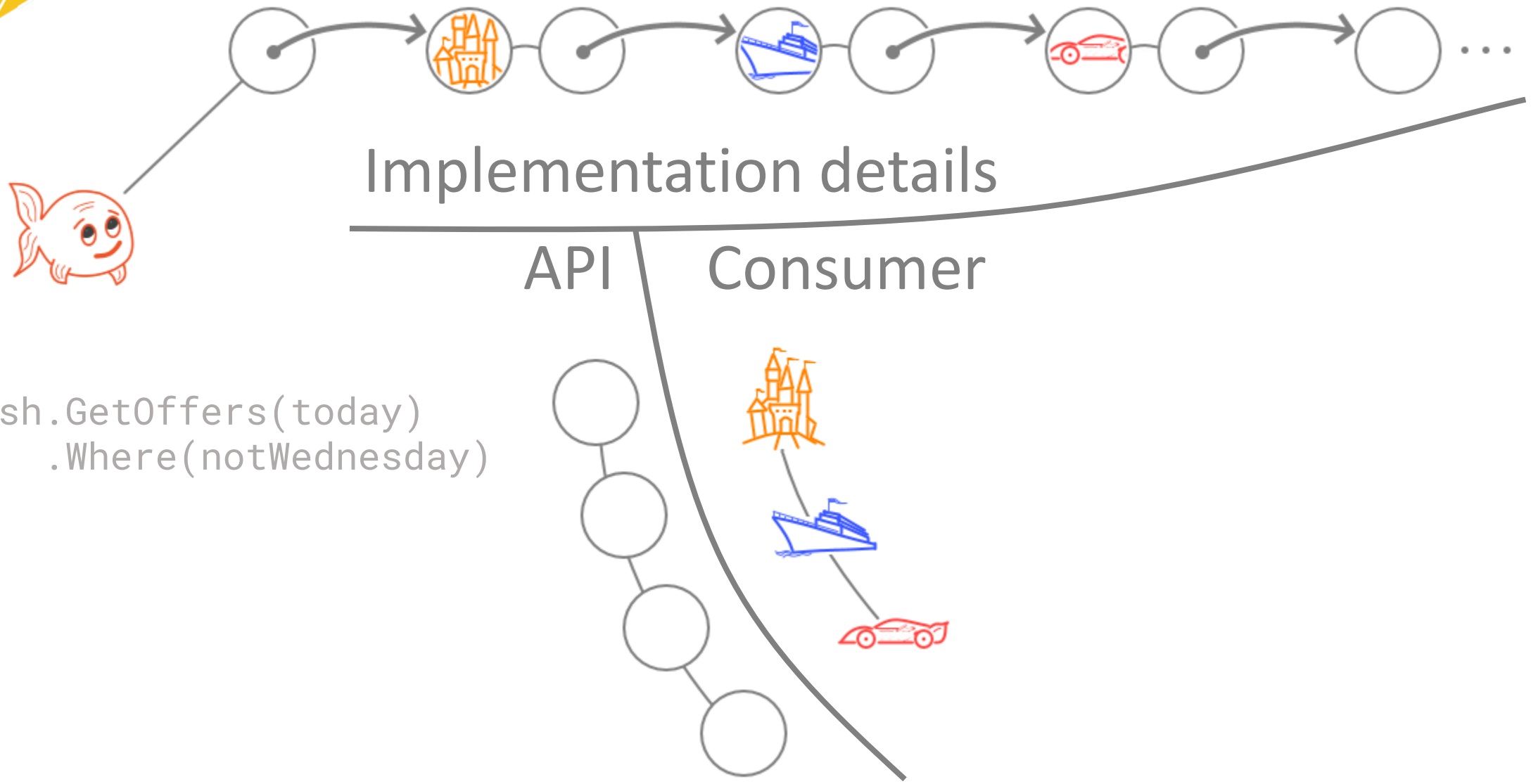
2022




```
fish.GetOffers(today)
    .Where(notWednesday)
```

`offer → bool`

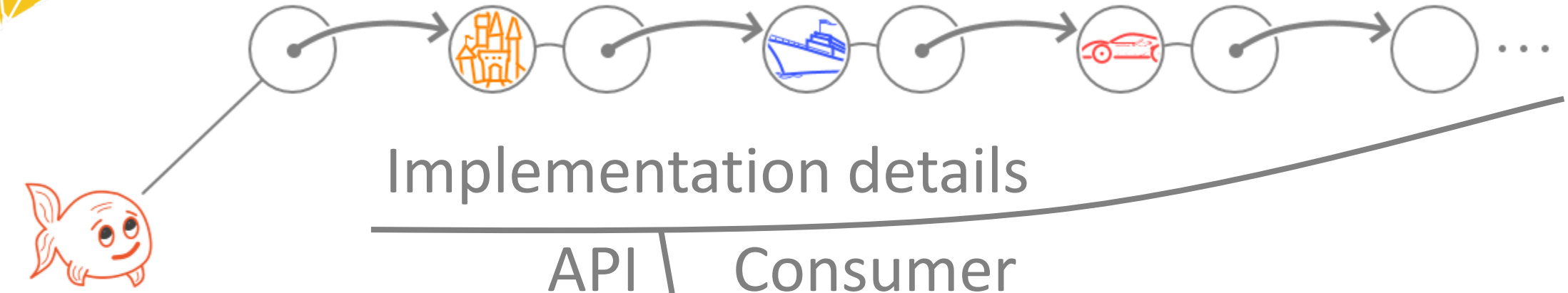
a.k.a. predicate




```
fish.GetOffers(today)
    .Where(notWednesday)
```

#GEEKSTONE

2022



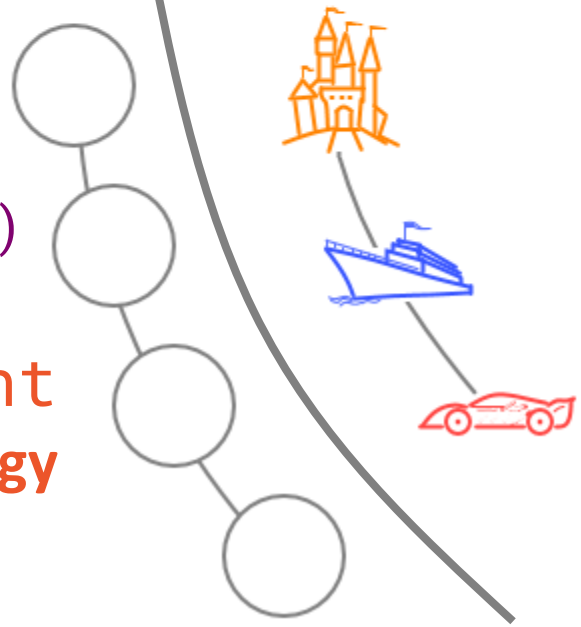
Implementation details

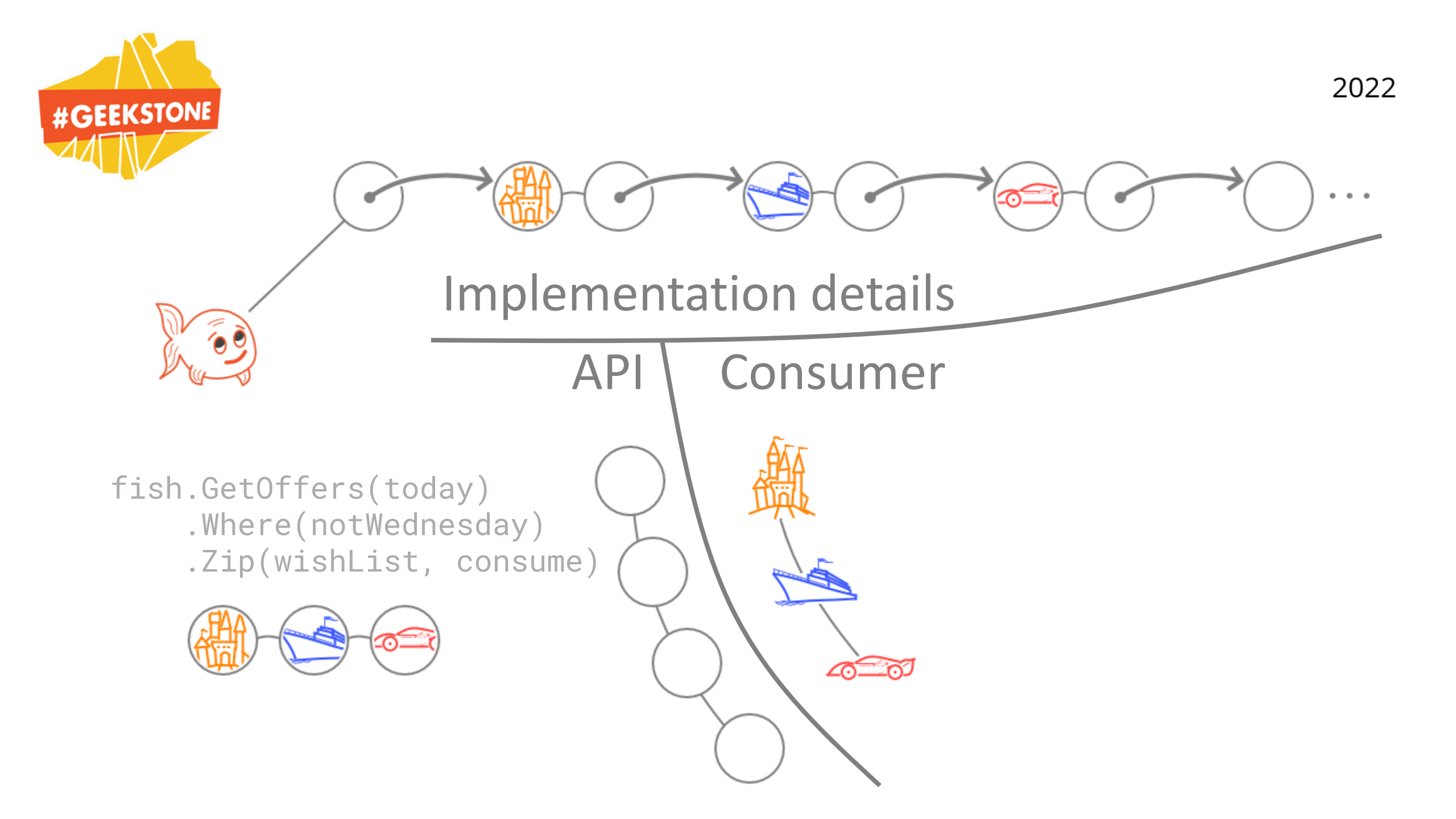
API | Consumer

```
fish.GetOffers(today)
    .Where(notWednesday)
    .Zip(wishList, consume)
```

```
(offer, wish) → fulfillment
```

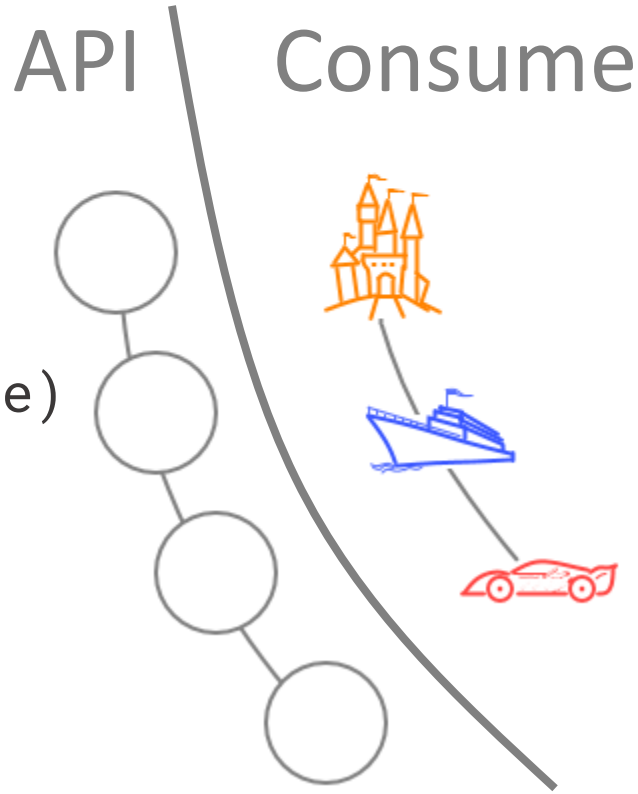
a.k.a. strategy




#GEEKSTONE
2022
Implementation details
API
Consumer
fish.GetOffers(today)
.Where(notWednesday)
.Zip(wishList, consume)

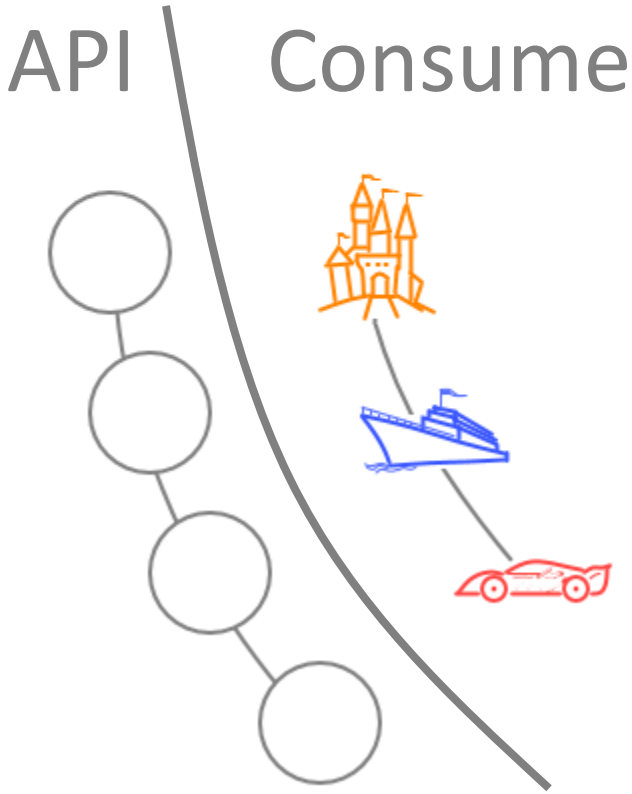
Language constructs
are not domain constructs
(should be obvious)

```
fish.GetOffers(today)
    .Where(notWednesday)
    .Zip(wishList, consume)
```



Language constructs
are not domain constructs
(should be obvious)

```
fish.GetOffers(today)
    .Except(onWednesday)
    .Fullfill(wishList)
```



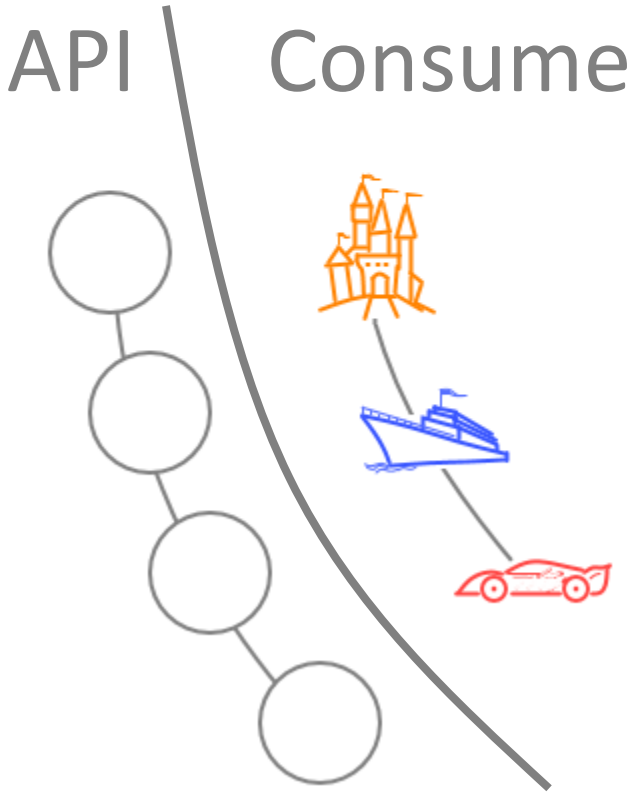
...

F# Function application (in functional languages)

Java Transparent stream proxy + interface default methods

.NET Extension methods

```
fish.GetOffers(today)
    .Except(onWednesday)
    .Fullfill(wishList)
```



2022

Tend to move towards declarative statements

Verbs of the domain language

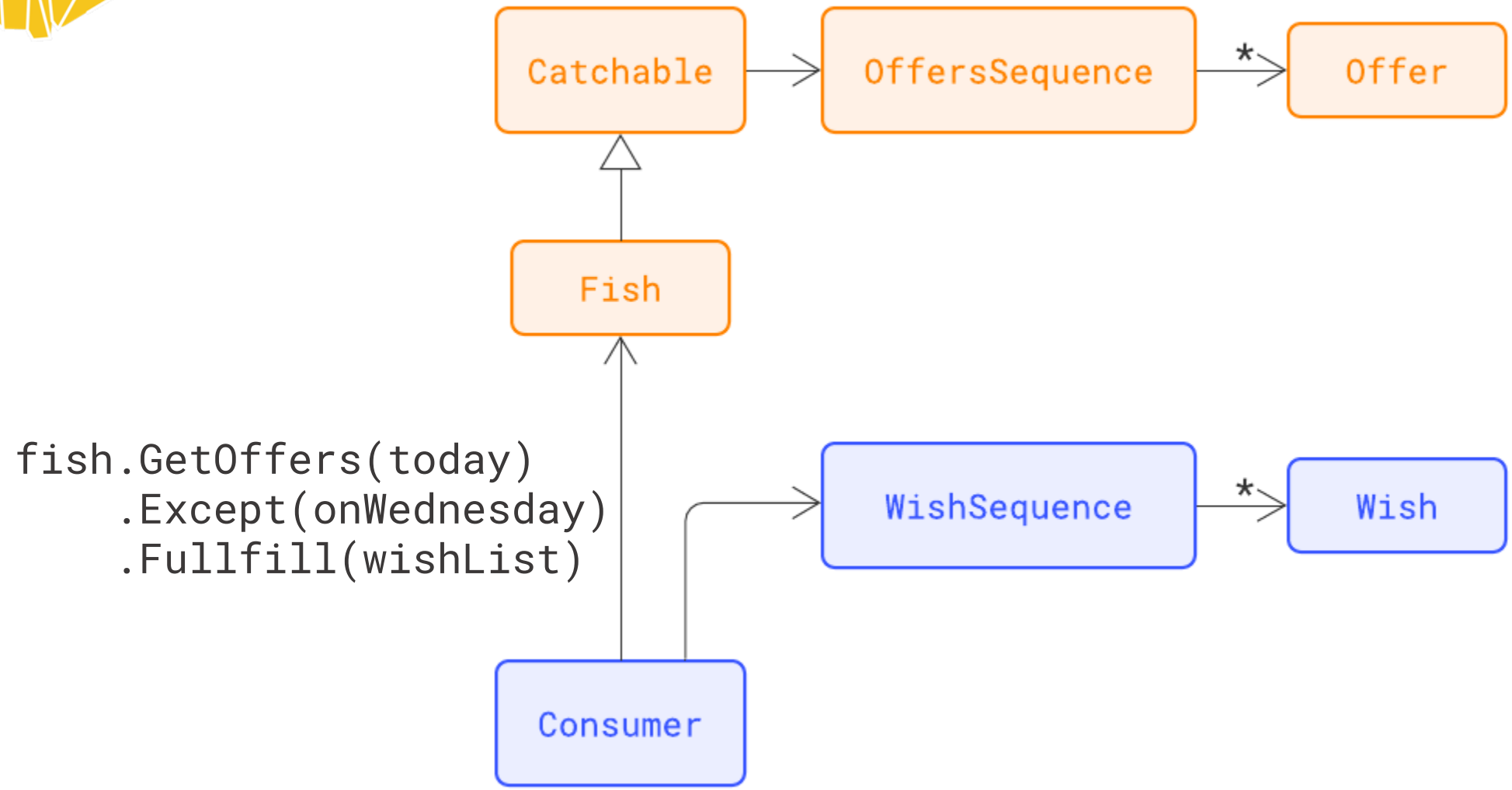
Let the code speak the same language domain experts speak

```
fish.GetOffers(today)
    .Except(onWednesday)
    .Fullfill(wishList)
```

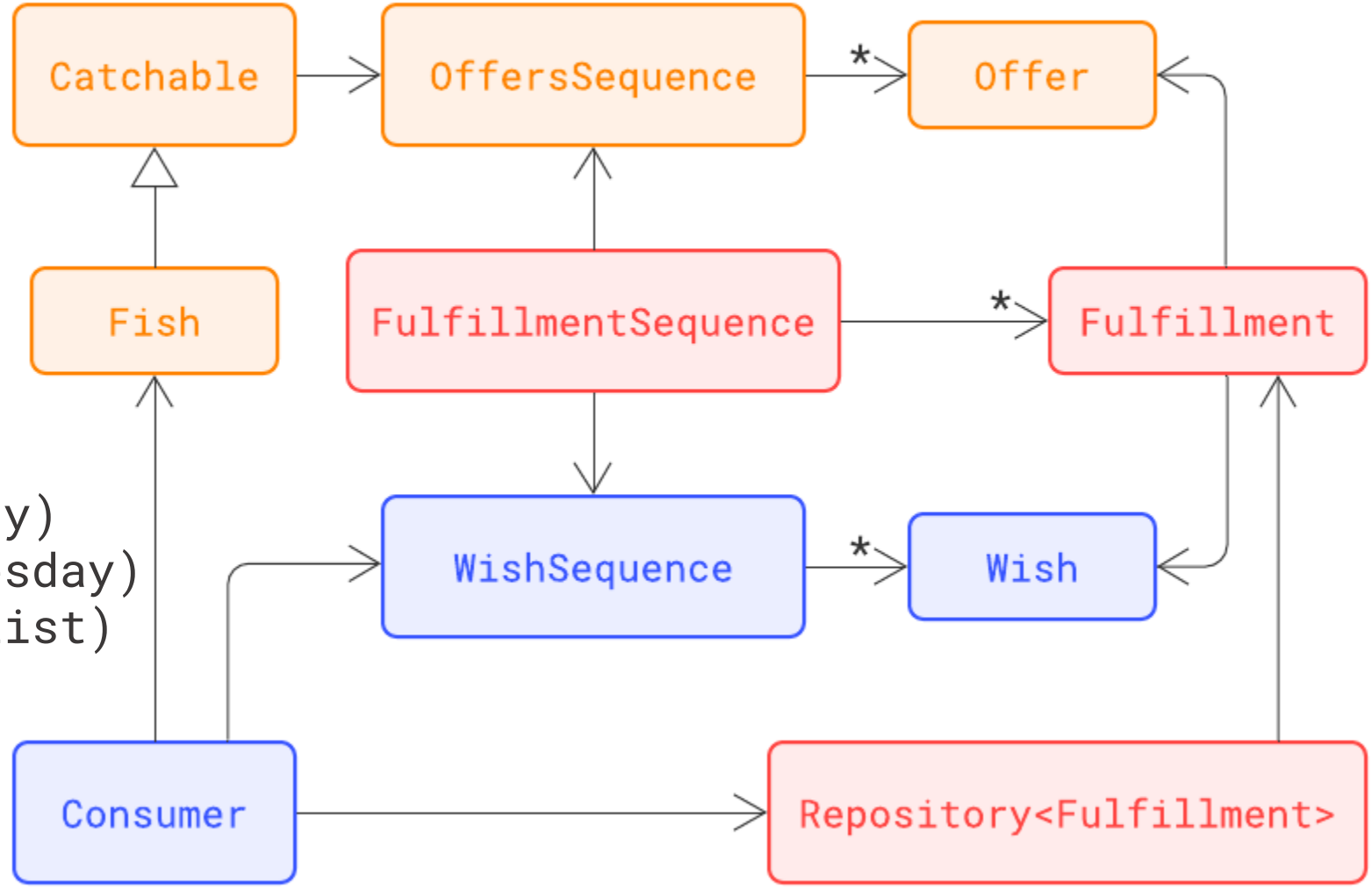
Implementing the internal domain-specific language (DSL)

nouns, prepositions, adverbs…


```
fish.GetOffers(today)
    .Except(onWednesday)
    .Fullfill(wishList)
```


Catchable
OffersSequence
*
Offer
Fish
fish.GetOffers(today)
.Except(onWednesday)
.Fullfill(wishList)
WishSequence
*
Wish
Consumer

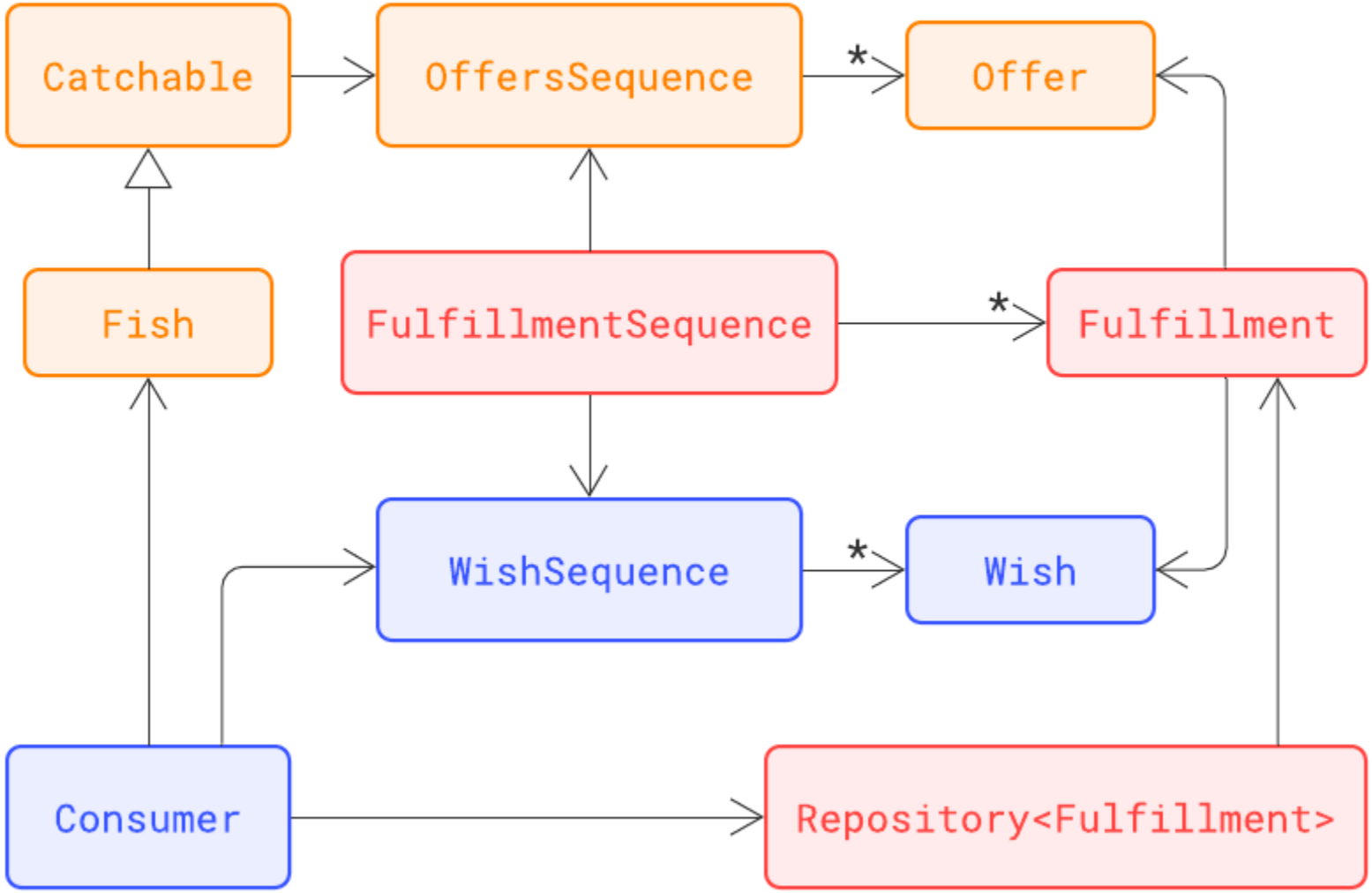
```
fish.GetOffers(today)
    .Except(onWednesday)
    .Fullfill(wishList)
```



Objects create other objects

Objects are composable

Functions are composable



Customer’s requirements:

There is the inventory

Inventory contains parts

Each part is tracked by a unique SKU

Customer’s requirements:

There is the inventory

Inventory contains parts

Each part is tracked by a unique SKU

SKU is unique within the company

Parts are purchased from other companies (vendors)

Vendors maintain their own SKUs for same parts

SKU is unique within a vendor

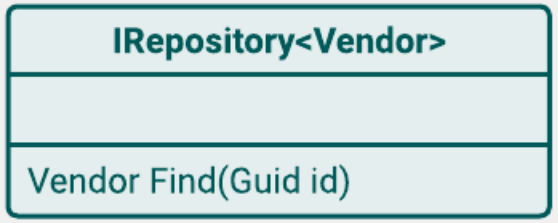
Customer’s requirements:

There is the inventory
Inventory contains parts
Each part is tracked by a unique SKU
SKU is unique within the company
Parts are purchased from other companies (vendors)
Vendors maintain their own SKUs for same parts
SKU is unique within a vendor
Company encodes SKU into a Code 128 barcode (configurable)
Barcodes printed on sticky labels
Vendors encode SKU into a barcode
Each vendor may use a different barcode format

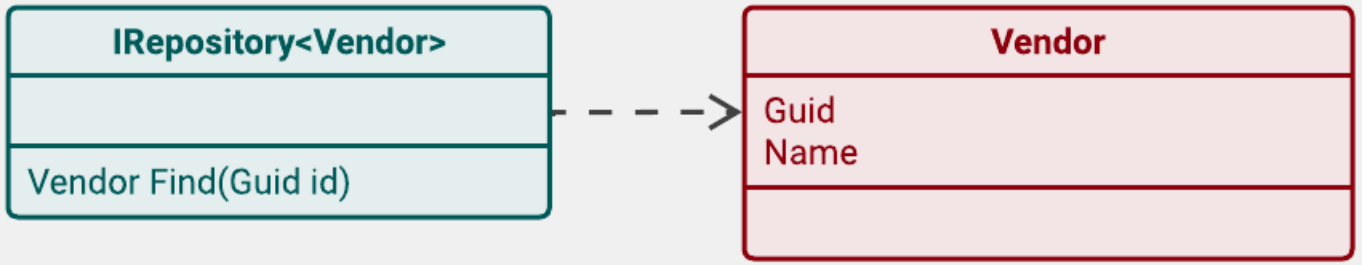
Customer’s requirements:

There is the inventory
Inventory contains parts
Each part is tracked by a unique SKU
SKU is unique within the company
Parts are purchased from other companies (vendors)
Vendors maintain their own SKUs for same parts
SKU is unique within a vendor
Company encodes SKU into a Code 128 barcode (configurable)
Barcodes printed on sticky labels
Vendors encode SKU into a barcode
Each vendor may use a different barcode format
Incoming items subjected to a barcode scanner
Incoming barcodes read off an image and stored
Inventory is tracking quantity of each item
Incoming items arrive with a quantity
The same item comes from multiple vendors
A vendor can change barcode technology
There can be multiple incoming barcode technologies used
Inventory can report low reserves on any item
There is a planner which tracks expenditure of parts
There are specifications of products built from parts
Specification consists of assembly steps
Each assembly step consumes one part
Specification exposes list of required parts with quantities
Inventory can verify a specification against current quantities

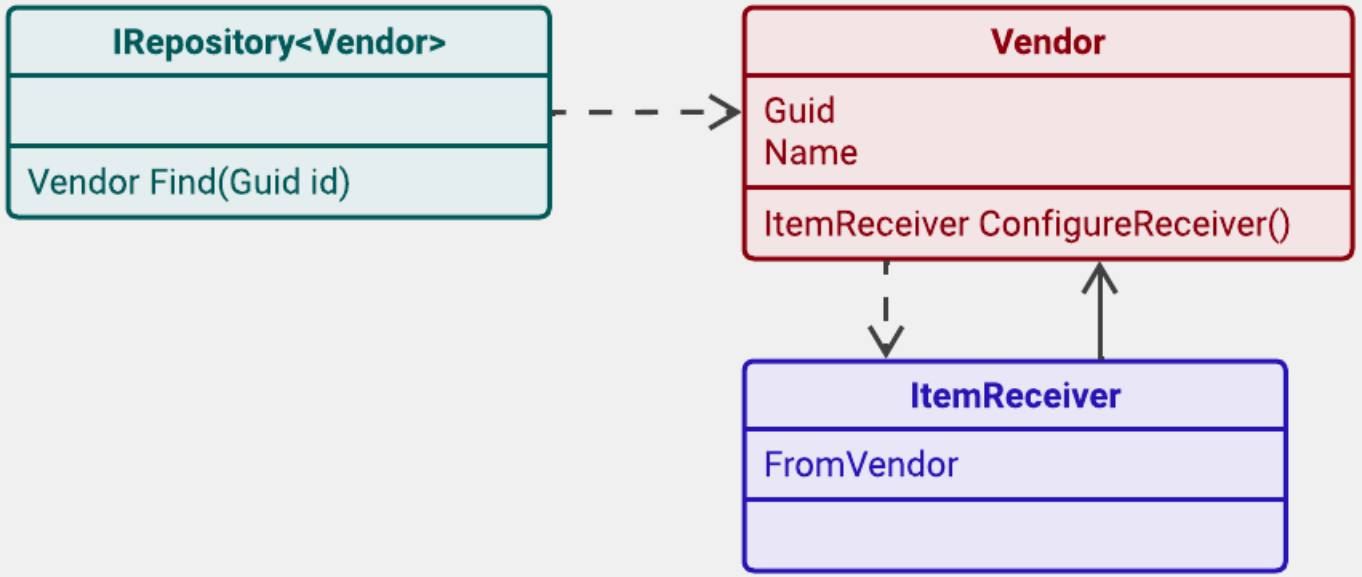
vendorRepo



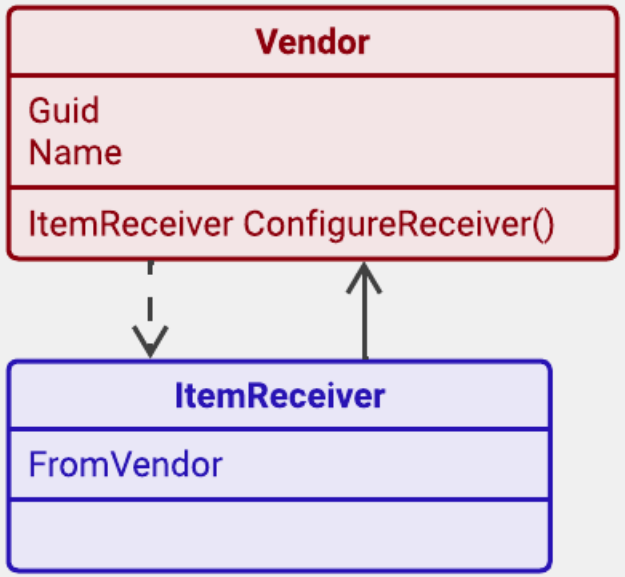
```
vendorRepo
    .Find(vendor)
          // Vendor
```



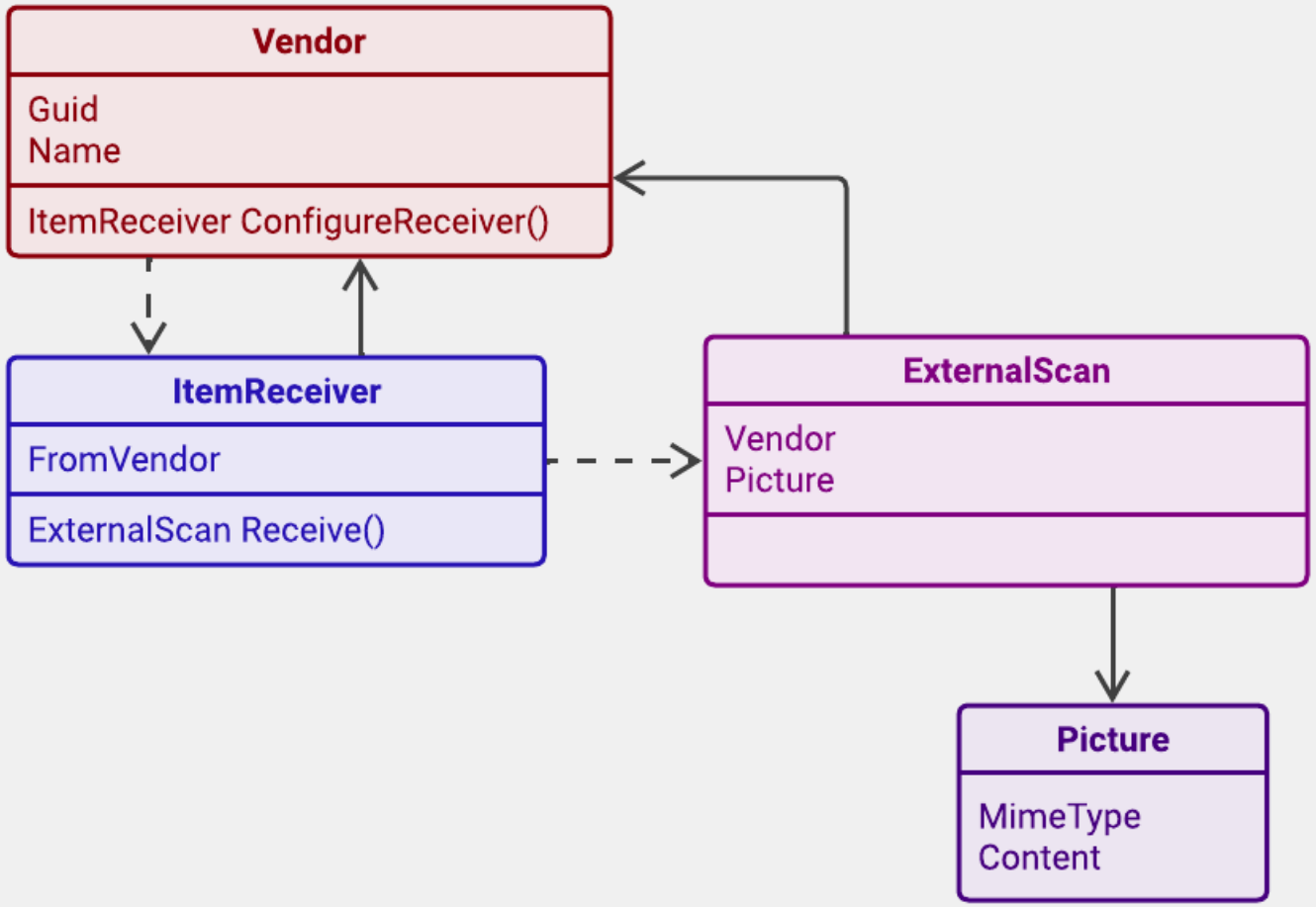
```
vendorRepo
    .Find(vendor)
            // Vendor

    .ConfigureReceiver()
            // ItemReceiver
```



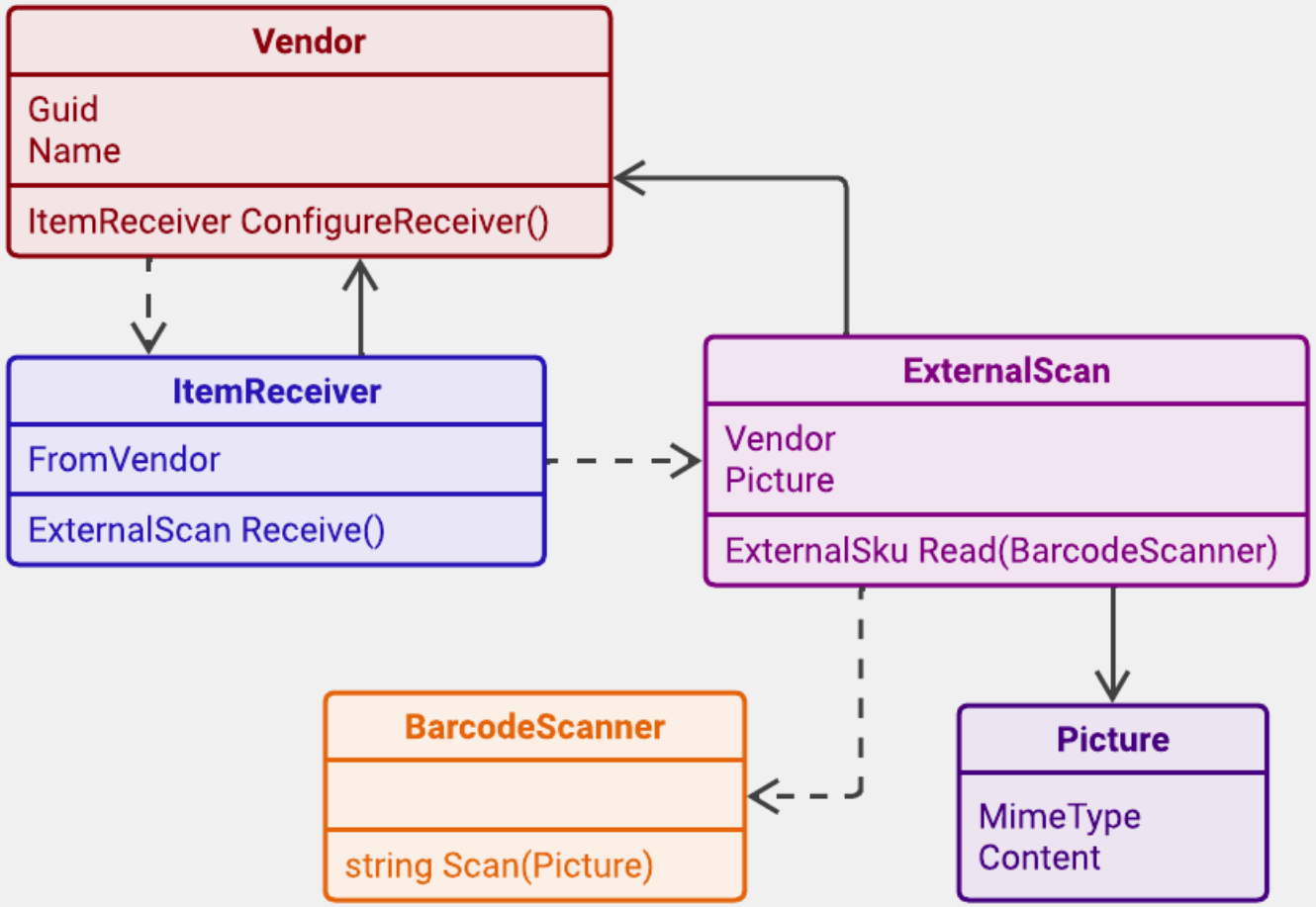
```
vendorRepo

   .Find(vendor)
         // Vendor

   .ConfigureReceiver()
         // ItemReceiver
```



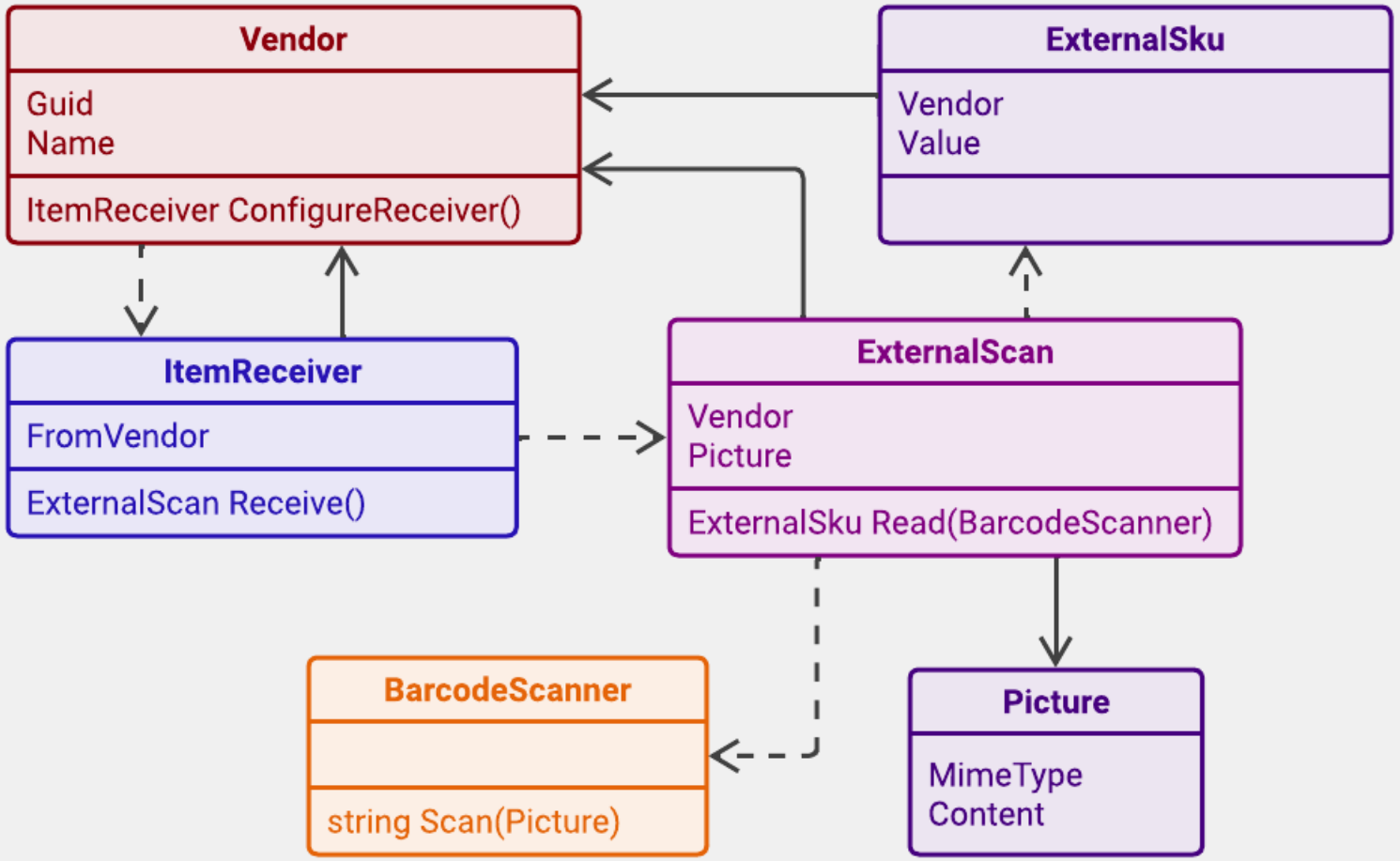
```
vendorRepo
    .Find(vendor)
          // Vendor

    .ConfigureReceiver()
          // ItemReceiver

    .Receive()
          // ExternalScan
```



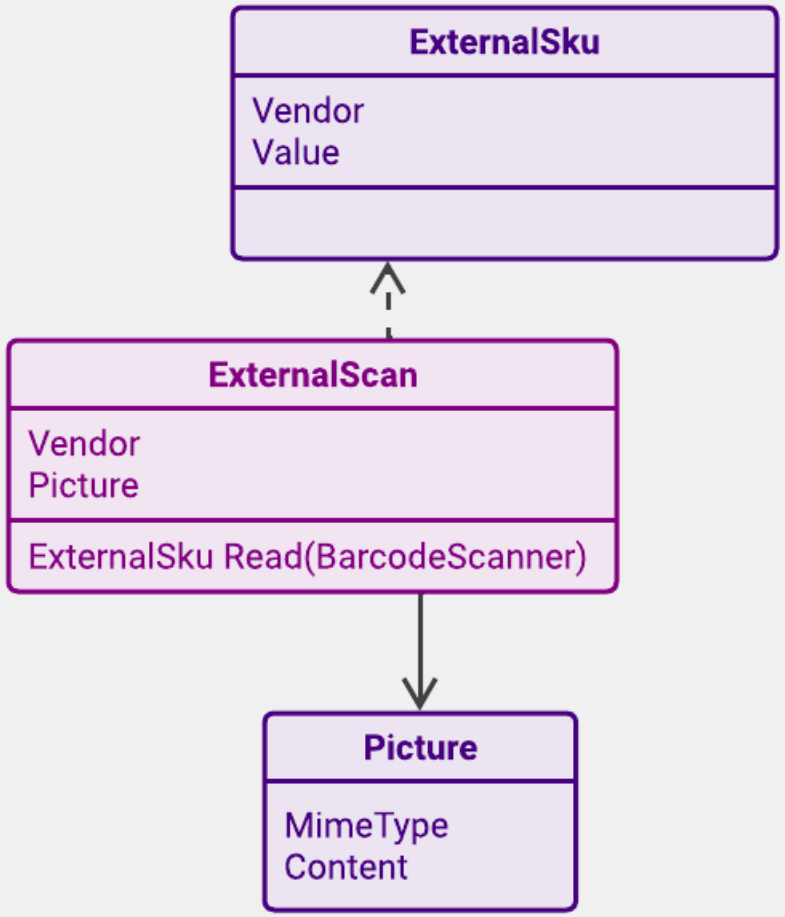
```
vendorRepo
    .Find(vendor)
          // Vendor

    .ConfigureReceiver()
          // ItemReceiver

    .Receive()
          // ExternalScan

    .Read(scanner)
```



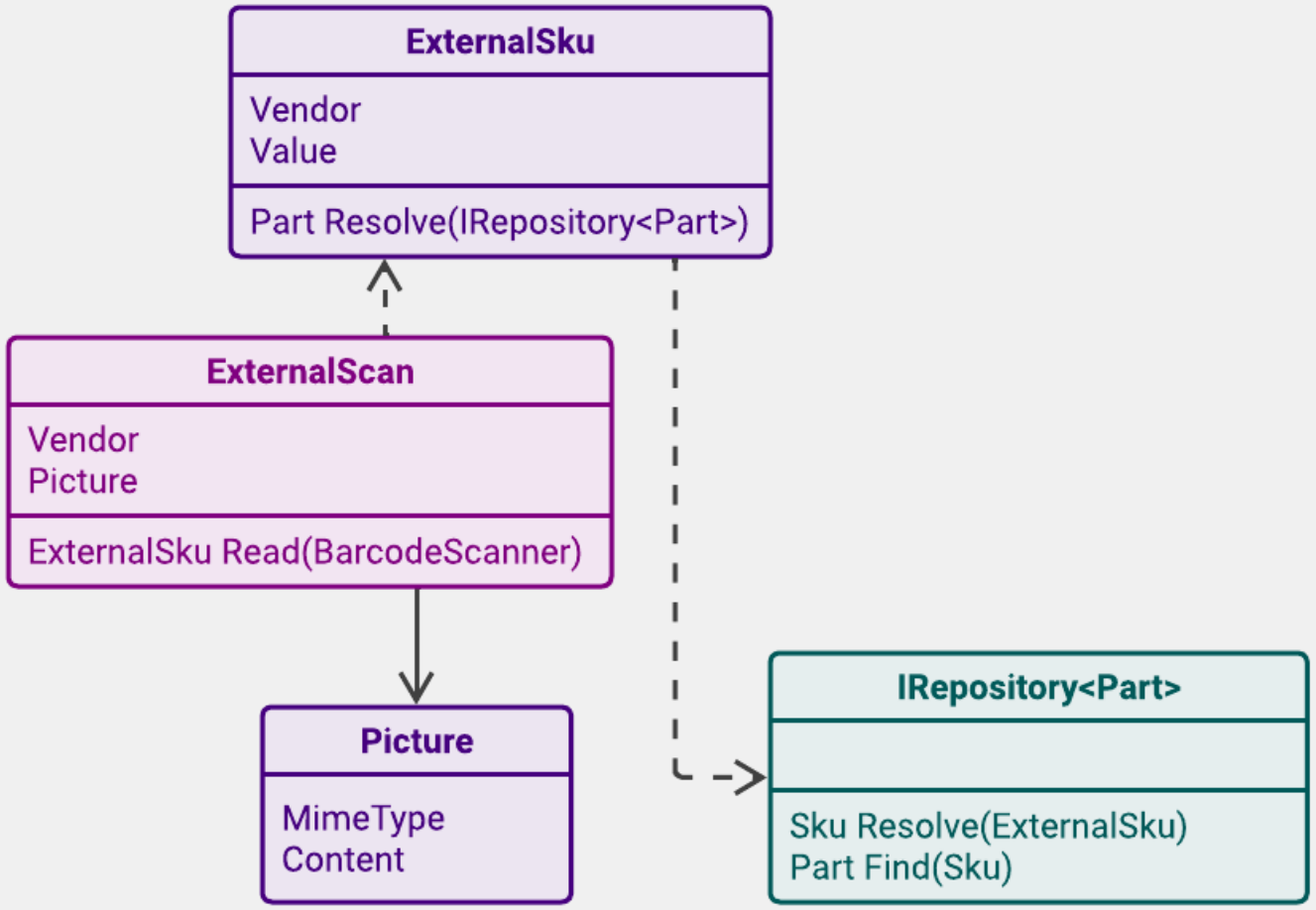
```
vendorRepo
    .Find(vendor)
          // Vendor

    .ConfigureReceiver()
          // ItemReceiver

    .Receive()
          // ExternalScan

    .Read(scanner)
          // ExternalSku
```



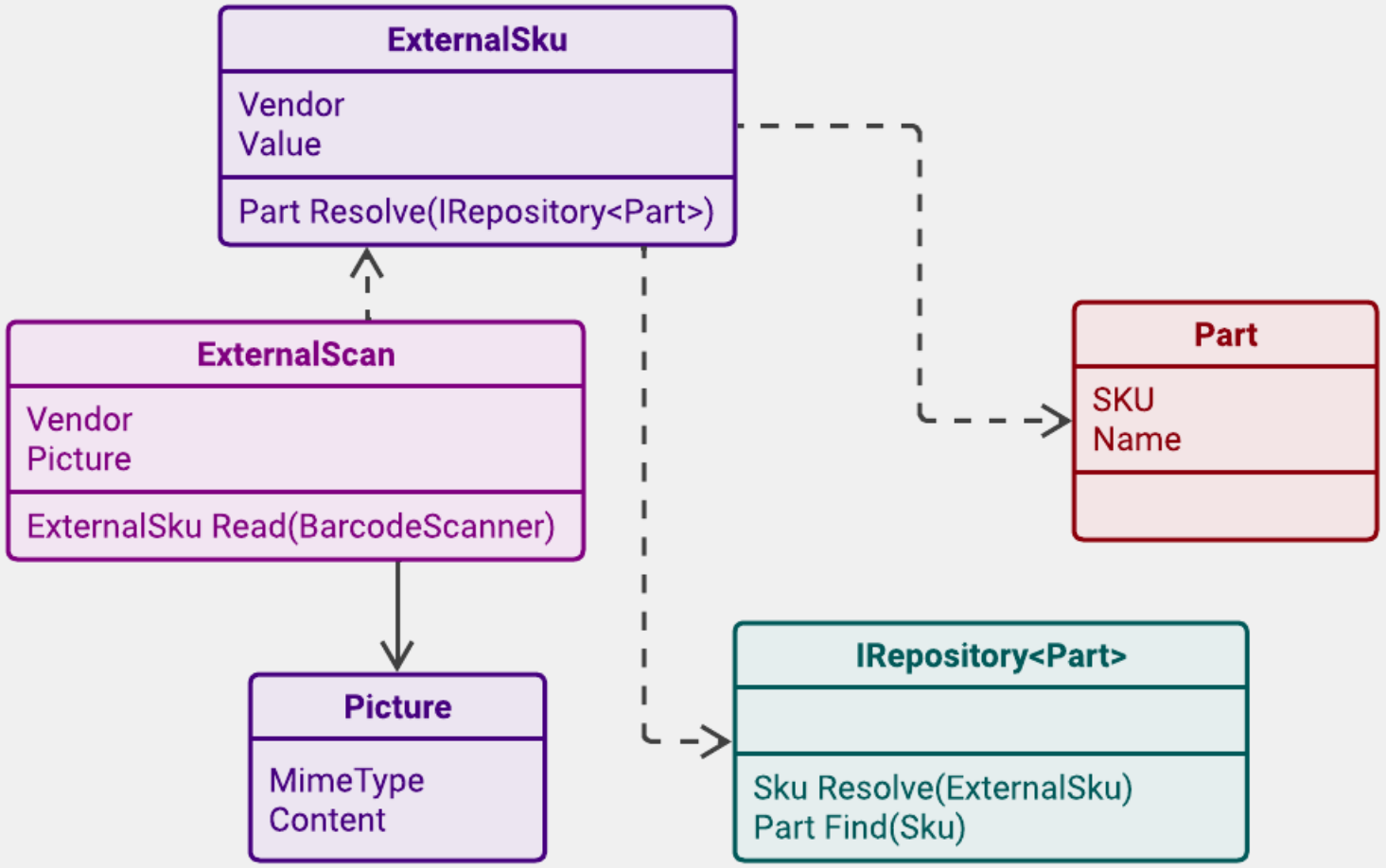
```
vendorRepo
    .Find(vendor)
          // Vendor

    .ConfigureReceiver()
          // ItemReceiver

    .Receive()
          // ExternalScan

    .Read(scanner)
          // ExternalSku
```



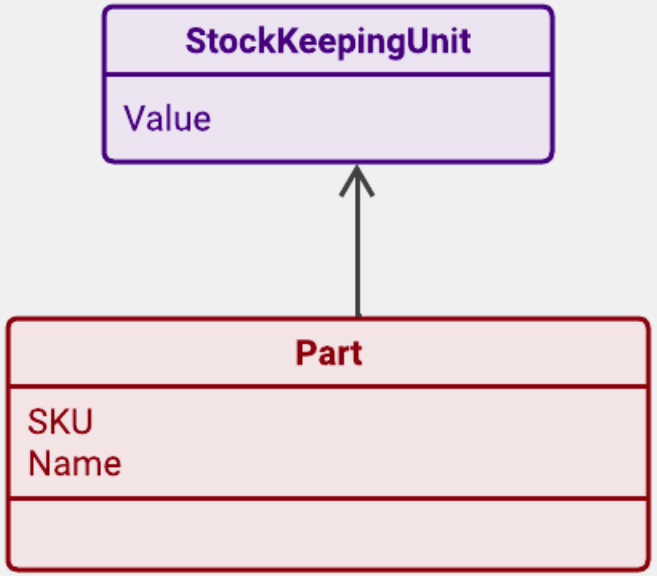
```
vendorRepo
    .Find(vendor)
          // Vendor
    .ConfigureReceiver()
          // ItemReceiver
    .Receive()
          // ExternalScan
    .Read(scanner)
          // ExternalSku
    .Resolve(partsRepo)
```



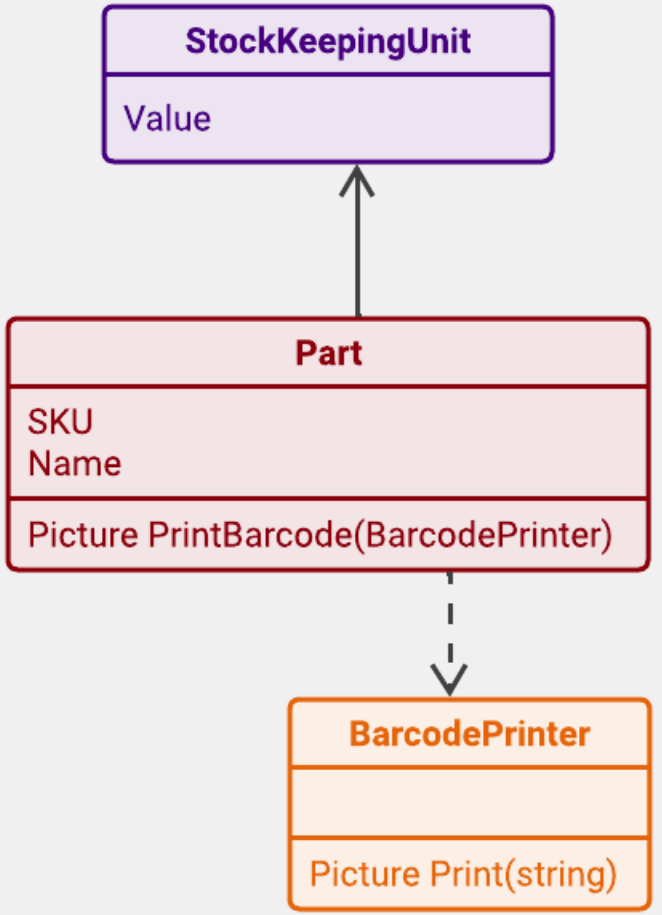
```
vendorRepo
    .Find(vendor)
          // Vendor
    .ConfigureReceiver()
          // ItemReceiver
    .Receive()
          // ExternalScan
    .Read(scanner)
          // ExternalSku
    .Resolve(partsRepo)
```



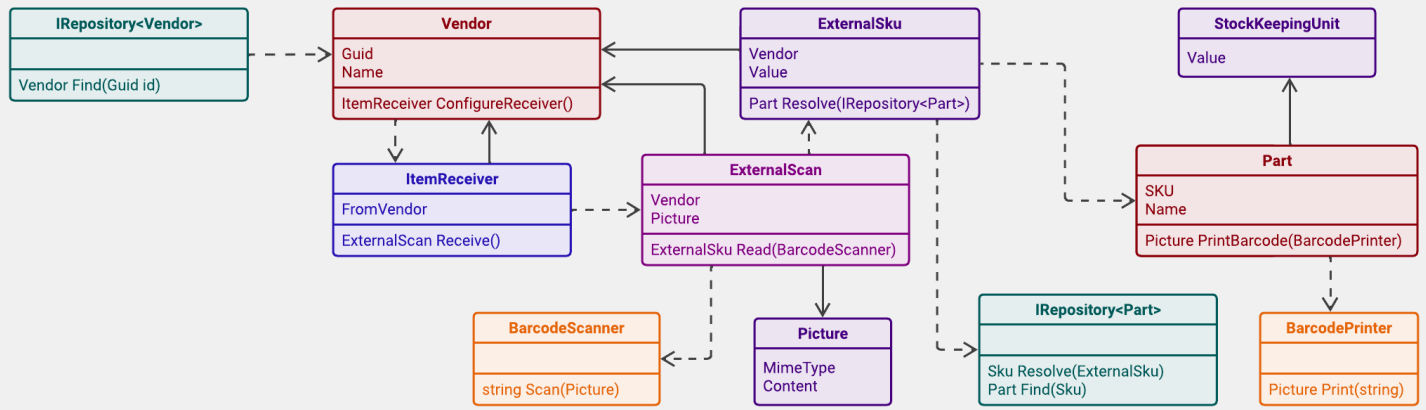
```
vendorRepo
    .Find(vendor)
          // Vendor

    .ConfigureReceiver()
          // ItemReceiver

    .Receive()
          // ExternalScan

    .Read(scanner)
          // ExternalSku

    .Resolve(partsRepo)
          // Part
```



```
var label = vendorRepo
    .Find(vendor)
          // Vendor
    .ConfigureReceiver()
          // ItemReceiver
    .Receive()
          // ExternalScan
    .Read(scanner)
          // ExternalSku
    .Resolve(partsRepo)
          // Part
    .PrintBarcode(printer);
          // Picture
```



```
var label = vendorRepo
   .Find(vendor)
   .ConfigureReceiver()
   .Receive()
   .Read(scanner)
   .Resolve(partsRepo)
   .PrintBarcode(printer);
```




#GEEKSTONE
THANK YOU!

SUPPORTED BY:

VEGA IT