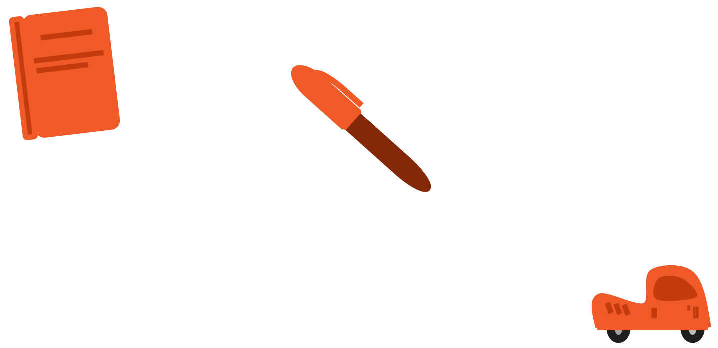
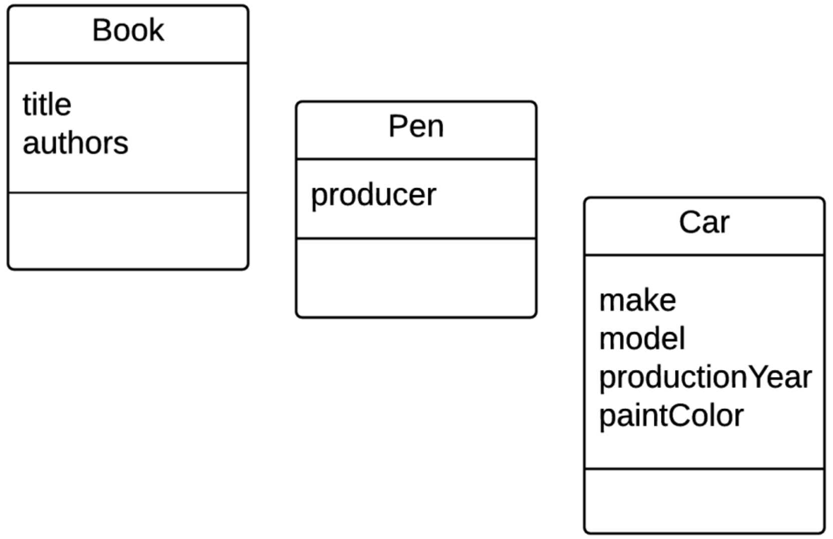
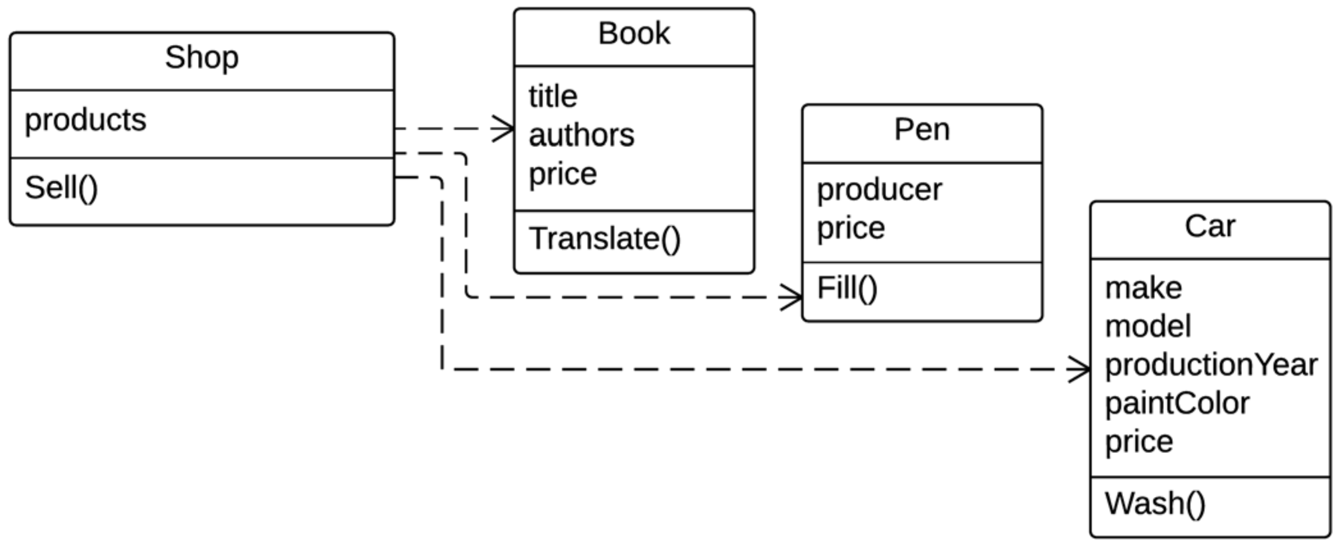
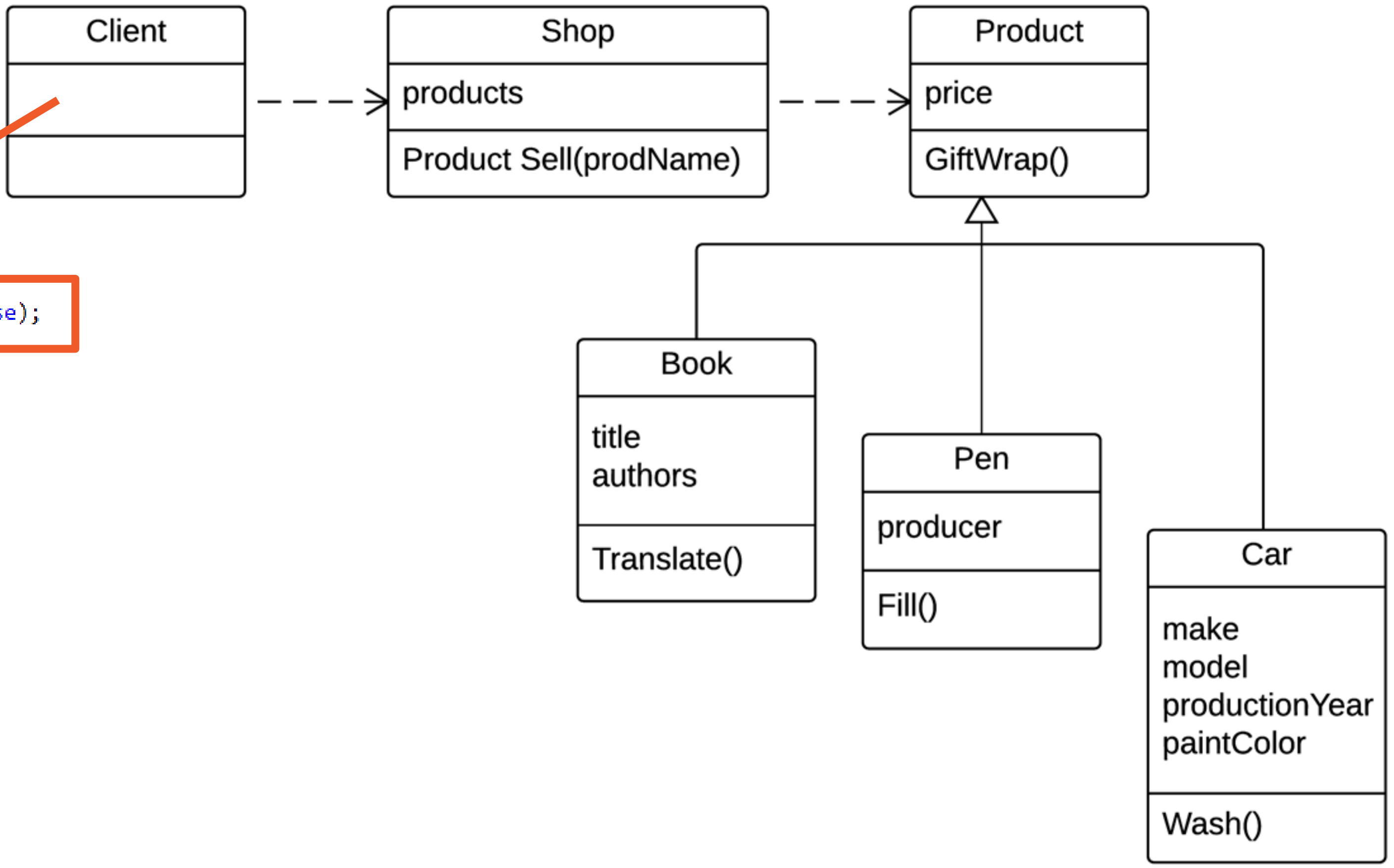
Option<T> Functional Type

Zoran Horvat
CTO at InterVenture GmbH
@zoranh75 www.codinghelmet.com

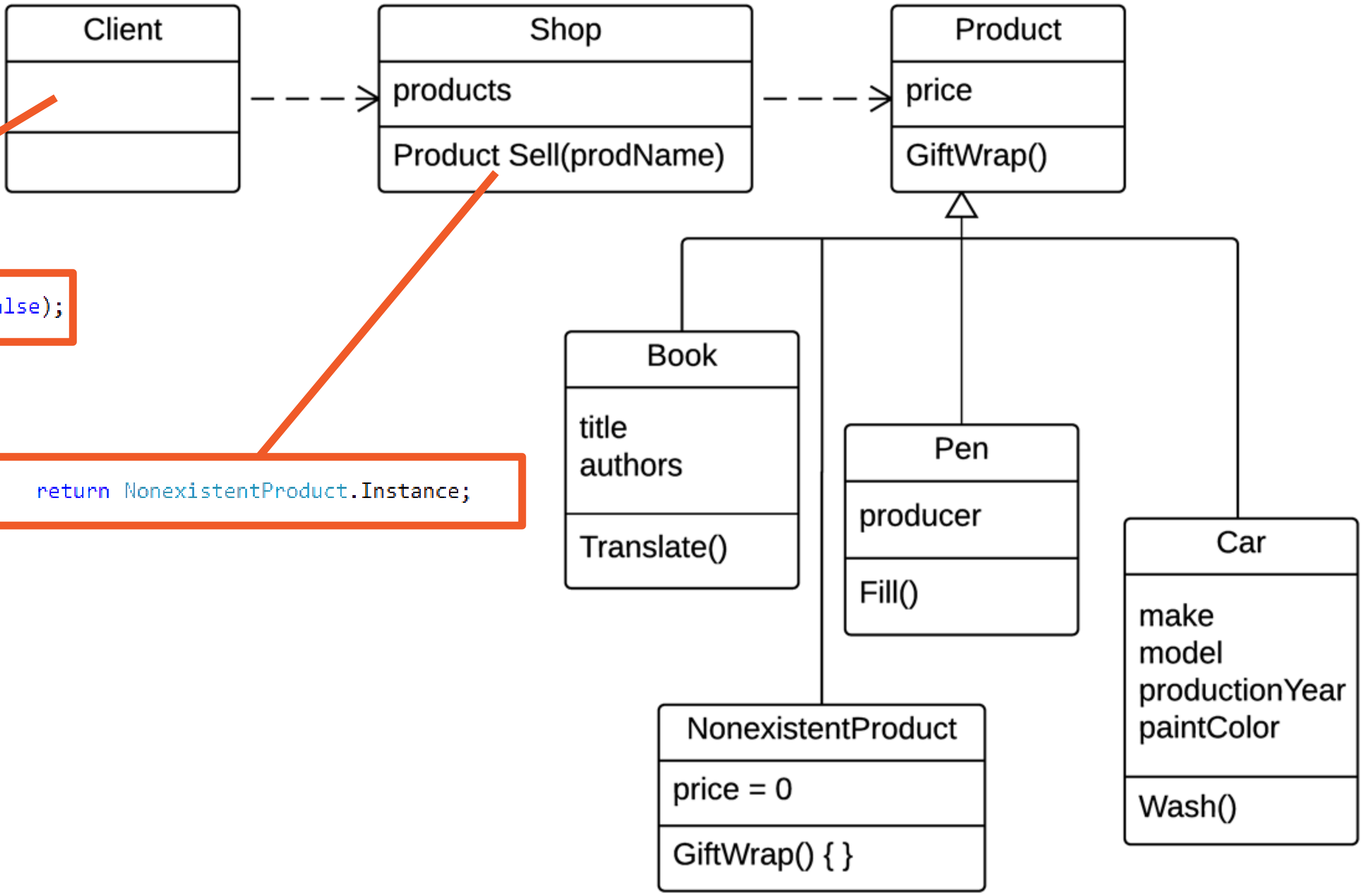



Book
title
authors
Pen
producer
Car
make
model
productionYear
paintColor


Shop
products
Sell()
Book
title
authors
price
Translate()
Pen
producer
price
Fill()
Car
make
model
productionYear
paintColor
price
Wash()

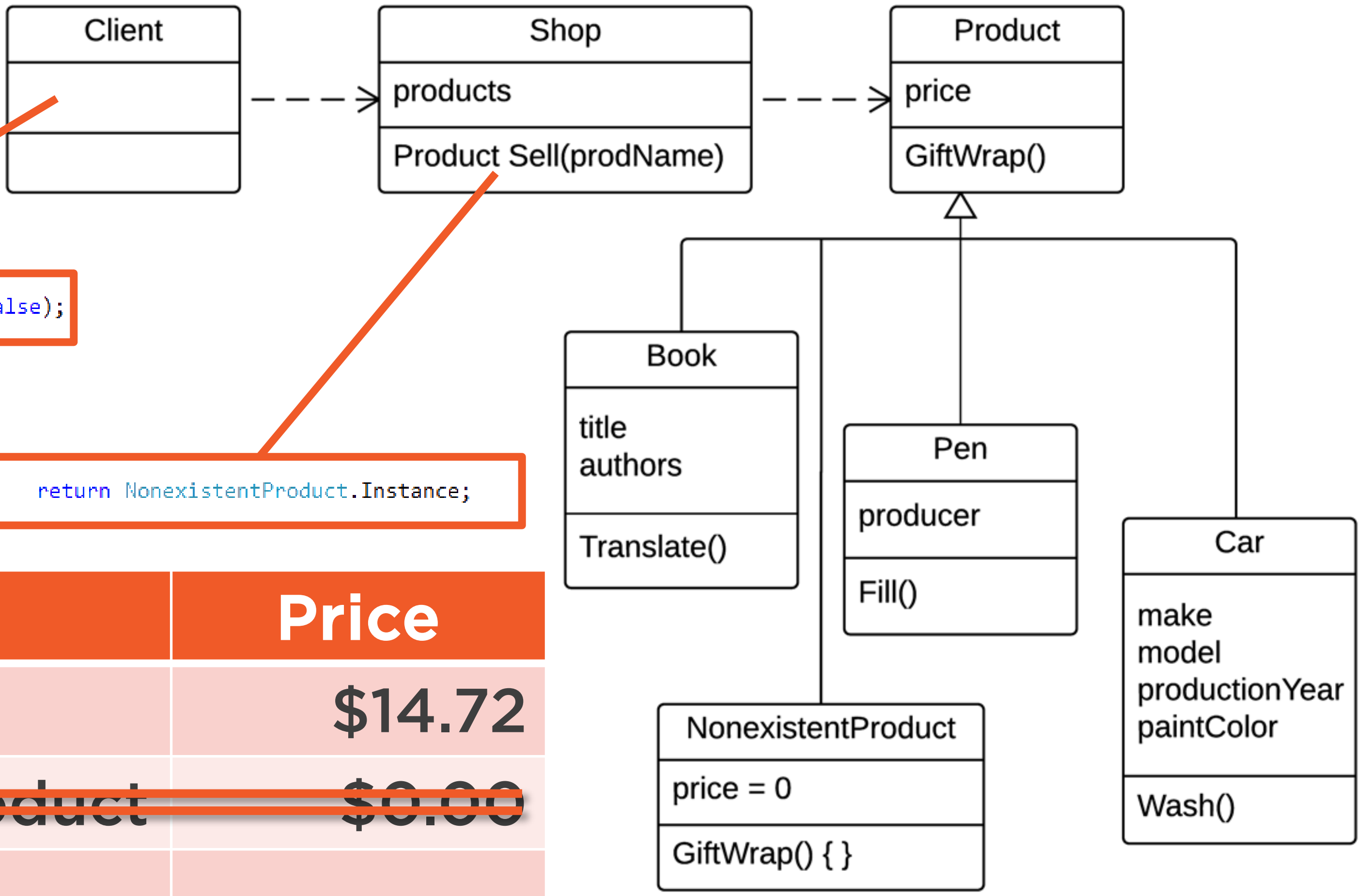


```
Product book = shop.Sell("book", false);
```



```
Product laptop = shop.Sell("laptop", false);
```

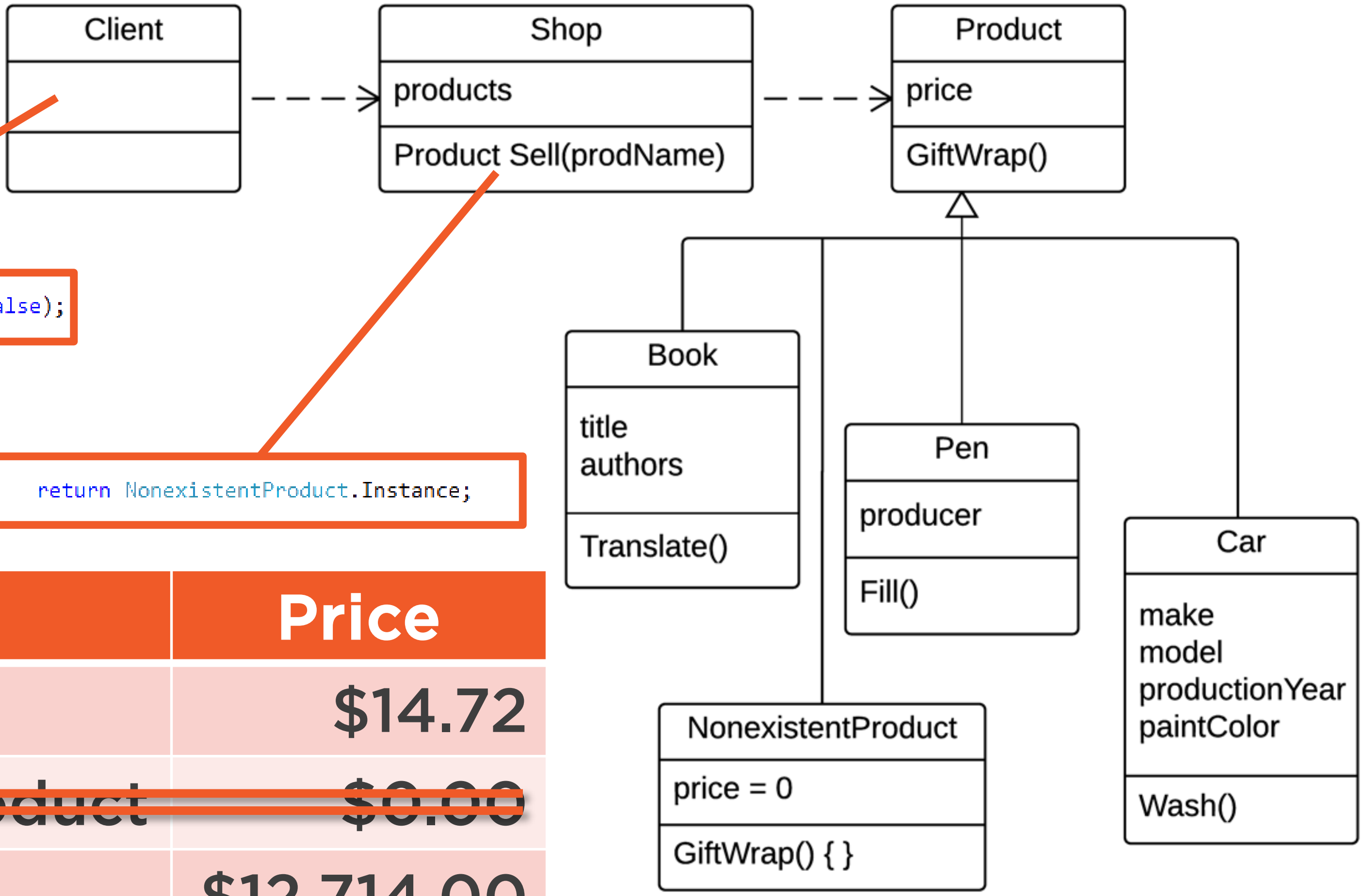
```
return NonexistentProduct.Instance;
```



```
Product laptop = shop.Sell("laptop", false);
```

```
return NonexistentProduct.Instance;
```

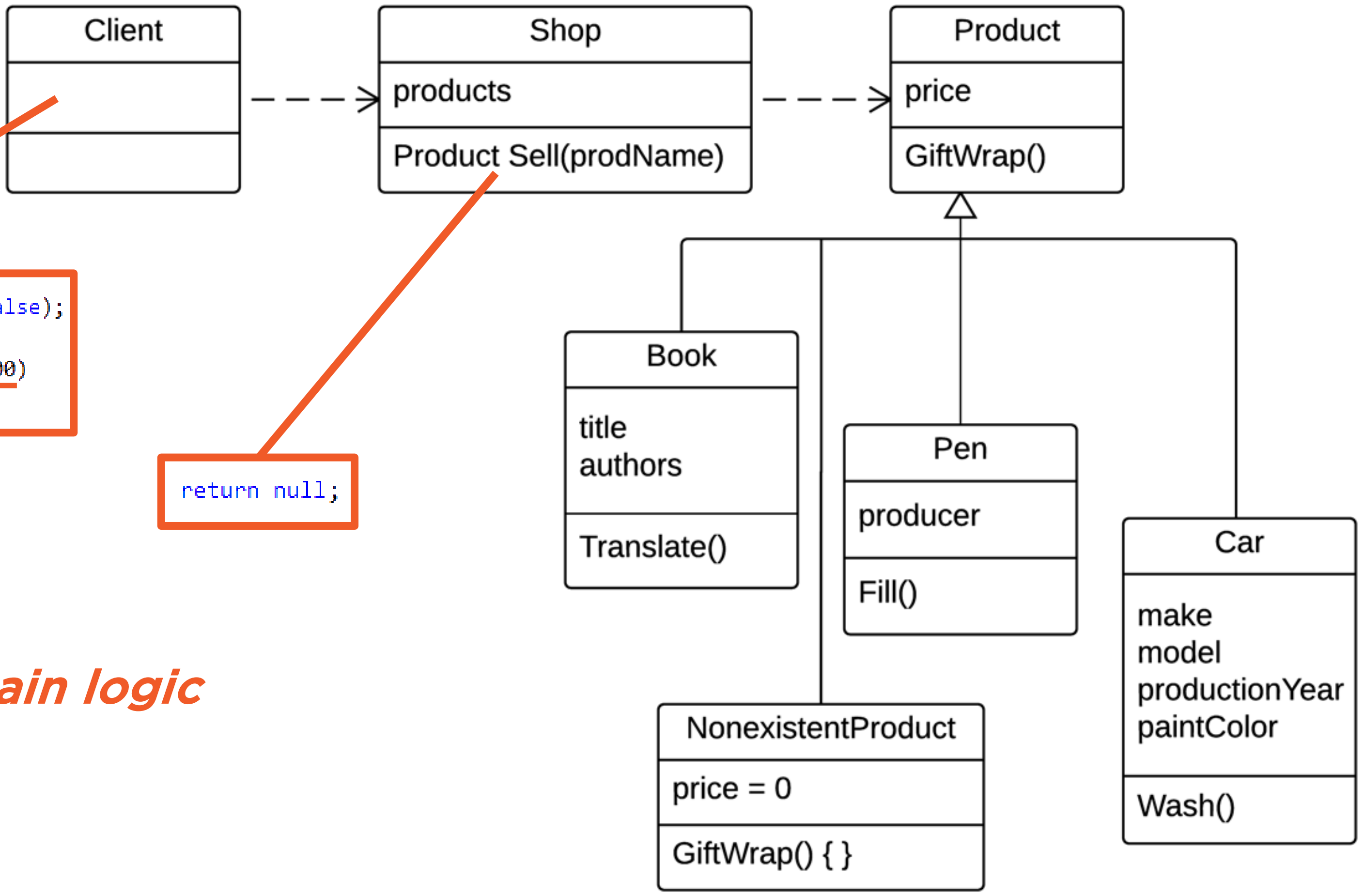
Purchase	Price
Book	$14.72
~~Nonexistent product~~	~~$0.00~~



```
Product laptop = shop.Sell("laptop", false);
```

```
return NonexistentProduct.Instance;
```

Purchase	Price
Book	$14.72
~~Nonexistent product~~	~~$0.00~~
Car	$12,714.00
Book	$\boxed{\$0.00}$



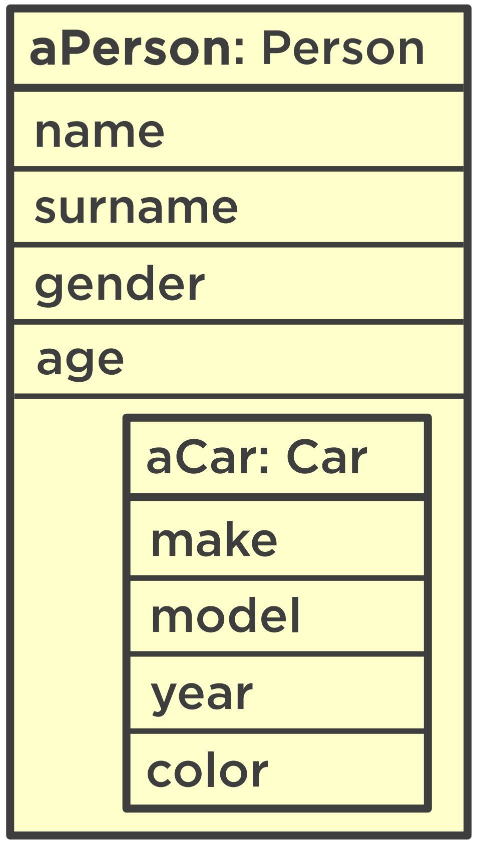
```
Product laptop = shop.Sell("laptop", false);

if (laptop != null && laptop.Price > 100)
    Report(laptop);
```

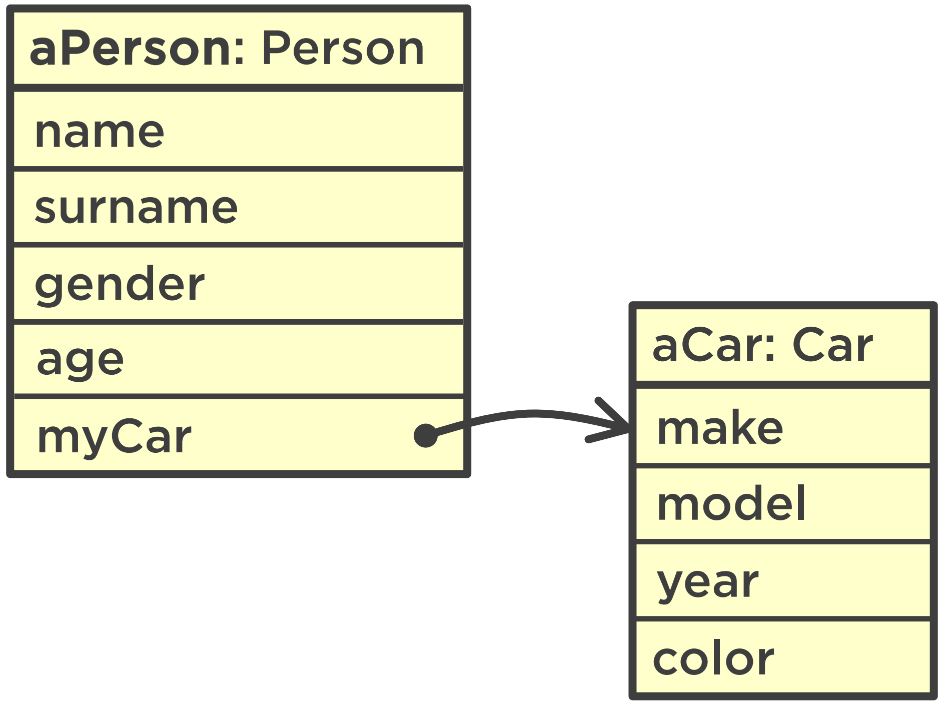
```
return null;
```

Infrastructure ***Domain logic***

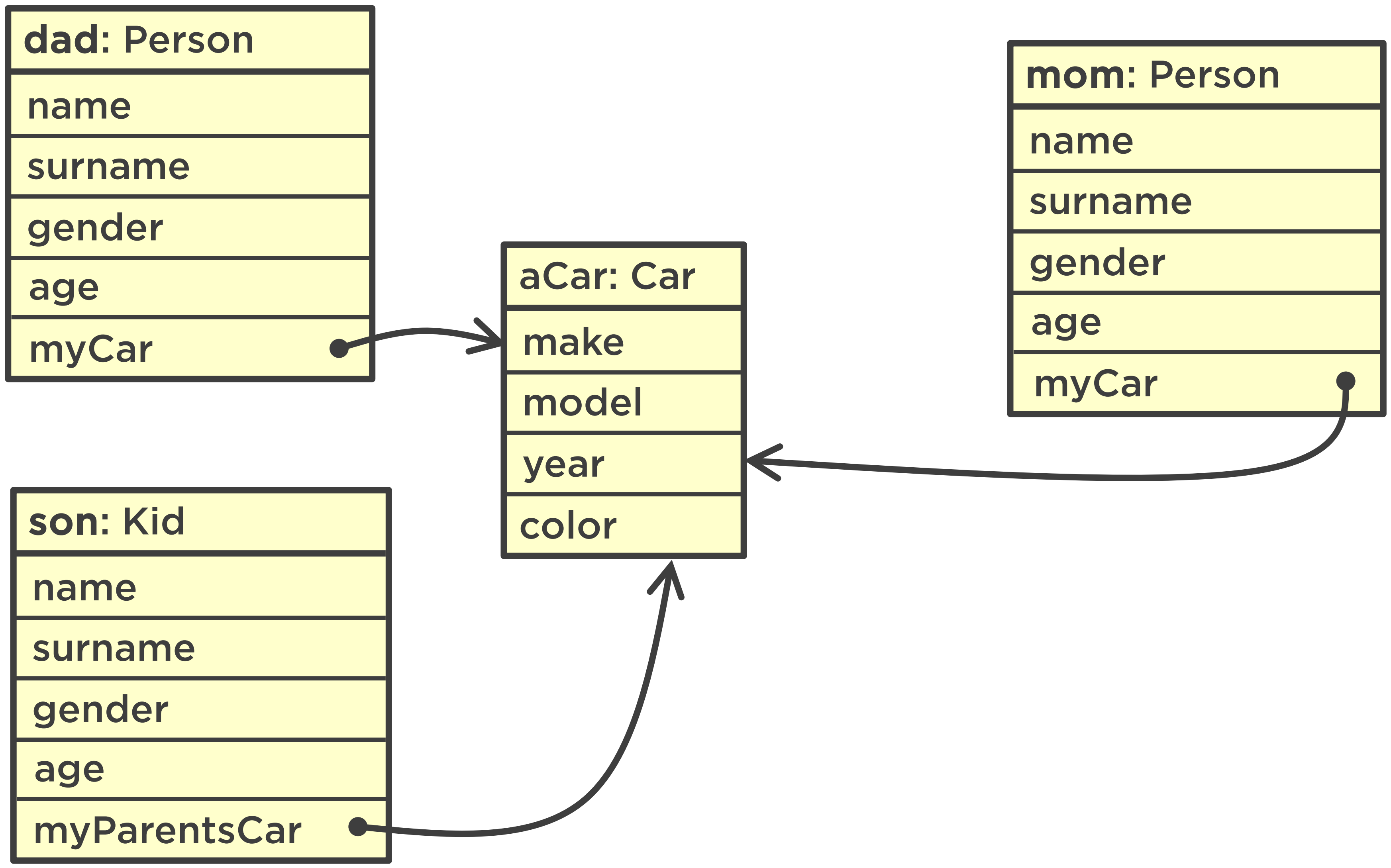
The Dawn of Objects



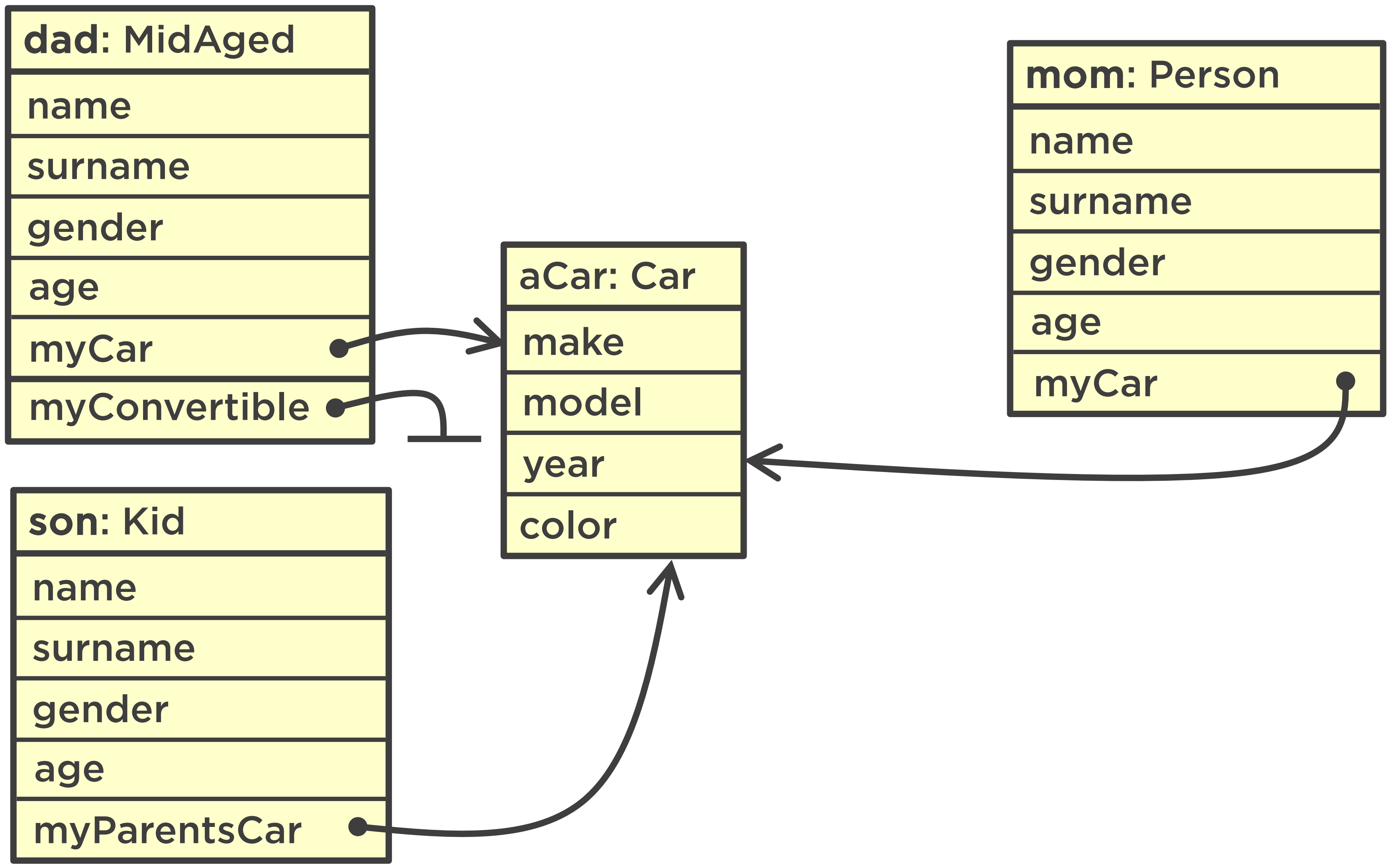
The Dawn of References



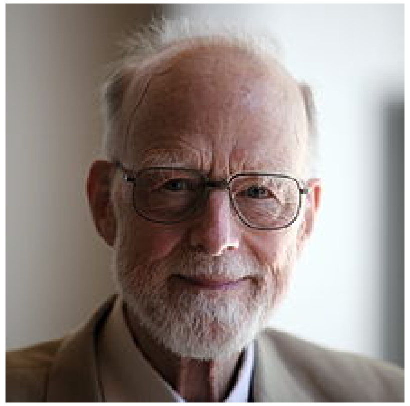
The Dawn of Aliases



The Dawn of Null References



Surviving Null References


Sir Charles Antony Richard Hoare

"I call it my billion-dollar mistake.

It was the invention of the null reference in 1965.
At that time, I was designing the first comprehensive type system
for references in an object oriented language (ALGOL W).

My goal was to ensure that all use of references should be absolutely safe,
with checking performed automatically by the compiler.

But I couldn't resist the temptation to put in a null reference,
simply because it was so easy to implement.

This has led to innumerable errors, vulnerabilities, and system crashes,
which have probably caused a billion dollars of pain and damage
in the last forty years."

Sir Tony Hoare, 2009

Fighting Nulls

Functional languages define a special type for potentially missing objects

- Option – Scala, OCaml, F#, Java…
- Maybe – Haskell, Idris, …

Option either contains a value or contains no value

- But Option is never null

What Follows Next

A short example in F#

- Demonstration of the Option type

Grow a similar type in C#

Collection can be used as Option in C#

- Contains one element if value is present
- Empty if value is not present

The collection idea leads to invention of Option type in C#

Summary

Optional call on an object (C#)
Option<T> type

```
return
    this.userRepository
    .Find(username)
    .Select(user => user.Balance)
    .DefaultIfEmpty(0)
    .Single();
```

Call on an optional object (C#)
if-then-else

```
IUser user = this.userRepository.Find(username);
if (user != null)
    return user.Balance;
return 0;
```

```
match getPrice itemName with
| Some(price) -> sprintf "You can have %s for $%f" itemName price
| None -> sprintf "We don't sell %s" itemName;;
```

Optional call on an object (F#)
Pattern matching

Collections
Map-Reduce
Sequences
Option<T>

In the following module:
Service Locator Pattern