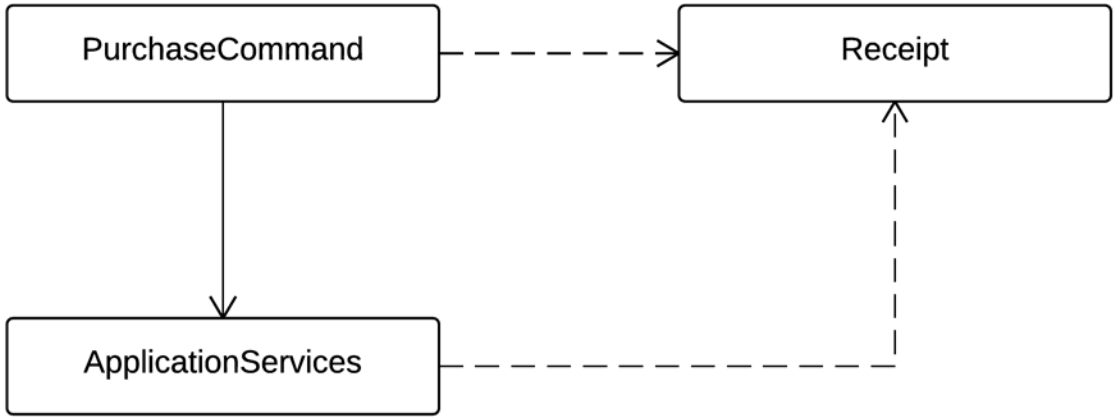
Null Object and Special Case Patterns

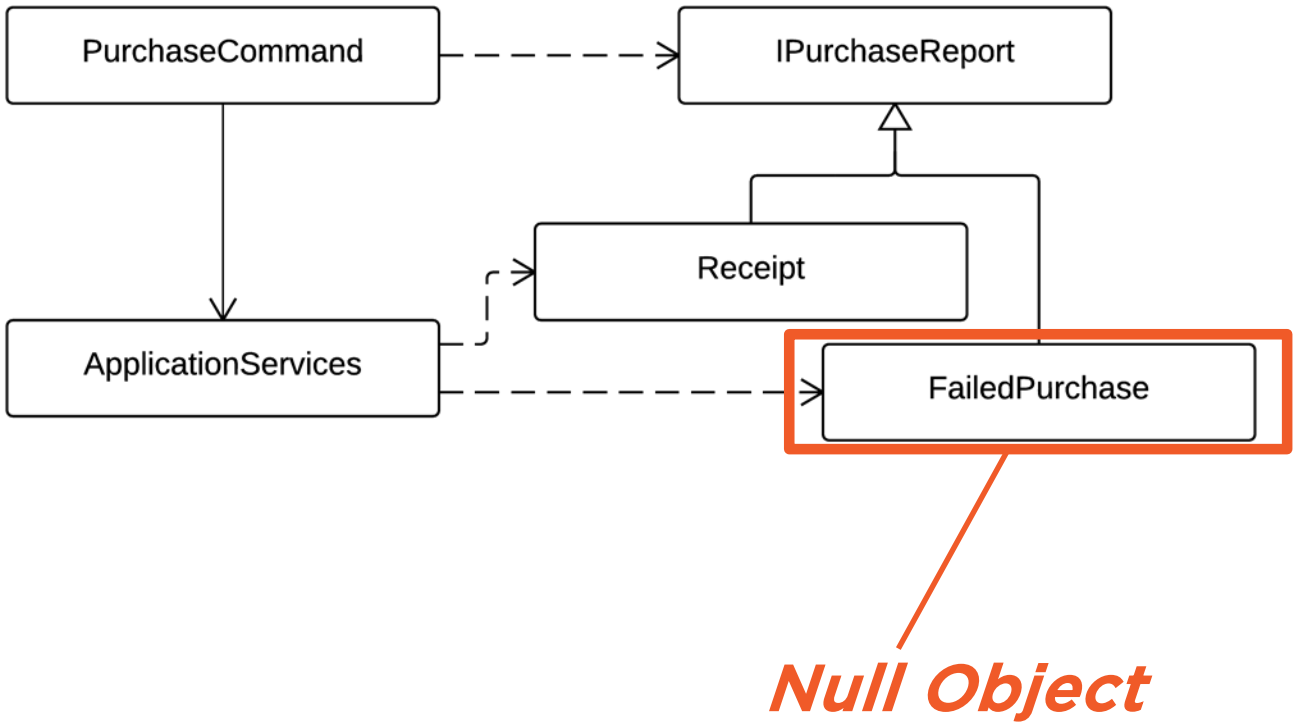
Zoran Horvat
CTO at InterVenture GmbH

@zoranh75 www.codinghelmet.com

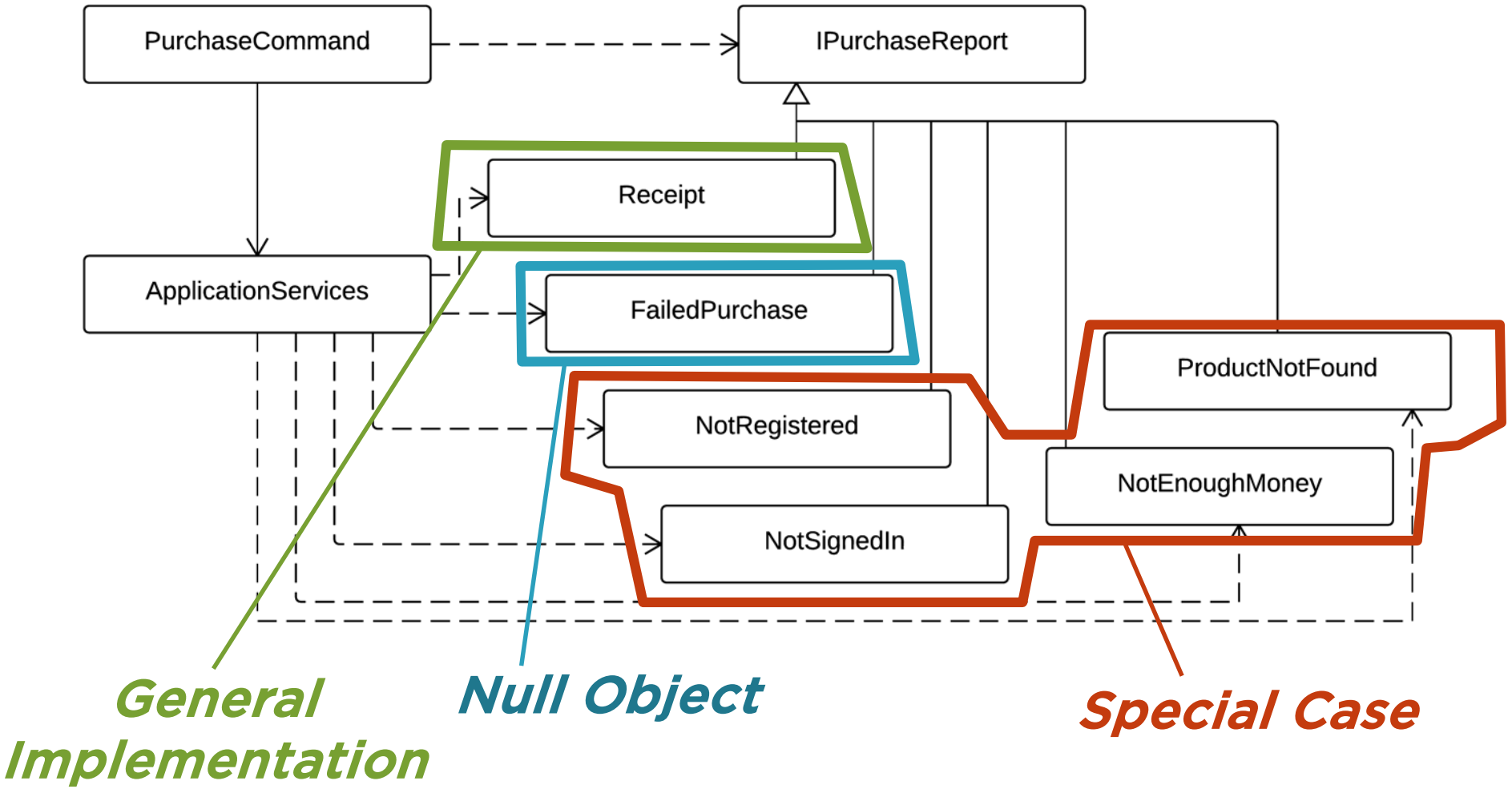
Null Object Pattern



Null Object Pattern



Special Case Pattern

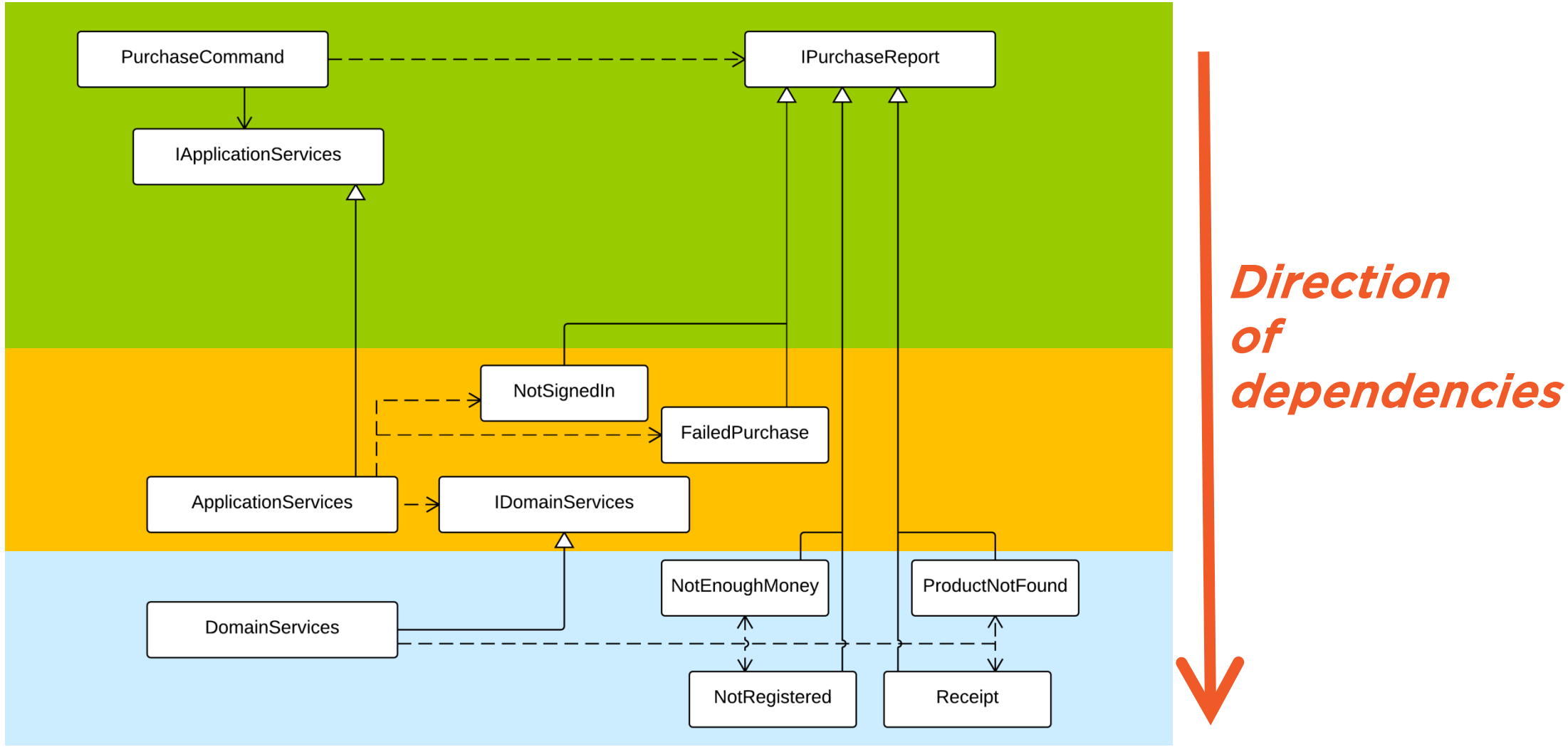


Null Object vs. Special Case

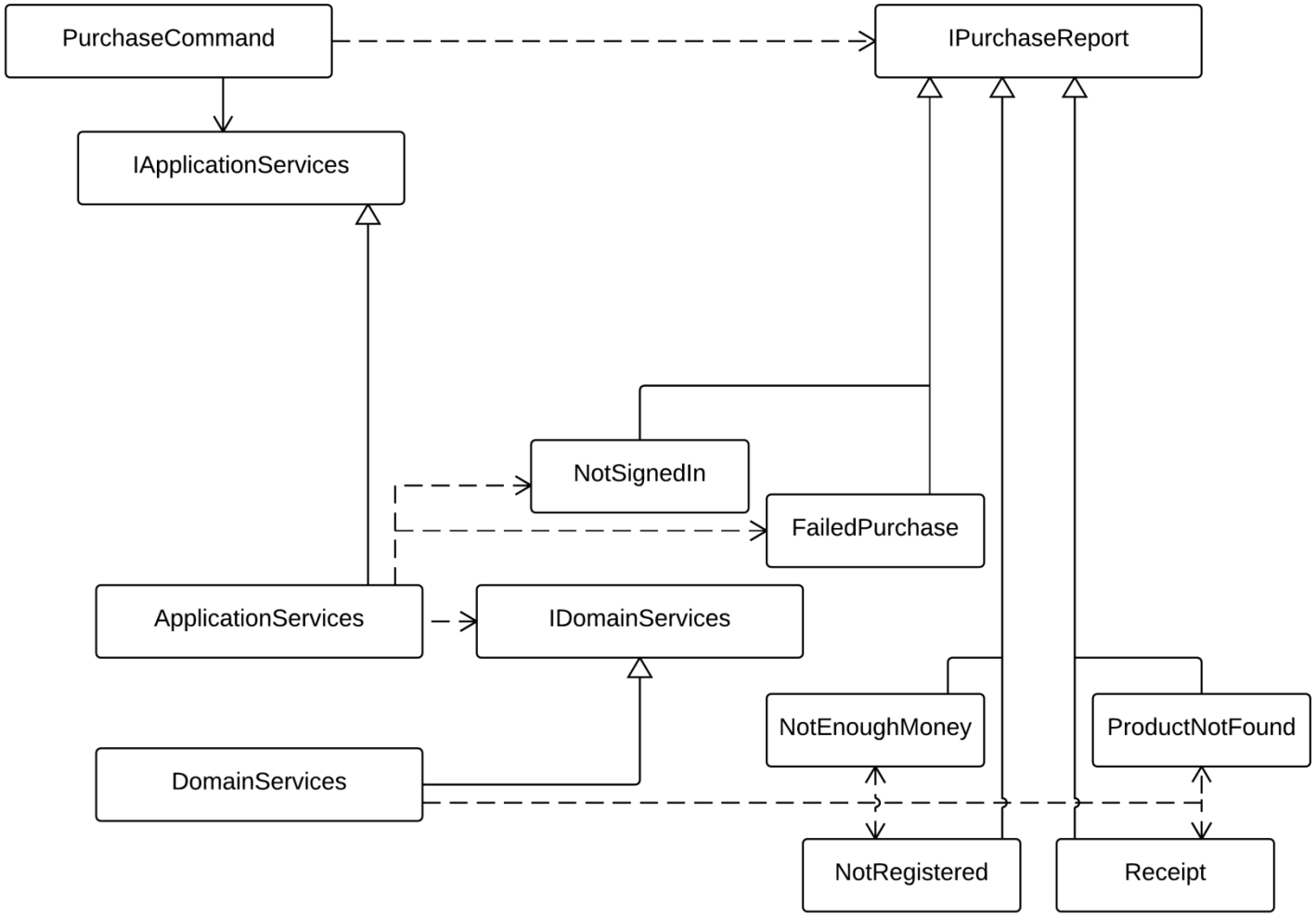
Electricity company dispatching bills

- Nonexistent customer
  - Nobody lives there, the bill won't be paid
  - Null Object
- Unknown person
  - Somebody rents the place
  - We don't care to know his name, but he will pay the bill
  - Special Case
- Foreign citizen
  - The person doesn't have SSN
  - We don't care to track this person's data, but he will pay the bill
  - Special Case

Where to Place Concrete Implementation?



Where to Place Concrete Implementation?

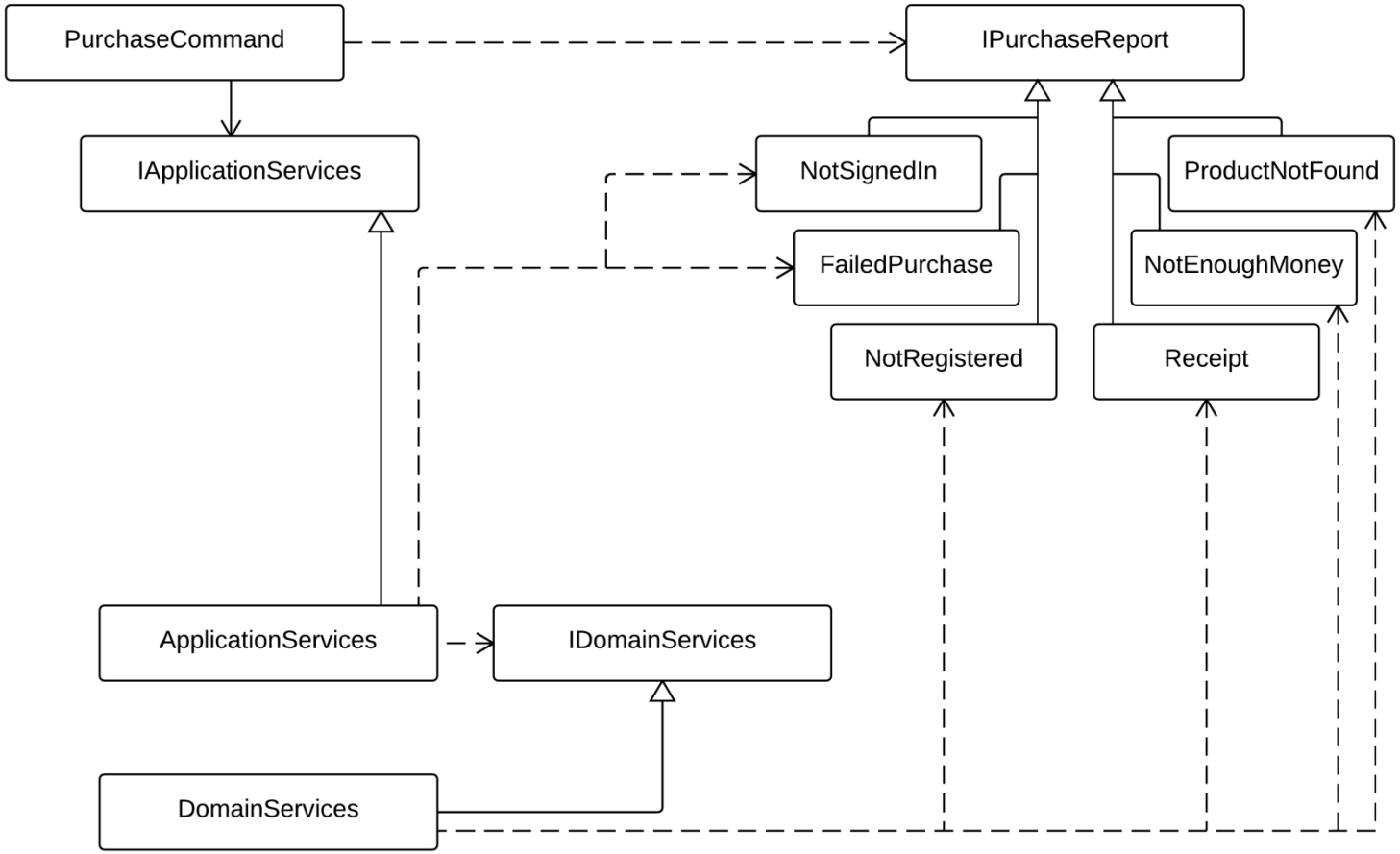


Dependency Inversion Principle:

High-level modules should not depend on low-level modules. Both should depend on abstractions.

Abstractions should not depend on details. Details should depend on abstractions.

Where to Place Concrete Implementation?

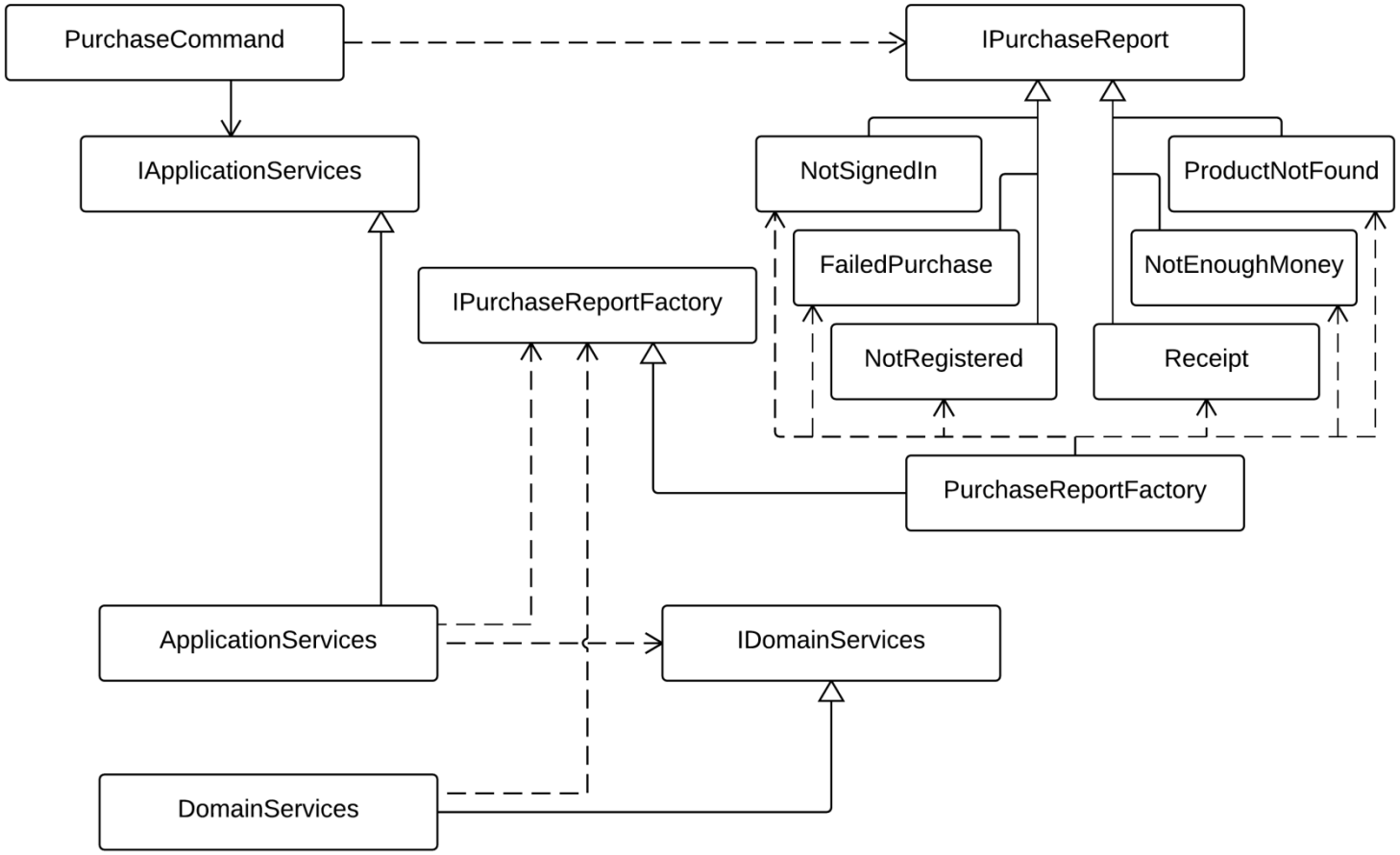


Dependency Inversion Principle:

High-level modules should not depend on low-level modules. Both should depend on abstractions.

Abstractions should not depend on details. Details should depend on abstractions.

Where to Place Concrete Implementation?

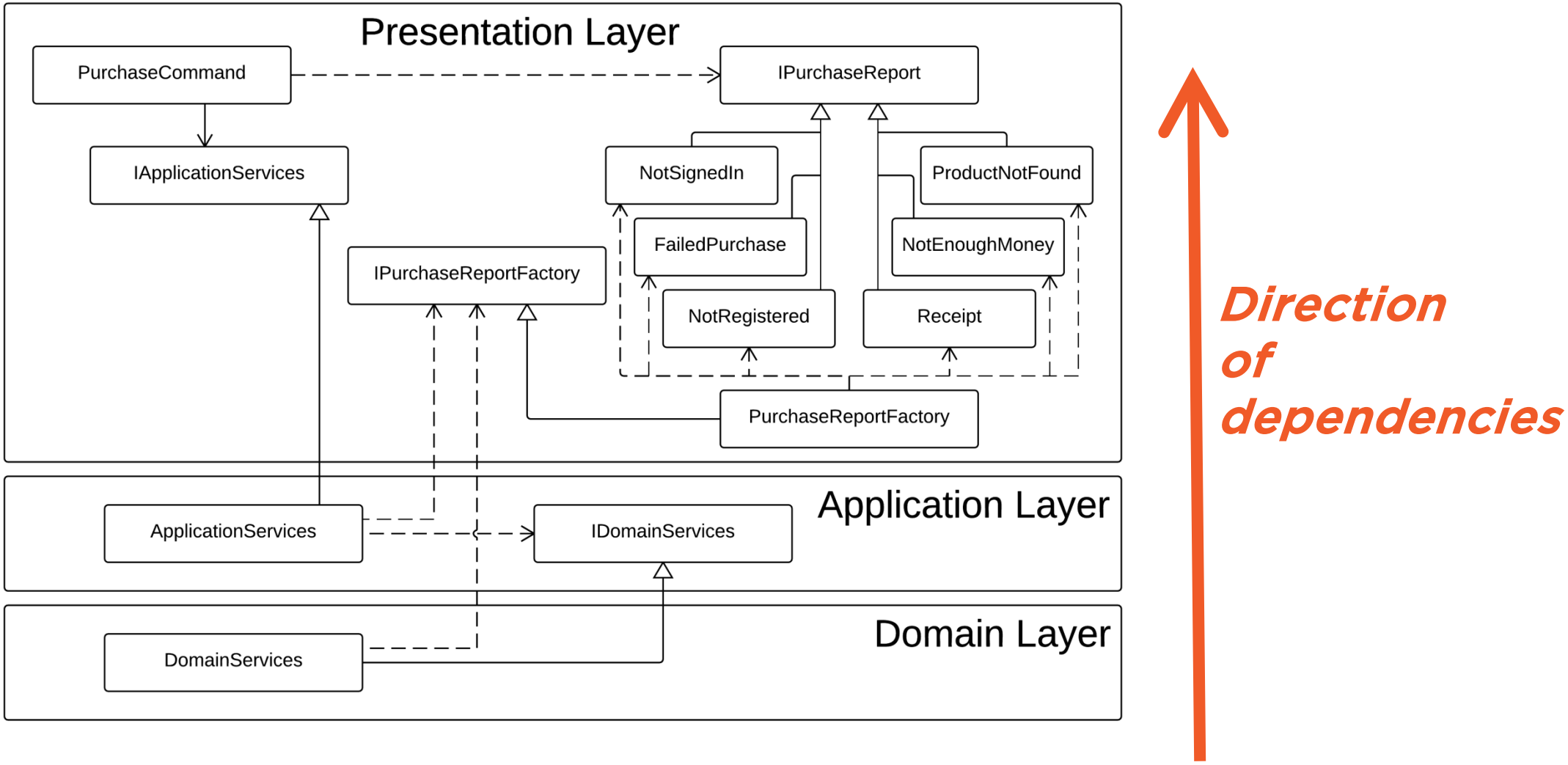


Dependency Inversion Principle:

High-level modules should not depend on low-level modules. Both should depend on abstractions.

Abstractions should not depend on details. Details should depend on abstractions.

Where to Place Concrete Implementation?



Summary

Using Null Object and Special Case to remove null references

Simplifies control flow on the client

- No branching when null reference is received

Special Case carries additional information back to the client

Control flow is flat

- There is no branching
- All operations flow through the same sequence of steps
- Information required to complete operation is carried by the returned object

Summary

The layering problem

- Introduced abstract factory

Factory class hides concrete implementations

- This reduces coupling between modules (layers in this case)
- All modules only depend on abstractions

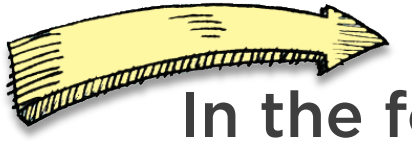
What Comes Next?

Working with collections

- Introducing LINQ to Objects simplifies control flow

When Null Object and Special Case are not applicable...

- There is no valid object to return from a method
- Do not fall back to using null references!
- There are better solutions that rely on collections



In the following module: Map-Reduce pattern in domain logic