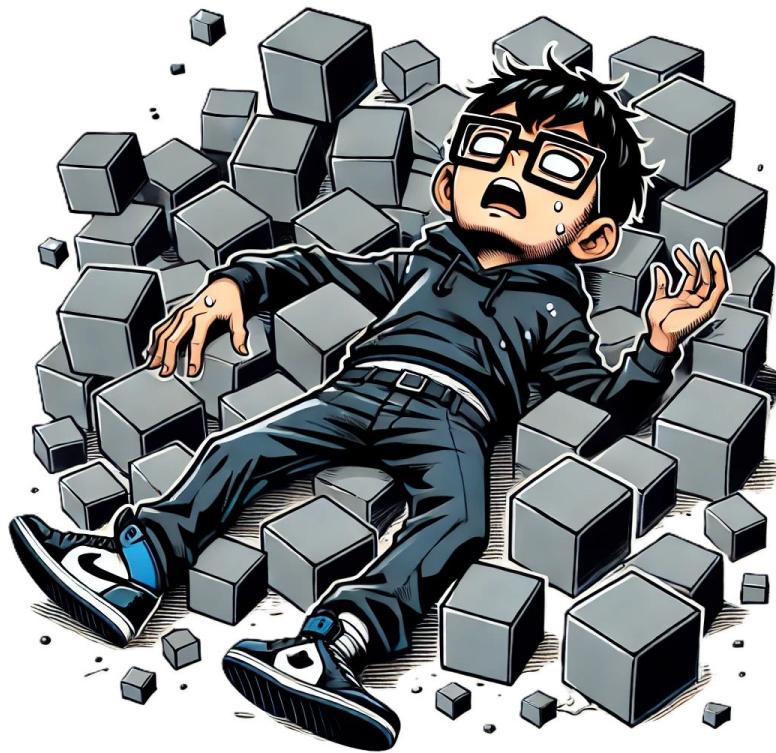


# J'écris des tests sans pleurer maintenant



## Jérémy Sorant







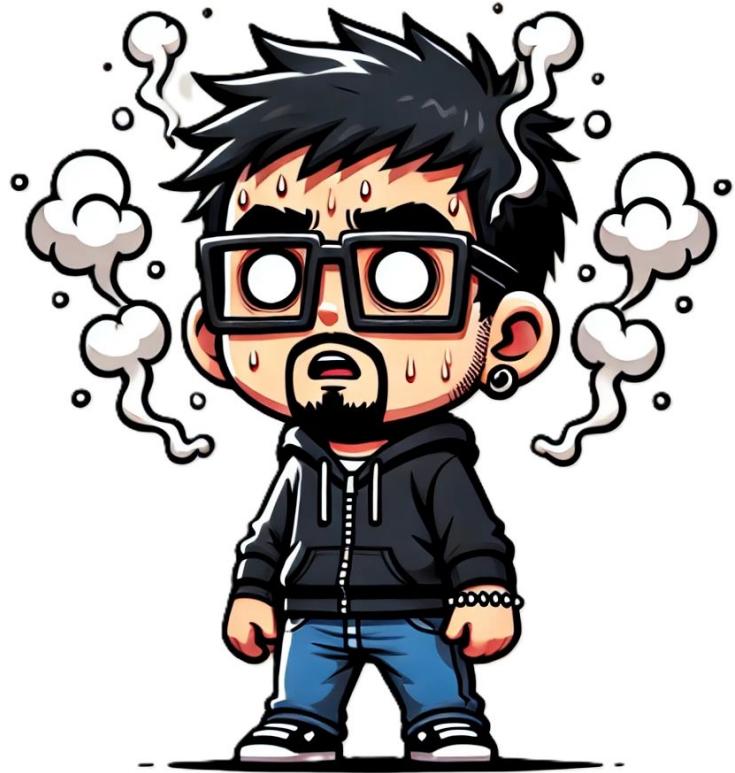












# Mes débuts avec les tests

Problèmes :

- ⇒ Code difficilement testable
- ⇒ Tests lents
- ⇒ Tests flacky
- ⇒ Tests fragiles
- ⇒ Tests difficiles à maintenir



# Les games changers

Axes :

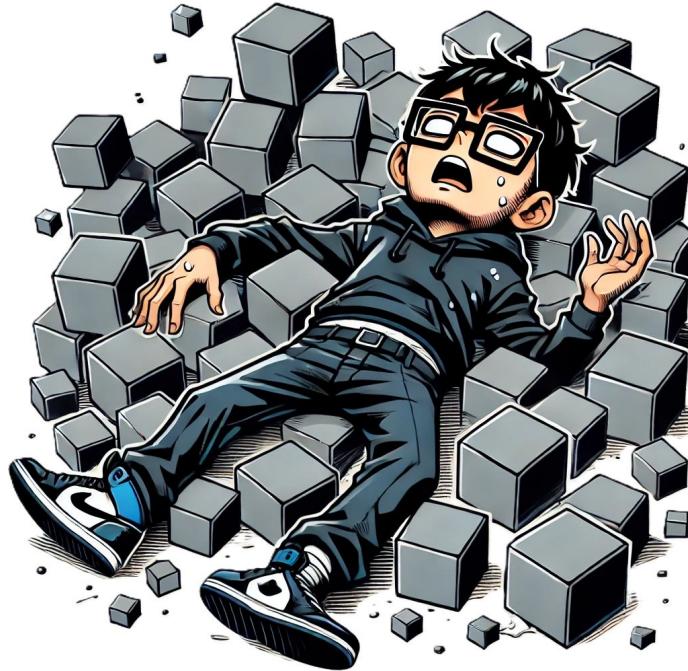
- ⇒ Fragilité
- ⇒ Lenteur
- ⇒ Maintenabilité / Lisibilité



# Fragilité



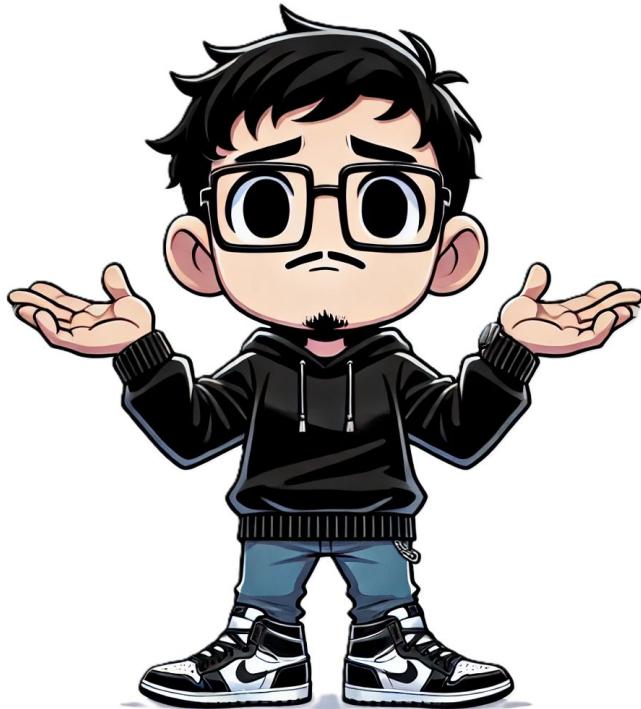
# Problème



# Perte de temps



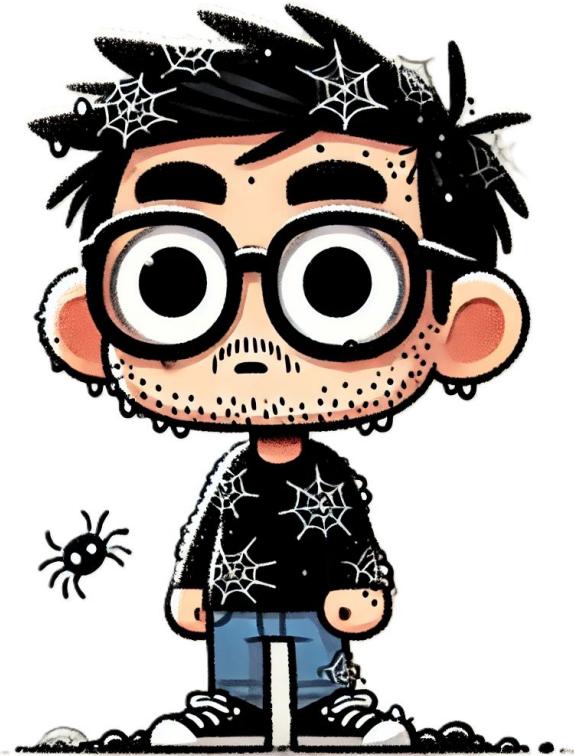
# Hésitation à refactorer



# Contournements



# Dégradation continue



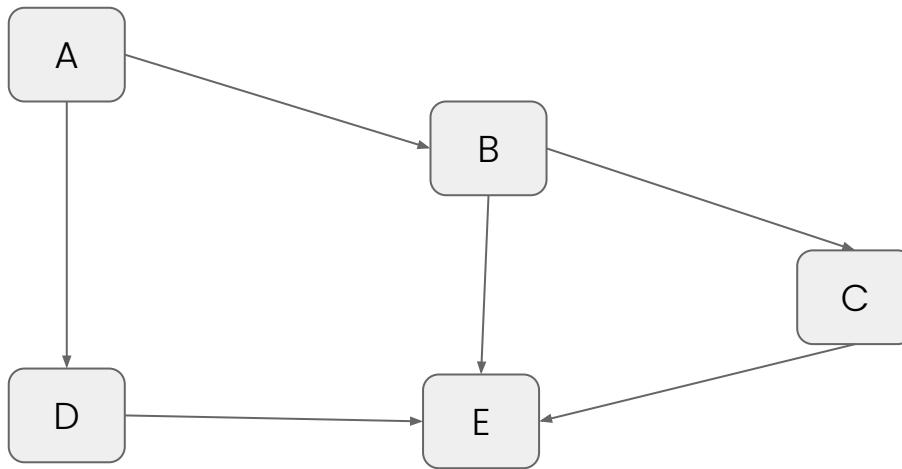
# Tristitude

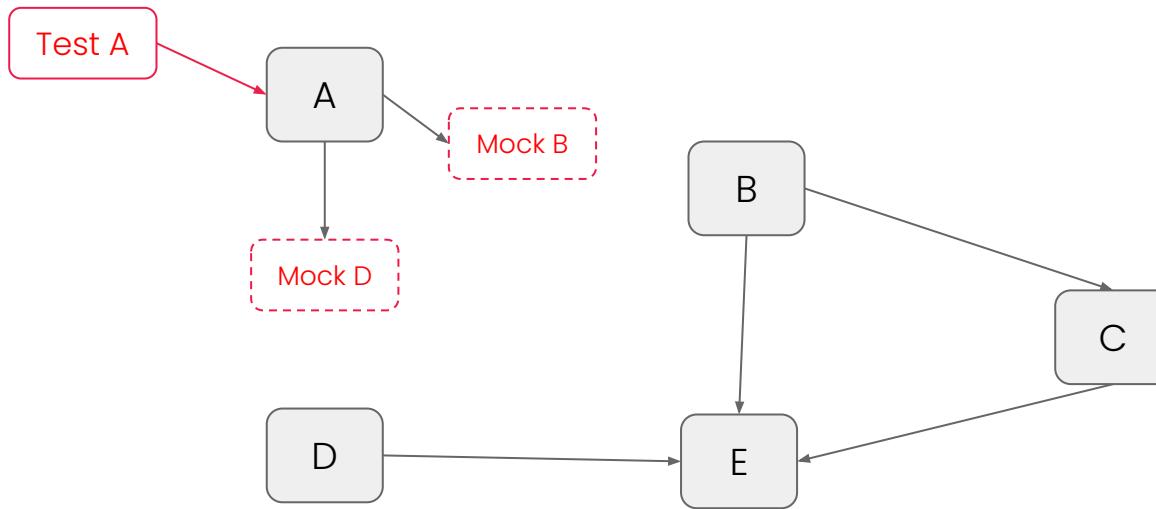


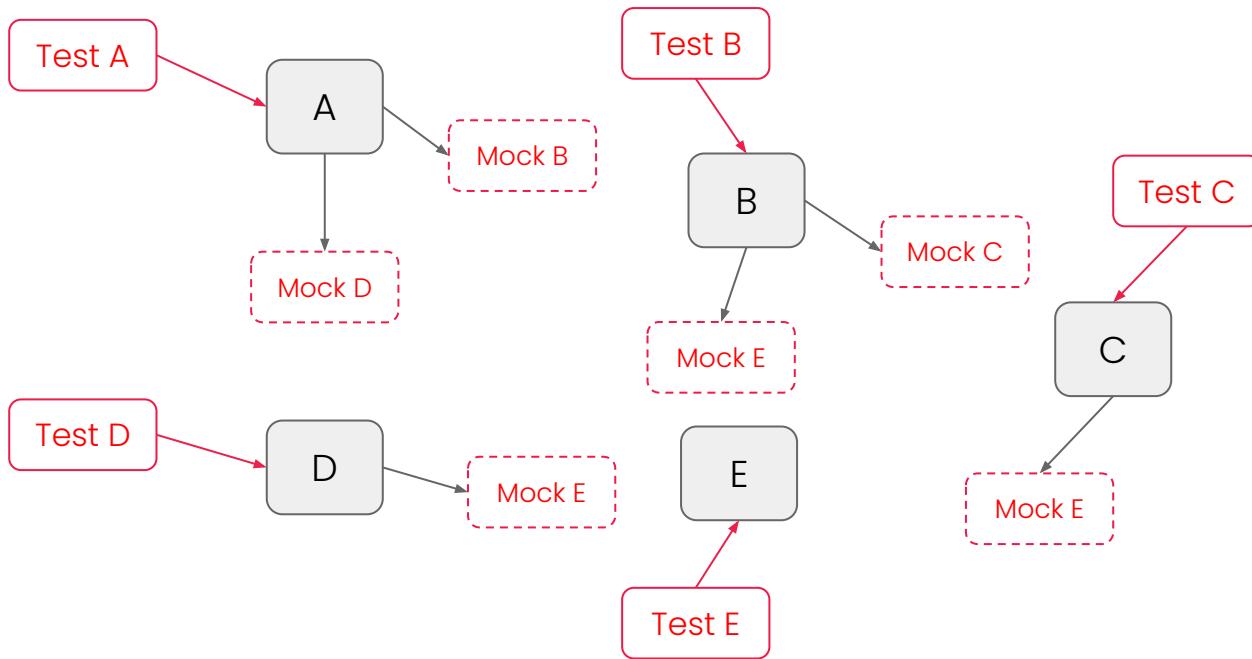
# Cause

Unitaire = une classe, une fonction

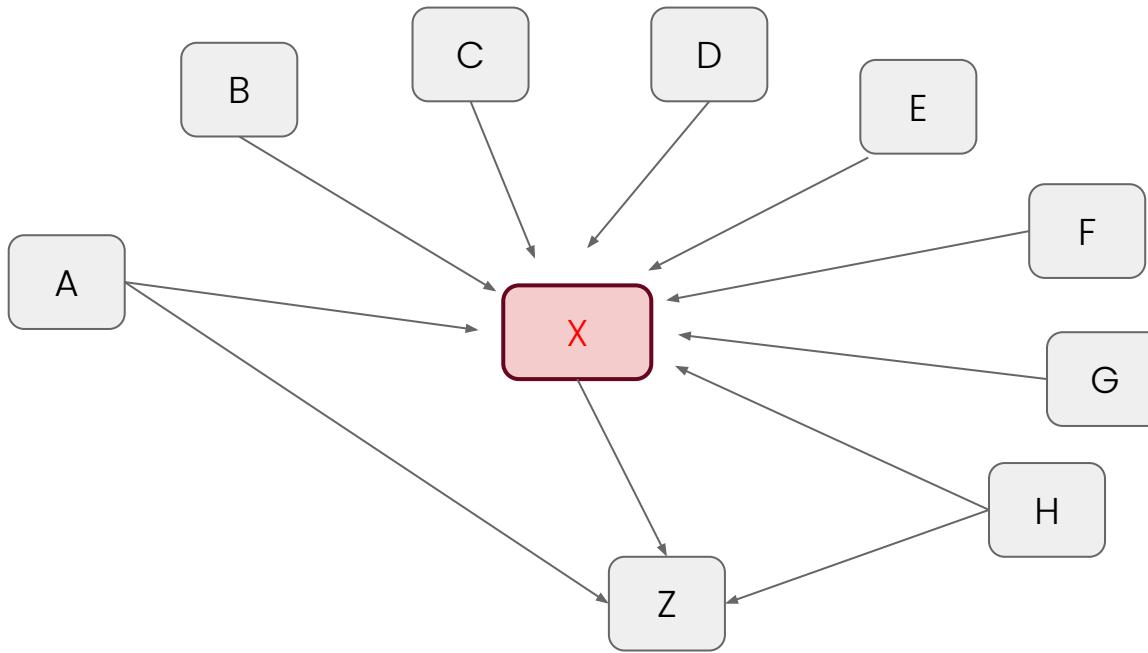




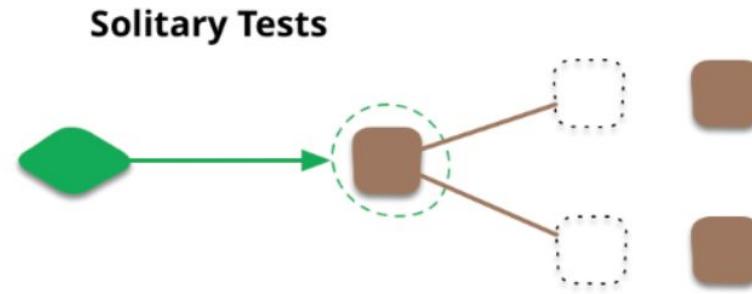
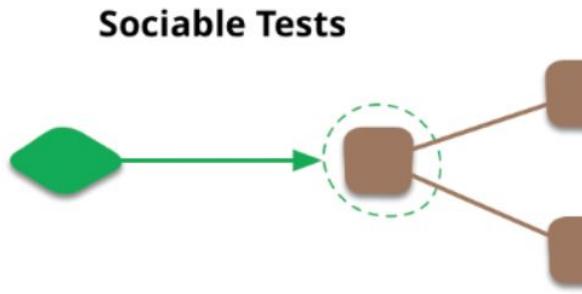




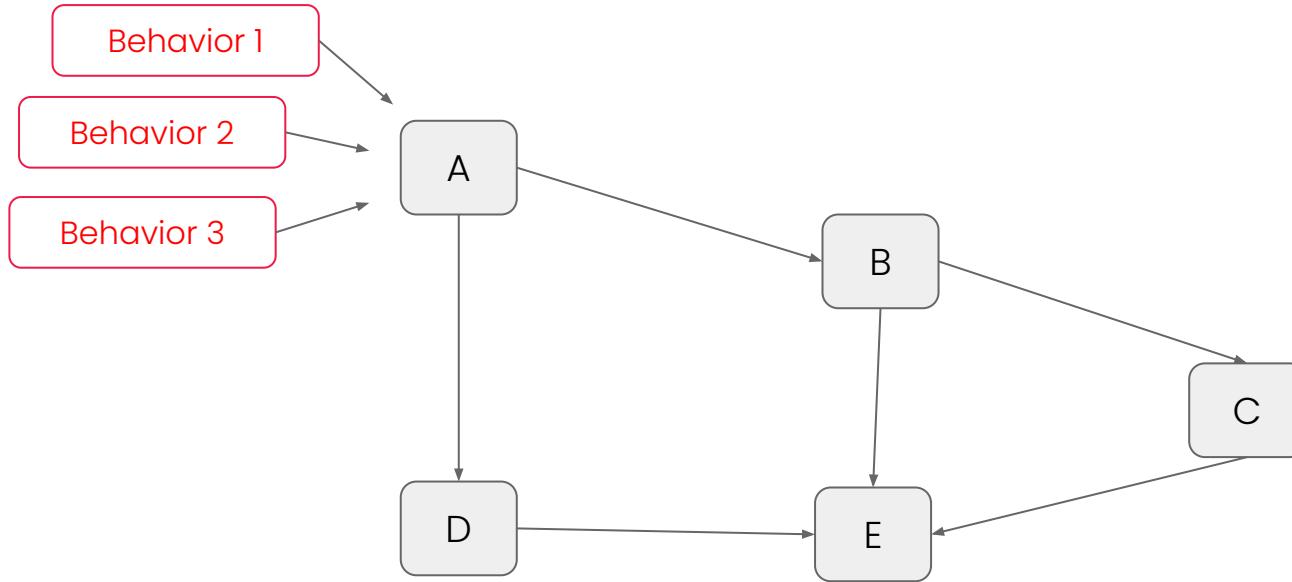
# Couplage excessif

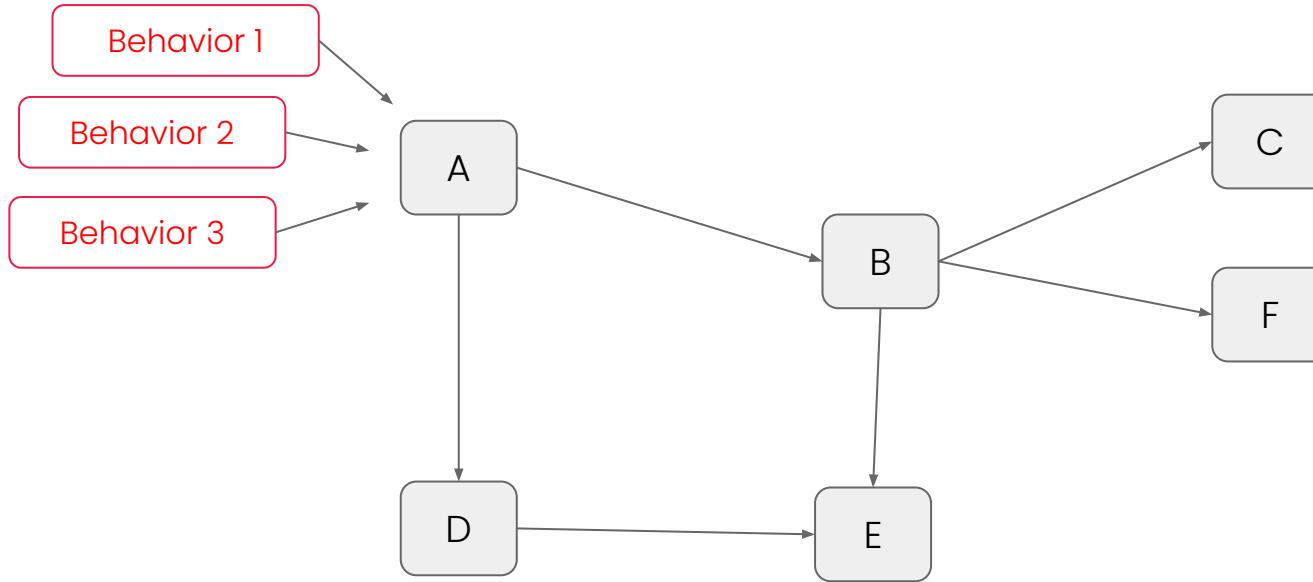


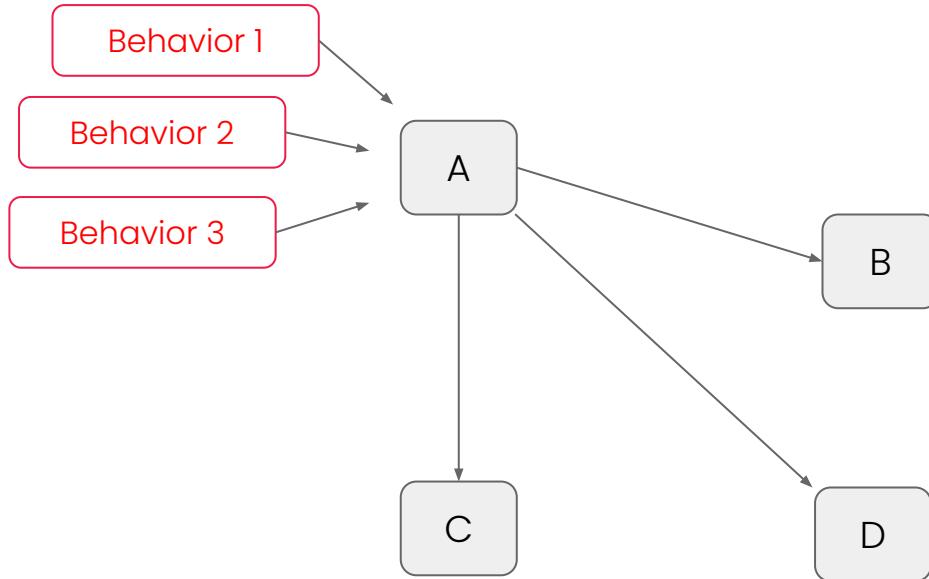
# Test social / traversant



- Martin Fowler







# Comportement vs implémentation

Le comportement est stable

L'implémentation est instable



# Exemple

Etant donné

Compte A  
1000 €

Compte B  
1000 €

Quand

Virement A => B  
100 €

Alors

Compte A  
900 €

Compte B  
1100 €

# Exemple

Ajouter / retirer  
un montant

Ajouter une  
transaction

Calculer la valeur  
depuis des  
transactions

# Connecteur BDD

Vérifier le contenu d'une table

```
@Test
void shouldSaveAndGetBook() {
    AnyBookRepository repository = new AnyBookRepository();

    repository.save(harryPotter());

    Optional<Book> retrievedBook = repository.get(harryPotter().id());

    assertThat(retrievedBook).contains(harryPotter());
}
```

# Bénéfices

- ⇒ Ne casse que si le comportement change
- ⇒ Facilite le refactoring
- ⇒ Peut servir de documentation





# Lenteur



# Problème

- Mon application est difficile à tester
  - ⇒ Obligé de tester avec la base de données
  - ⇒ Tests e2e
  - ⇒ Vérification manuelle



# Teste moins souvent



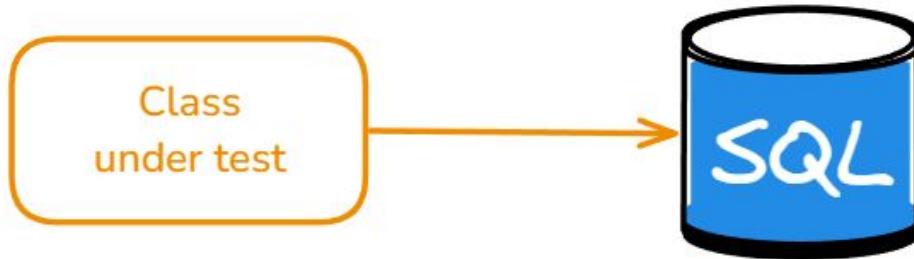
# Perte de temps



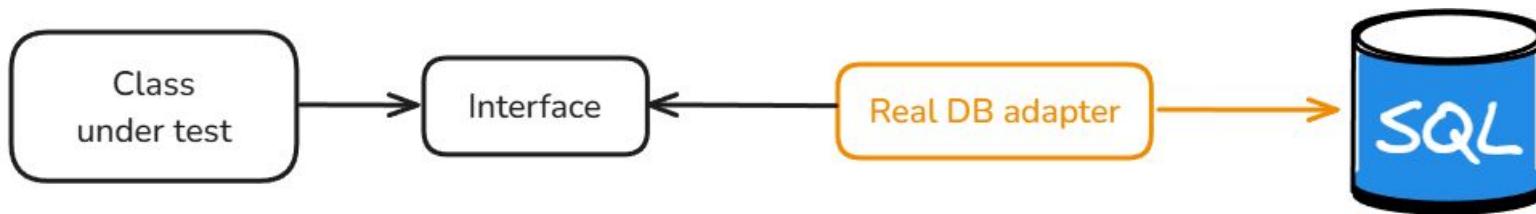
# Tristitude



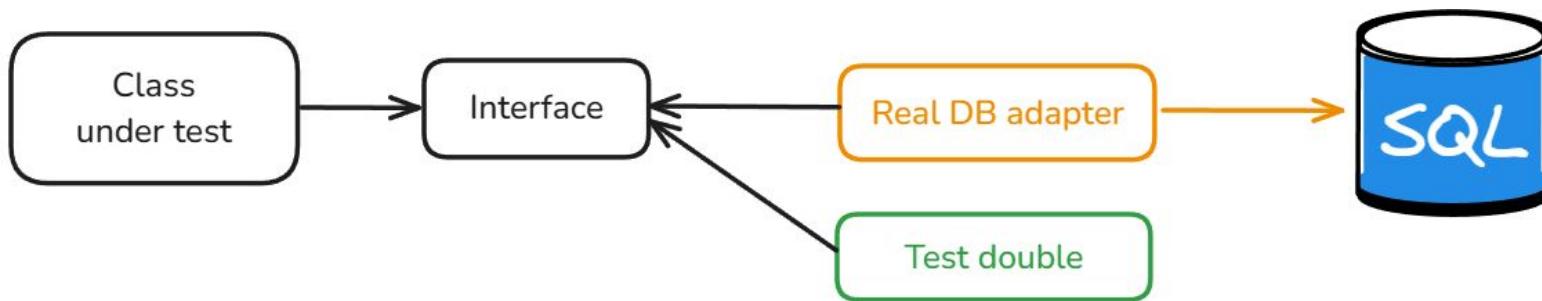
# Cause



# Dependency Inversion Principle



# Dependency Inversion Principle



# Feedback <3

✓ Borrow a book	82 ms
✓ should return book borrowed event when book is borrowed	76 ms
✓ should throw when book is not owned by the library	2 ms
✓ should throw when book is already borrowed	2 ms
✓ should throw when borrower has already four books borrowed	2 ms



# Compromis

Surface  
couverte

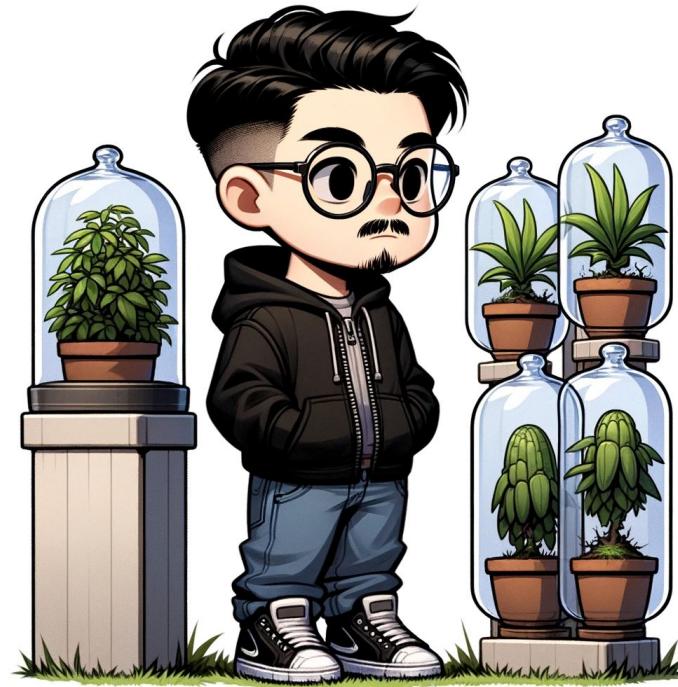


Rapidité

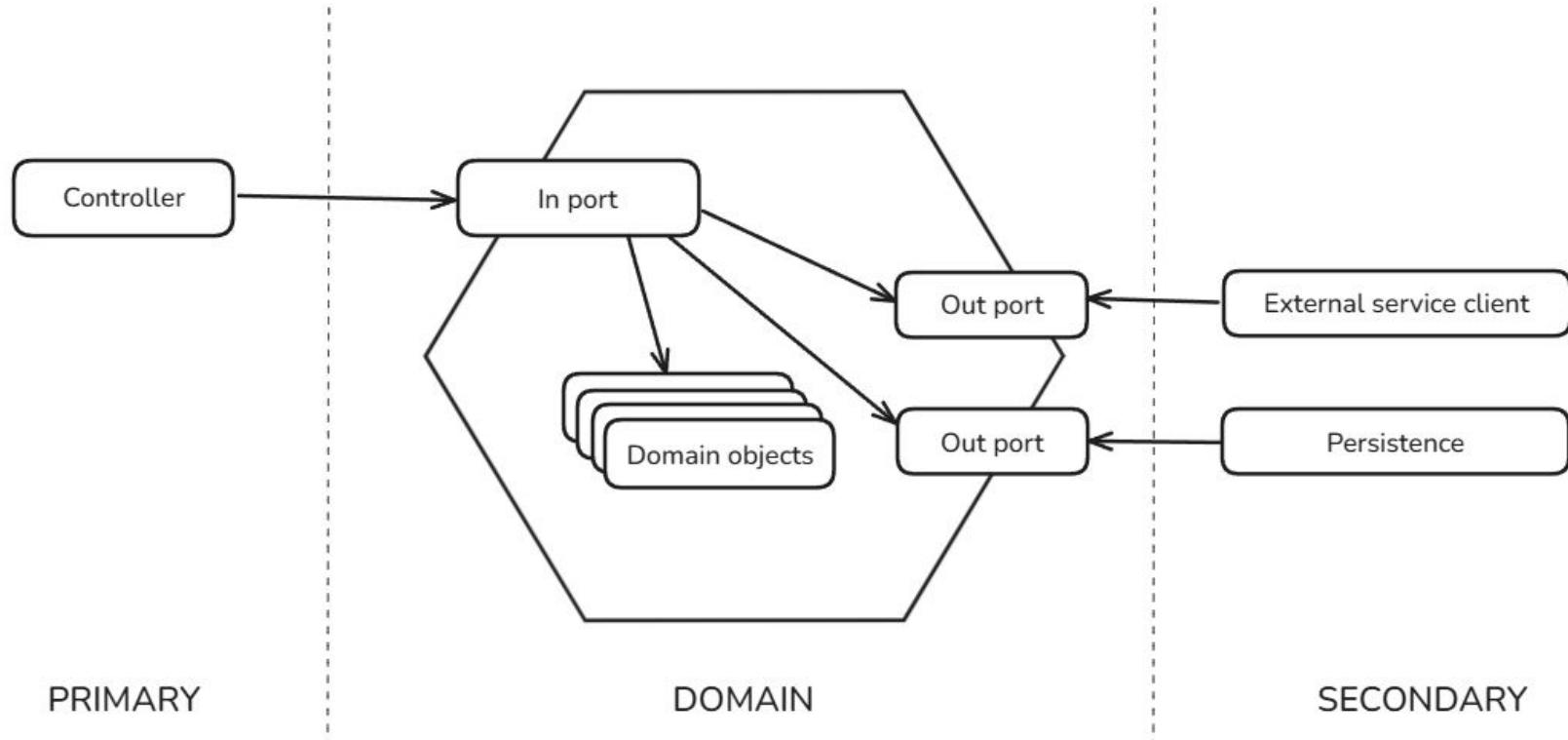
**Testez aussi les vraies implémentations  
et la tuyauterie !**

# Architecture testable

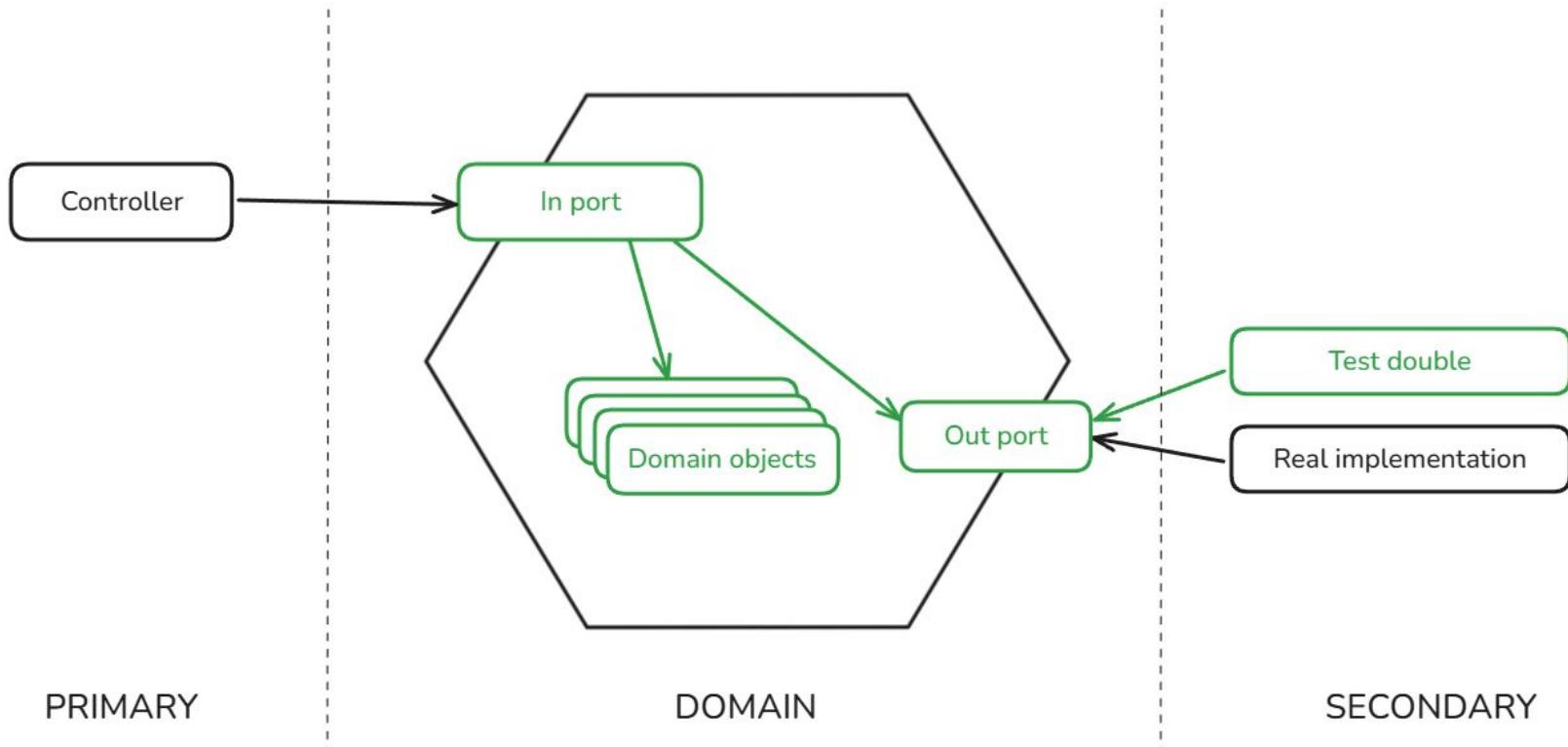
- ⇒ Clean architecture
- ⇒ Architecture hexagonale



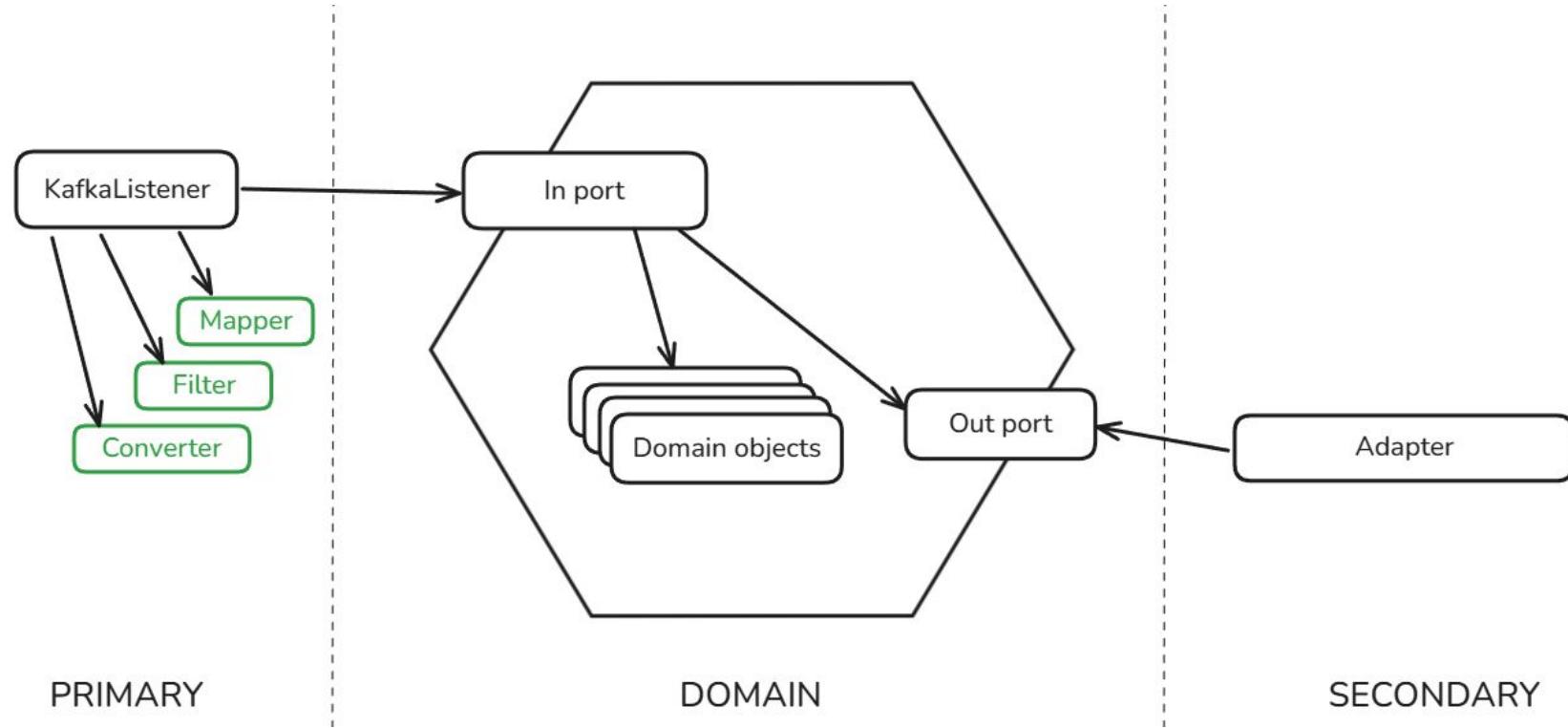
# Architecture hexa



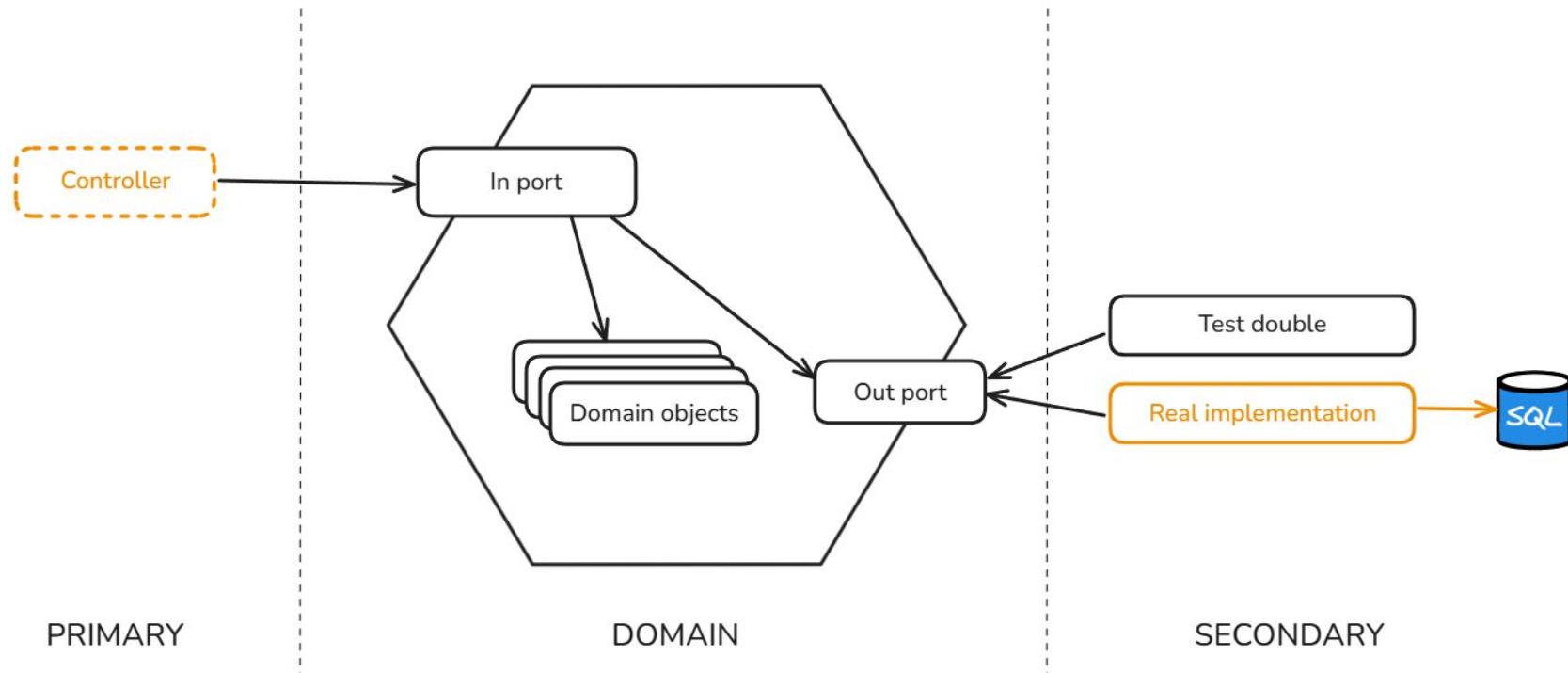
# Acceptation



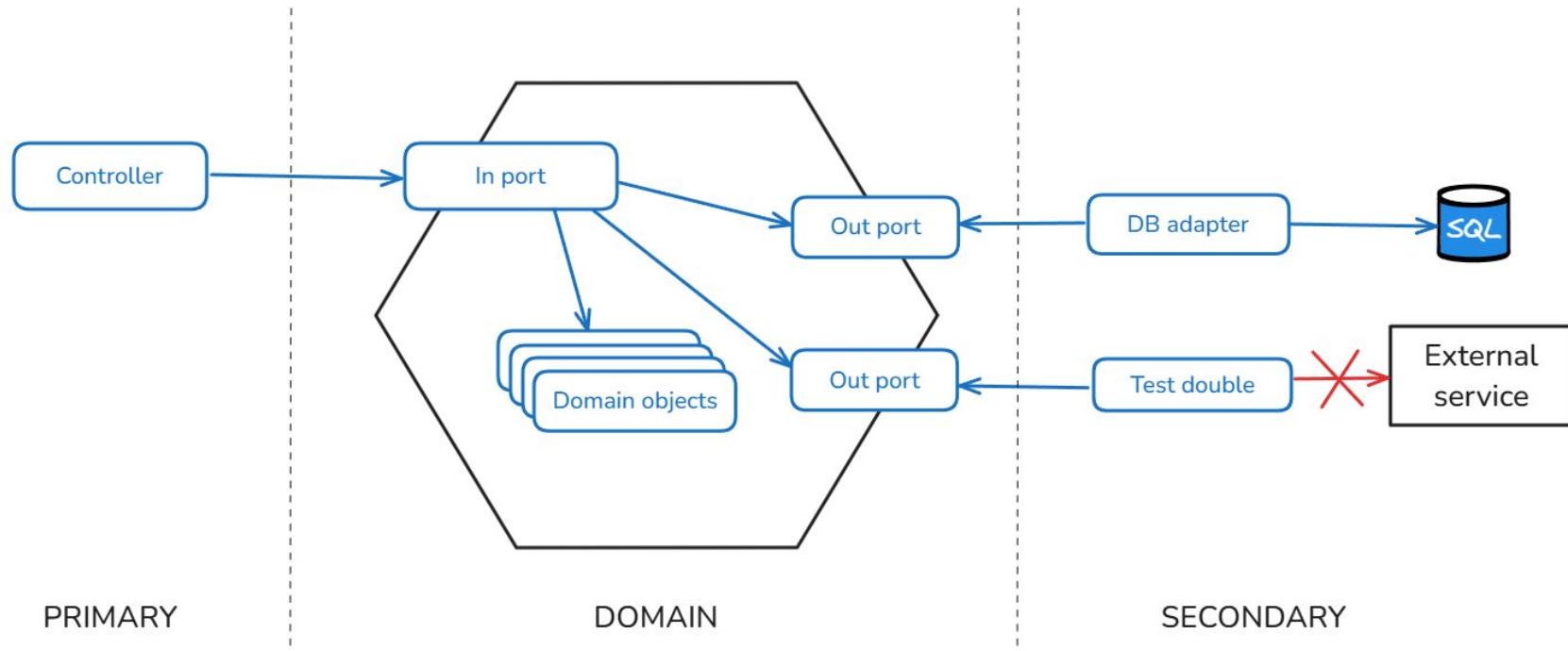
# Unitaire



# Intégration



# Composant



# Résumé

- ⇒ Architecture testable
- ⇒ Stratégie adaptée





# Maintenabilité



# Problèmes

- ⇒ J'ai du mal à comprendre l'intention du test
- ⇒ Je ne comprends pas l'erreur retournée par mon test
- ⇒ Je ne comprends pas le rapport entre mon test et le métier



# Charge cognitive



# Perte de temps

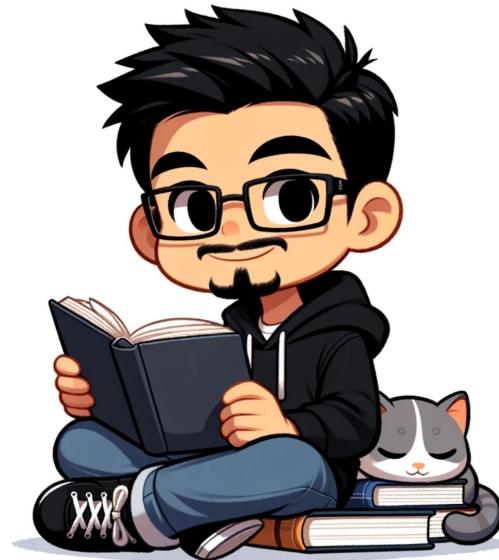


# Tristitude

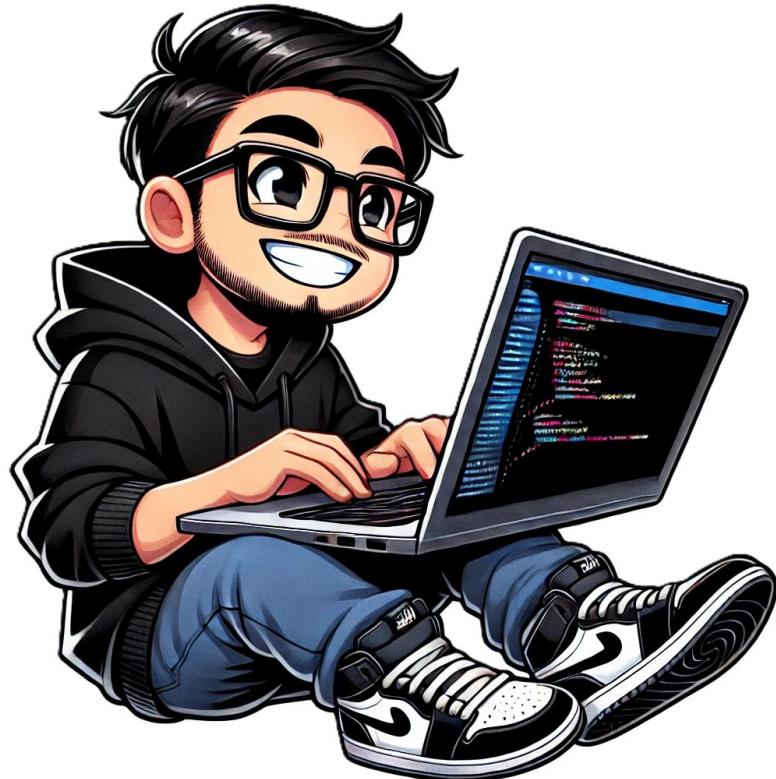


# Exemple

Gérer les emprunts de livres dans une bibliothèque



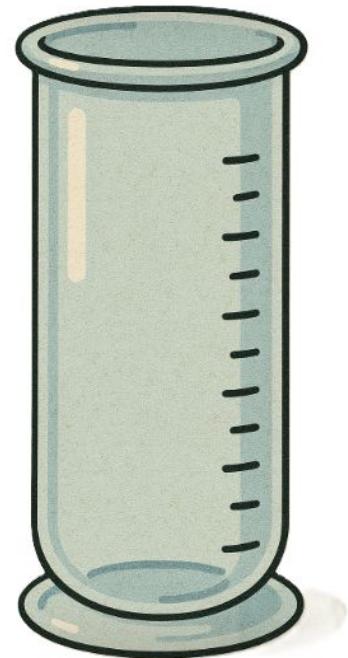
# Live coding - Maintenabilité



# Résumé

Révéler l'intention

- ⇒ Nomenclature adaptée, langage métier
- ⇒ GWT / AAA + une seule règle par test
- ⇒ Cacher les détails d'implémentation
  - ⇒ Extract / Fixture / Builder
- ⇒ Utiliser les outils du framework de test
- ⇒ Custom assertions
- ⇒ Messages d'erreur explicites



# Takeaways



# Takeaways

- ⇒ Testez des comportements plutôt que des implémentations
- ⇒ Optez pour une architecture qui facilite l'écriture des tests
- ⇒ La maintenabilité des tests est aussi importante que celle du code de prod
- ⇒ TDD peut vous permettre de voir le dev sous un angle totalement différent





# CRAFTING SUSTAINABLE CODE WE DO



## Jérémie Sorant

Développeur chez SHODO Lyon

Co-organisateur des Software Crafters Lyon



[linkedin.com/in/jeremy-sorant](https://www.linkedin.com/in/jeremy-sorant)



[github.com/jsorant](https://github.com/jsorant)



# CRAFTING SUSTAINABLE CODE WE DO



**Code & slides**

 [linkedin.com/in/jeremy-sorant](https://linkedin.com/in/jeremy-sorant)  
 [github.com/jsorant](https://github.com/jsorant)



**Feedback**

COACHING . CRAFT . JUSTICE SOCIALE



 shodo.io

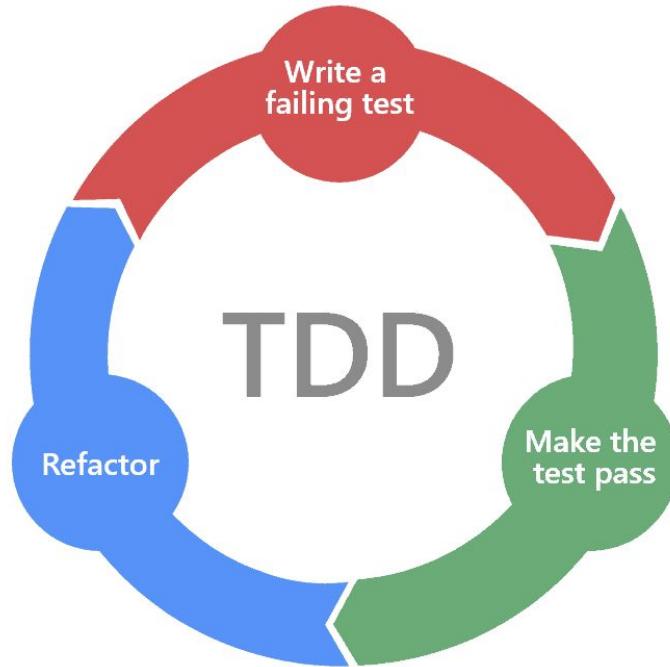
 SHODOioFR

SHODO

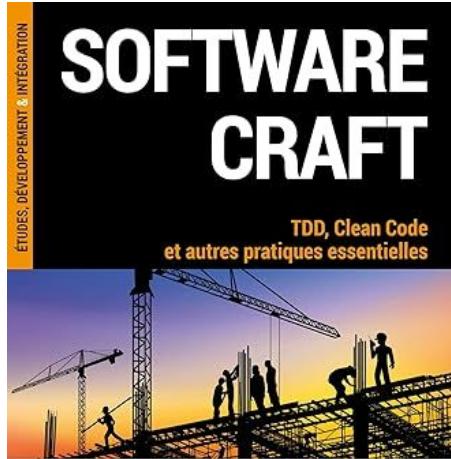
# Annexes

# TDD

- ⇒ Aide à se concentrer sur le besoin
- ⇒ Produit du code testé et couvert à 100%
- ⇒ Limite l'over engineering
- ⇒ Aide à découper le problème
- ⇒ Apporte de la sérénité
- ⇒ Méthode de dev plus fun

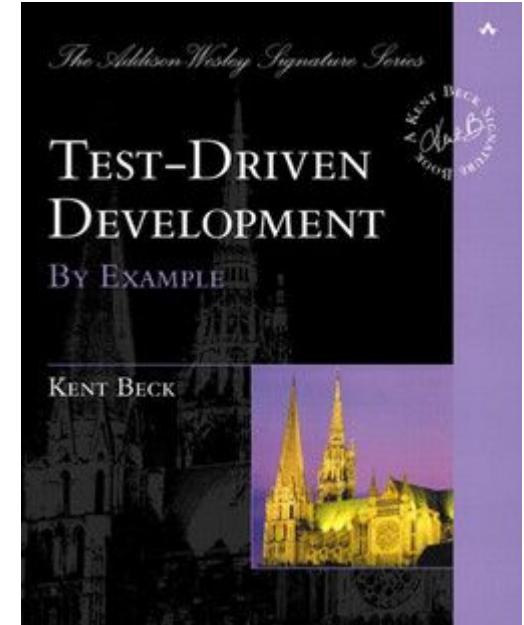
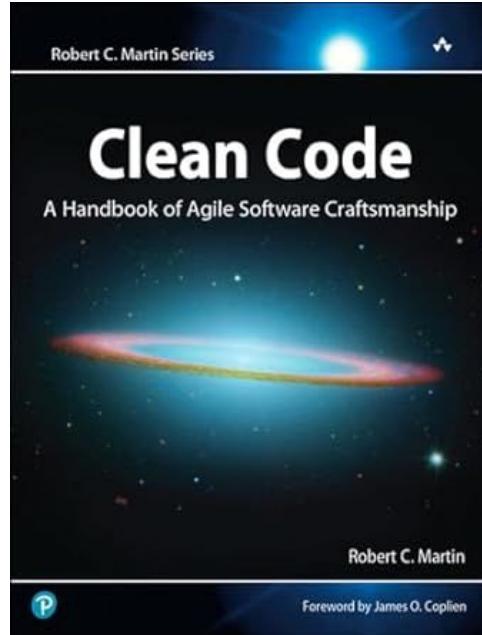


# Lectures



Cyrille Martraire  
Arnaud Thiéfaine  
Dorra Bartagüiz  
Fabien Hiegel  
Houssam Fakih

DUNOD



# Comportement vs implementation

Unit Test

Martin Fowler

<https://martinfowler.com/bliki/UnitTest.html>

TDD, Where Did It All Go Wrong

Ian Cooper

<https://www.youtube.com/watch?v=EZ05e7EMOLM>

Outside-in Diamond pour écrire des tests Antifragiles & orientés métier

Thomas Pierrain

<https://www.youtube.com/watch?v=09R8ROv3aKU>

# Hexa & Tests

Architecture Hexagonale : Comment bien écrire ses tests ?

Julien Topçu

<https://www.youtube.com/watch?v=4vBJAN3ttkc>

Architecturoplastie hexagonale d'un backend Node.js

Jordan Nourry, Adrien Joly, Julien Topçu

<https://www.youtube.com/watch?v=r2XMwAUqZBA>

# AssertJ - Fluent assertions

<https://assertj.github.io/doc/>

# Custom assertions

<https://assertj.github.io/doc/#assertj-core-custom-assertions>

<https://www.baeldung.com/assertj-custom-assertion>

# Tests front

Tester son interface efficacement (front)

Manon Carbonel & Jérémy Chauvin

<https://dev.to/singebob/tester-son-interface-efficacement-des-patterns-pour-des-tests-d-robustes-2402>

# Données de test

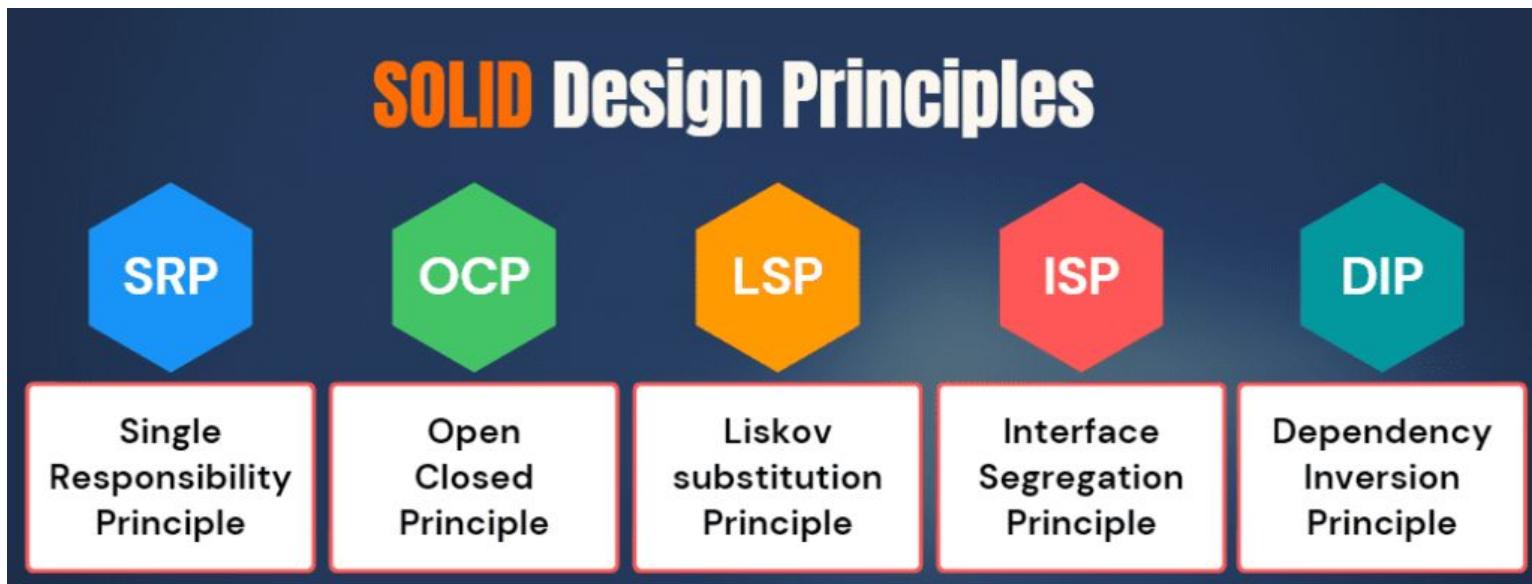
Mieux tester en optimisant la génération de données de test : Panorama des patterns

Manon Carbonel & Jérémy Chauvin

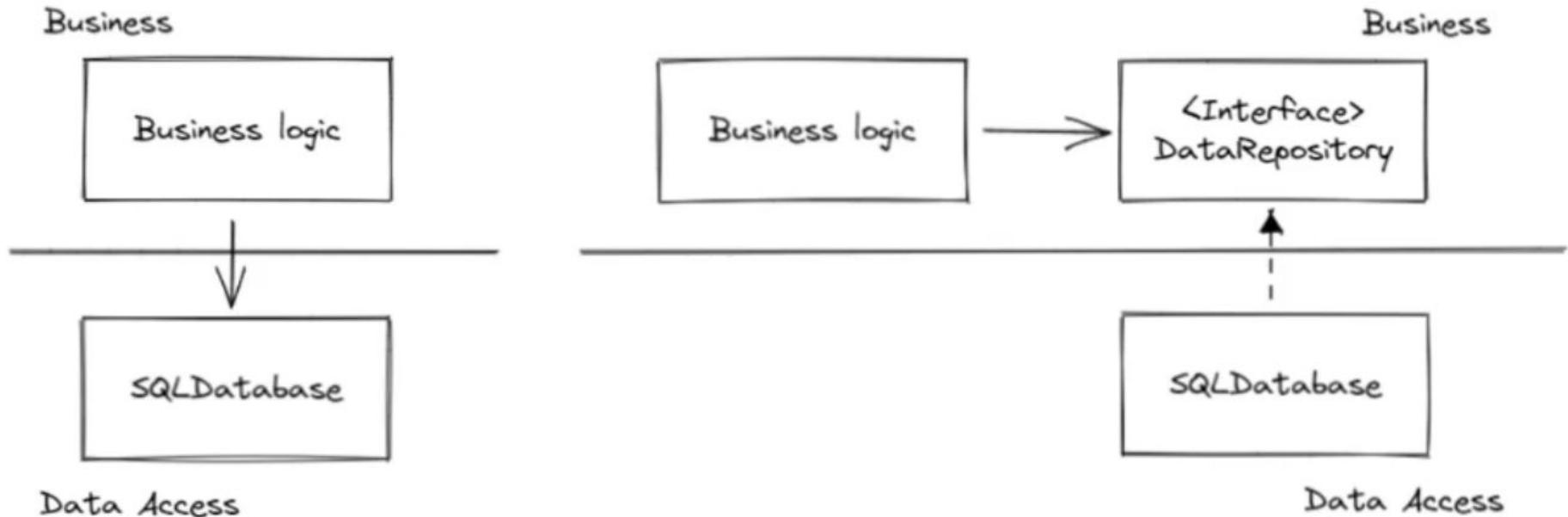
<https://dev.to/singebob/mieux-tester-en-optimisant-la-generation-de-donnees-de-test-panorama-des-patterns-1cde>

# SOLID

<https://blog.cleancoder.com/uncle-bob/2020/10/18/Solid-Relevance.html>

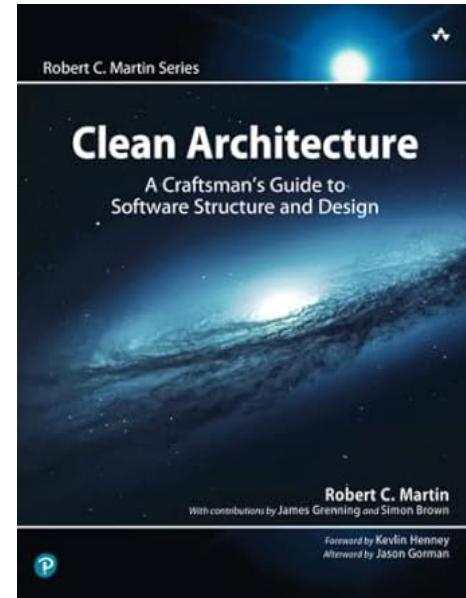
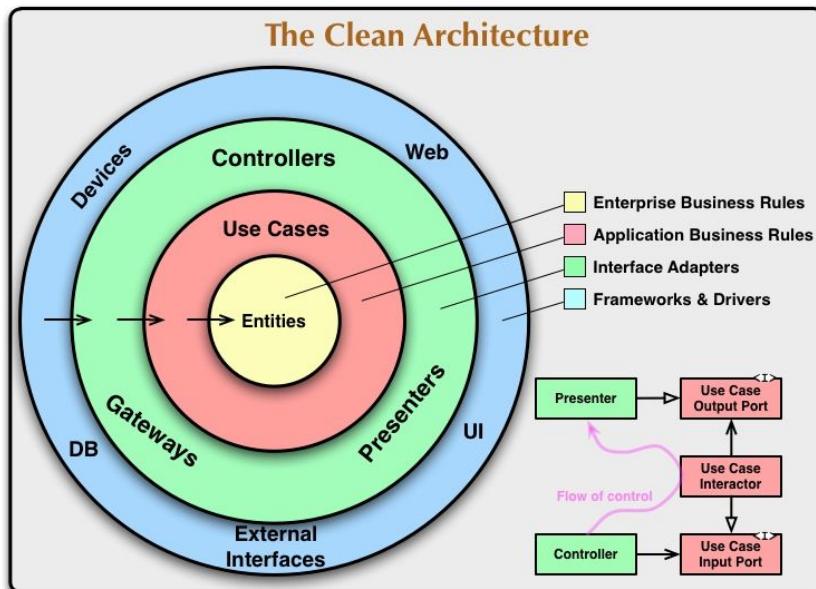


# Dependency Inversion Principle



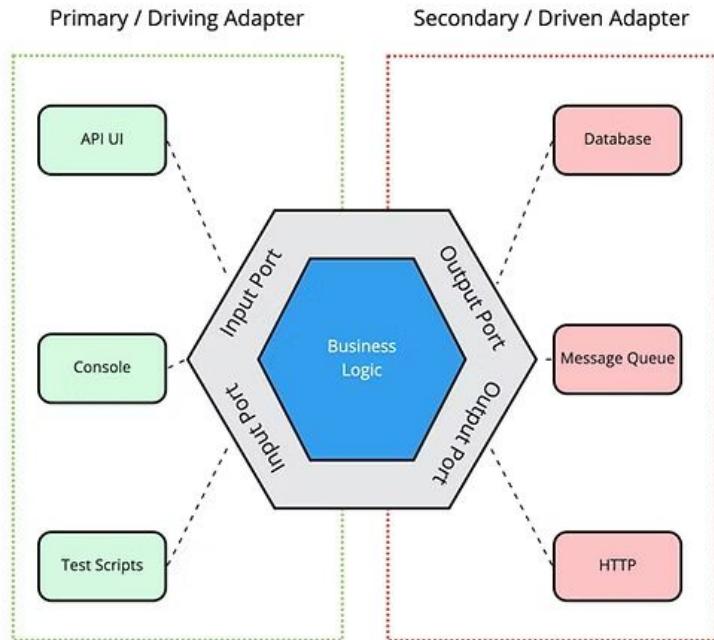
# Clean architecture

<https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>

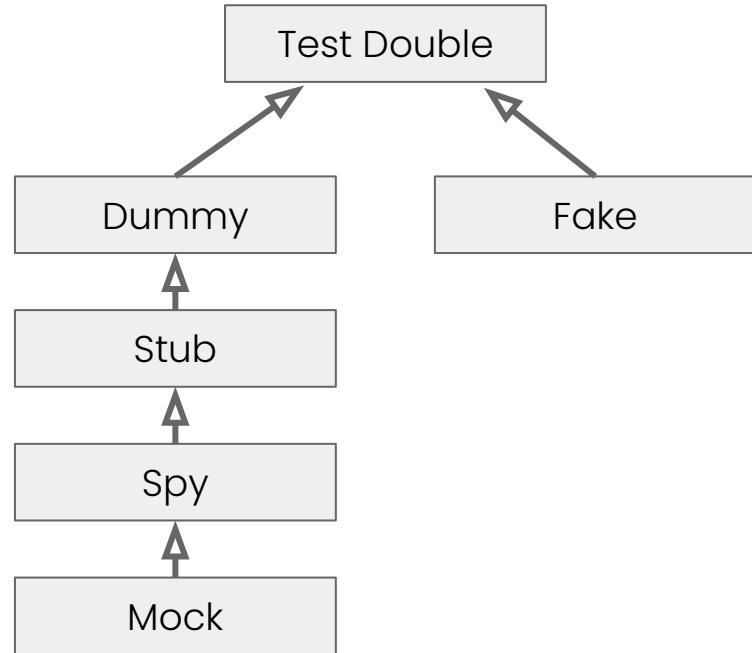


# Architecture hexagonale

<https://alistaircockburn.com/Hexagonal%20Budapest%202023-05-18.pdf>



# Doublures de test



# Doublures de test

<https://blog.cleancoder.com/uncle-bob/2014/05/14/TheLittleMocker.html>

<https://www.martinfowler.com/articles/mocksArentStubs.html>

# Contrat d'adapters

