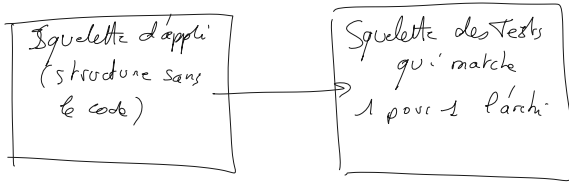
TDD

- Agile
-

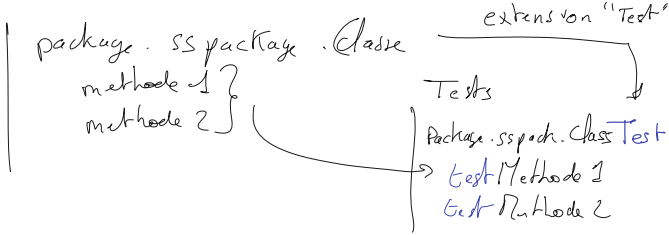
Test Driven Development

Objectif → avoir visibilité sur l'avancement
→ orienté specs, tests exhaustifs

1) Dev Cahier spec, exigences —
→ Analyse → squelette UML
→ Dabord → écrire les tests (avant le dev)
→ 1er grande partie = dev non productifs

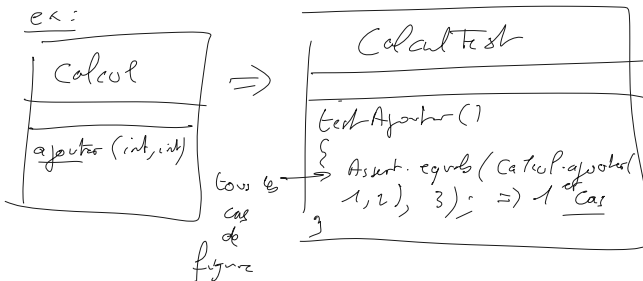


Dev

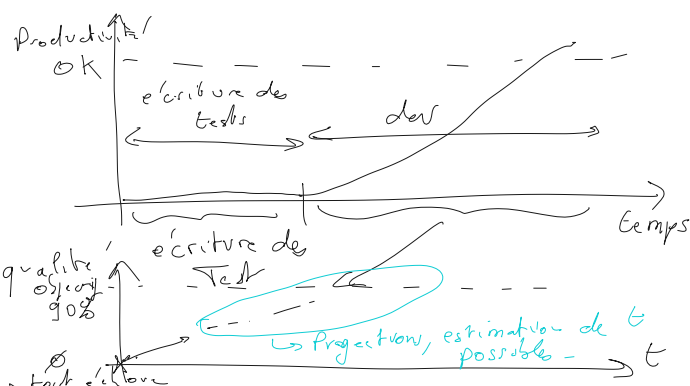


3) Pour chaque methode de la classe à tester
- On écrit tous les tests à effectuer

ex :



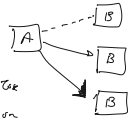
→ On fait tous les affirmations (Assertions) à Vraies, et le message en cas de non respect.



Mock, Shim, Stubs

Mock(ing) "se moquer"
mimer, singer

Principe
A utilise B

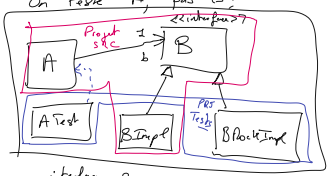


- utilise
- est un

Devra devs travailler chacun sur son système

A - ne peut attendre B
- ne doit pas compter sur B

On teste A, pas B !



interface B
{
void foo();
}

---

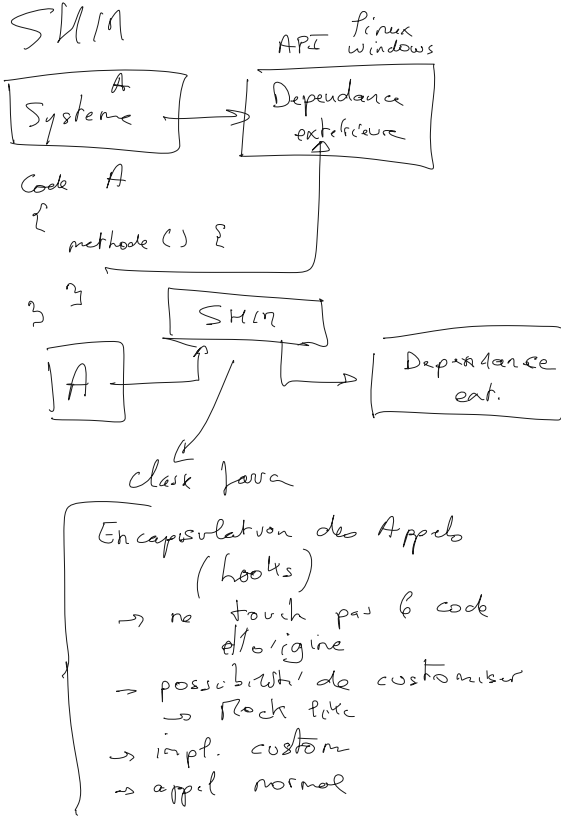
class BMockImpl implements B
{
void foo() {}
}

---

class A
{
e - - - - ?
B b.

b.methode(); -> tjRS ok en test
}

SHIM



Encapsulation des Appels (hooks)

- → ne touch pas le code d'origine
- → possibilité de customiser
  - → Rock like
- → impl. custom
- → appel normal

Stubs. ≃ intrusiveness overhead

Stub = Mock + simulation qui fonction~~n~~ toujours de f° de B

-