Back Office & Tests

Animé par Sylvain Labasse

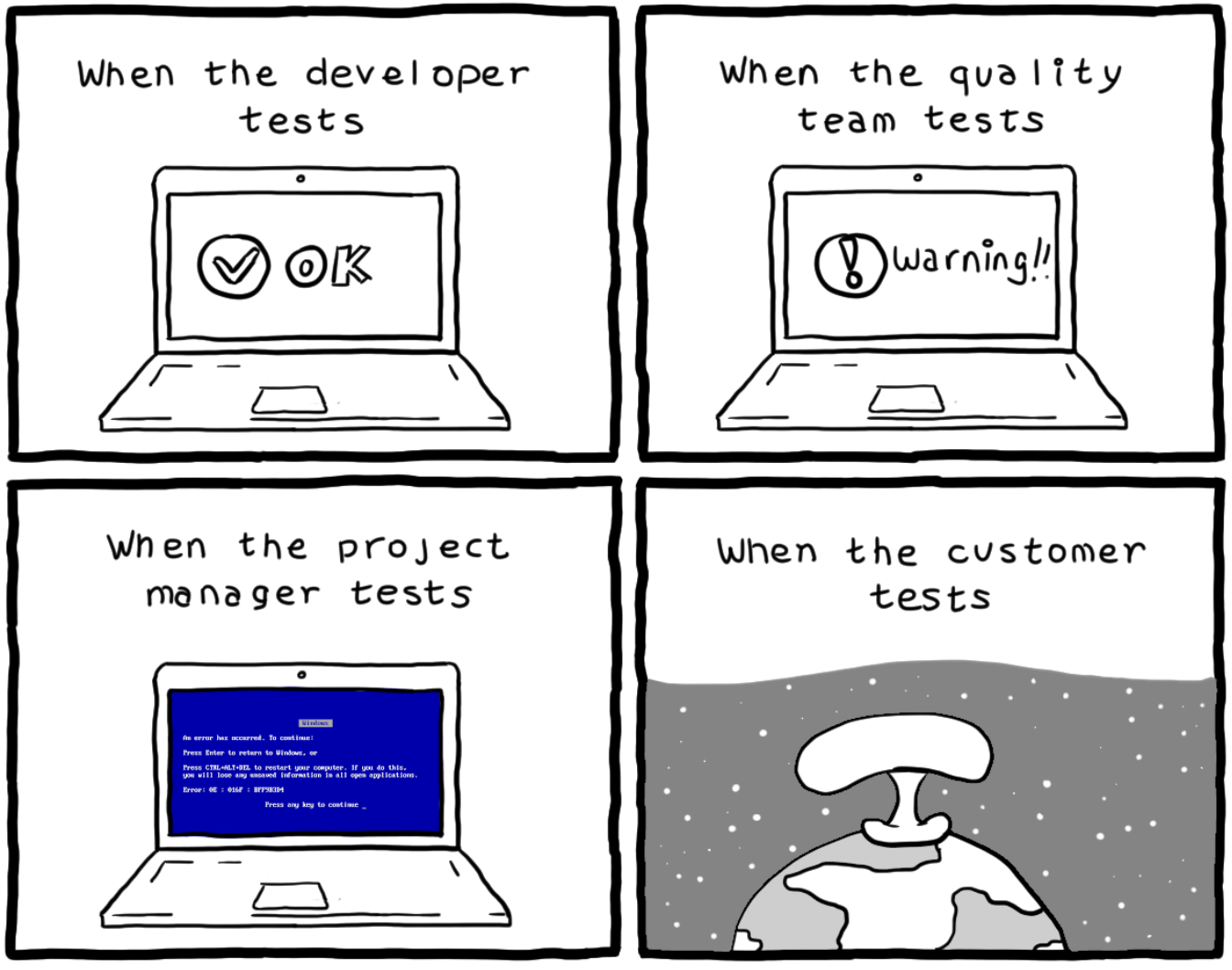


Figure 1 - src : {turnoff.us}

APRES CE MODULE, VOUS SAUREZ...

Appréhender l'importance des tests pour un Back Office

Créer un projet de test unitaire JUnit

Organiser et coder les cas de test d'une classe

S'assurer de la bonne gestion des erreurs

Remplacer les dépendances dans une classe testée

Vérifier les appels effectués par une classe

Diriger la conception d'une classe par les tests

Pre-Requis

Public

Développeur

Nécessaire

Classes en Java

Support

Copie des slides

Notes de cours

Réalisation des ateliers

L'environnement

Matériel

PC sous Windows / Linux

Connexion Internet

Logiciel

Eclipse, Java

Back Office & Tests

JUnit

- Vue d'ensemble
- Principes de conception
- Premiers tests
- Exceptions

Dépendances

- Classification
- Stubs
- Mocks

Back Office & Tests

Test Driven Development

- Red/Green/Refactor
- Triangulation
- Emergence

JUNIT

Objectifs

Apports des frameworks de test unitaire

Organisation et conception des tests

Prise en main de JUnit

Gestion des erreurs

JUnit

➔ Vue d’ensemble

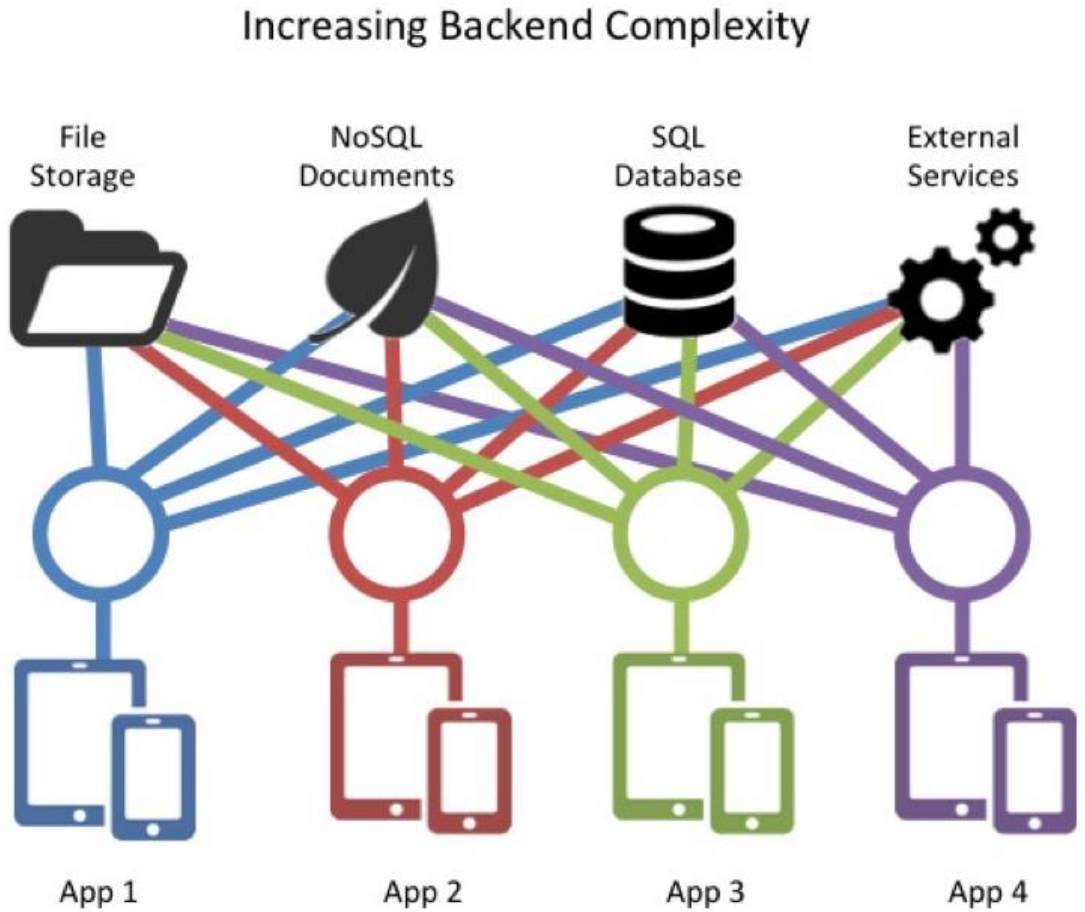
Principes de conception

Premiers tests

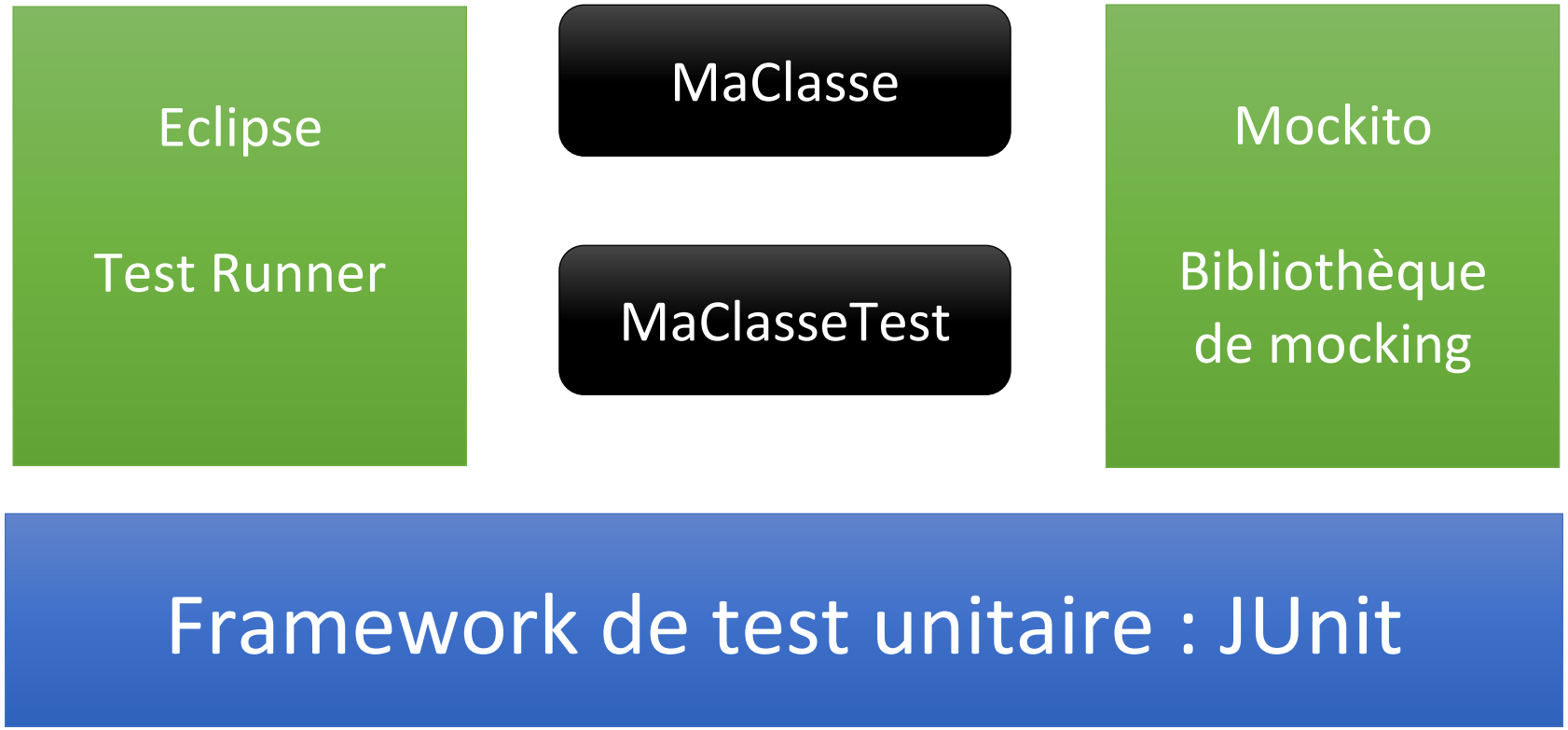
Exceptions

Résumé

VUE D'ENSEMBLE - BACKOFFICE



Vue d'ensemble - Tests



JUnit

- ✓ Vue d'ensemble
- ➔ Principes de conception
- Premiers tests
- Exceptions
- Résumé

Principes de conception

Organisation

Projet de test indépendant du projet testé, mais même dépôt

1 classe testée = 1 classe de test

Critères

Autonome et déterministe

Facile à écrire et facile à lire

Exécution rapide

Organisation

Unité de travail

Groupe de méthodes concourant à une action commune

Par unité de travail : Cas nominal, cas extrême, cas d'erreur

Méthode de test

1 cas = 1 méthode de test

Triple A - Arranger, Agir et faire un Assertion

JUnit

- ✓ Vue d'ensemble
- ✓ Principes de conception
- ➔ Premiers tests
- Exceptions
- Résumé

Premiers tests

Projet

Création d'un projet test dans la même solution

Ajout dépendance du projet

Classe de test

Nom de méthode explicite = cas de test

Setup rarement pertinent / préférer des méthodes privées

JUnit

- ✓ Vue d'ensemble
- ✓ Principes de conception
- ✓ Premiers tests
- ➔ Exceptions
- Résumé

Exceptions

Quand ?

Cas d'erreur

A vérifier : Instruction et Type de l'exception

Assertion

Instruction levant l'exception en lambda

```
assertThrows(XxxException.class, () -> instruction()) ;
```

JUnit

- ✓ Vue d'ensemble
- ✓ Principes de conception
- ✓ Premiers tests
- ✓ Exceptions
- ➔ Résumé

Resume

Apports des frameworks de test unitaire

Organisation et conception des tests

Prise en main de JUnit

Gestion des erreurs

ATELIER

Stocks

Ajouter à la classe Food un attribut « quantity », initialisé à zéro

Implémenter deux méthodes :

- Replenish(int qty) qui ajoute qty à la quantité déjà en stock
- Remove(int qty) qui retire qty à la quantité en stock

Tests

Proposer des tests nominaux, extrêmes et erreurs pour l'initialisation, l'ajout et le retrait des stocks.