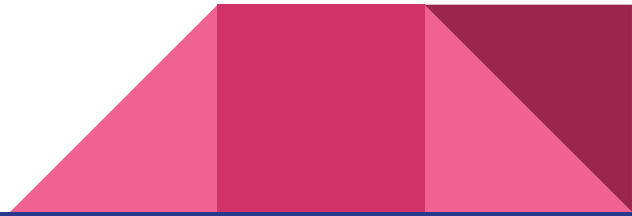
Software Architecture and Quality Assessment AL4T

Plan de cours

Jean-Guillaume Louis
(j3l@ecam.be)

Objectifs du cours

- Identifier et être capable de proposer des solutions quant aux problèmes de design dans le code
  - Violation de principes (SOLID)
  - Code smells
  - Techniques pour améliorer le code (injection de dépendance et autre design patterns)
- Mieux structurer ses projets informatiques
  - Conception d'interface
  - Organisation du code
  - Analyse des dépendances
  - Compréhension des architectures n-tiers, SOA, microservices
- Comprendre comment délivre le software de manière professionnelle
  - Propre utilisation d'un système de contrôle de version (Git) dans un environnement collaboratif
  - Ecriture de tests automatisés
  - Déploiement (mobile/cloud)
- JAVA (langage de support)
  - Base POO sur le langage -> classes, interfaces et héritage
  - Multithreading



Plan de cours

1. Notions de qualité et terminologie
2. POO en Java
3. Designs Patterns
4. Principes SOLID
5. Système de version de contrôle (bon usage de)
6. Automatisation des tests
7. Etude de cas: architecture microservice

Plan de cours

1. **Notions de qualité et terminologie**
2. POO en Java
3. Designs Patterns
4. Principes SOLID
5. Système de version de contrôle (bon usage de)
6. Automatisation des tests
7. Etude de cas: architecture microservice

Critères de qualité et métriques

“Familles” d’architecture -> Monolithique, N tiers, SOA, Microservices

Notion de dette technique

Problématiques typiques (dont Memory Leak, Race conditions)

Présentation de la structure d’un chaîne de livraison de logiciel sur deux cas de figures (web backend et mobile)

Plan de cours

1. Notions de qualité et terminologie
2. **POO en Java**
3. Designs Patterns
4. Principes SOLID
5. Système de version de contrôle (bon usage de)
6. Automatisation des tests
7. Etude de cas: architecture microservice

Base de la programmation Java (pratique)

- Classes et héritage
- Interfaces
- Exceptions
- Générique
- Bonnes pratiques

Plan de cours

1. Notions de qualité et terminologie
2. POO en Java
3. **Designs Patterns**
4. Principes SOLID
5. Système de version de contrôle (bon usage de)
6. Automatisation des tests
7. Etude de cas: architecture microservice

- Creational patterns
  - Builder, Factory, Singleton
- Structural patterns
  - Decorator, Facade
- Behavioral patterns
  - Command, Strategy, Observer
- Functional patterns
  - Lambda, Closure, Memoization
- Concurrency patterns
  - Structurer un programme multithread en Java
  - Memory model
  - Lock, Mutex, Semaphore
  - *Maybe CSP*

Plan de cours

1. Notions de qualité et terminologie
2. POO en Java
3. Designs Patterns
4. **Principes SOLID**
5. Système de version de contrôle (bon usage de)
6. Automatisation des tests
7. Etude de cas: architecture microservice

Chaque principe expliqué au travers d'exemples.

- Comment identifier les symptômes d'une violation de principes
- Comment corriger les problèmes de design

… Suivi d'une introduction plus poussée à l'injection de dépendance. Comment la mettre en place "from scratch" / en utilisant un framework (Dagger2)

Plan de cours

1. Notions de qualité et terminologie
2. POO en Java
3. Designs Patterns
4. Principes SOLID
5. **Système de version de contrôle (bon usage de)**
6. Automatisation des tests
7. Etude de cas: architecture microservice

Etude approfondie de Git et de son bon usage en travail collaboratif.

Analyse du modèle de donnée (orienté objet) de Git et de la structure en arbre de l'historique des versions.

Parallèle avec les système de Blockchain

Plan de cours

1. Notions de qualité et terminologie
2. POO en Java
3. Designs Patterns
4. Principes SOLID
5. Système de version de contrôle (bon usage de)
6. **Automatisation des tests**
7. Etude de cas: architecture microservice

Principe du TDD

Utilisation de JUnit

Mise en application des techniques d'inversion de dépendances.

Plan de cours

1. Notions de qualité et terminologie
2. POO en Java
3. Designs Patterns
4. Principes SOLID
5. Système de version de contrôle (bon usage de)
6. Automatisation des tests
7. **Etude de cas: architecture microservice**

Réalisation au cours d'une architecture microservice rudimentaire.

- Backend en Spring Boot
- Frontend Android (...parce que Java)

Implémentation d'une API Gateway

Deploiement backend en Container

Implémentation (et analyse du protocole) OAuth2.0 et OpenID Connect

Evaluation

Examen oral en deux partie:

- 1 question théorique sur un des chapitres du cours
- 1 exercice portant sur les notions de design (ex: code smells à éliminer)