# Révisions de Java, mise en pratique de git+maven

**Version: develop**

Emmanuel BRUNO

2023-10-24

### Objectif

L'objectif de ce TP est réviser les concepts de base du langage Java que vous devez maîtriser. Il permettra aussi une révision de la mise en pratique de Git et de Maven. Appuyez vous sur https://bruno.univ-tln.fr/notebooks/.

Vous lirez le sujet en entier. Vous mettrez en place un projet Github à partir lien fournit en cours pour suivre l'avancement de vos révisions (mise en place d'une roadmap, de tickets pour chaque question, suivi des bug et livraisons).

Pour tous les exercices, vous devez écrire la javadoc minimale, des tests unitaires utiles et vérifier votre code avec sonarlint au fur et à mesure de l'avancement.

La compilation, la génération de la documentation et des artefacts (fichier jar et executables) seront réalisés avec Maven.

### Au fur et à mesure du développement

### Livraison du résultat

Fabriquer automatiquement avec maven un fichier jar exécutable qui contient toutes les classes nécessaires. Exécuter l'application à partir de ce fichier.

### Gestion des log

Modifier vos classes pour que les logs sont gérés proprement à l'aide du Logger du jdk ou mieux de Sl4J.

### Test de vos classes avec JUnit

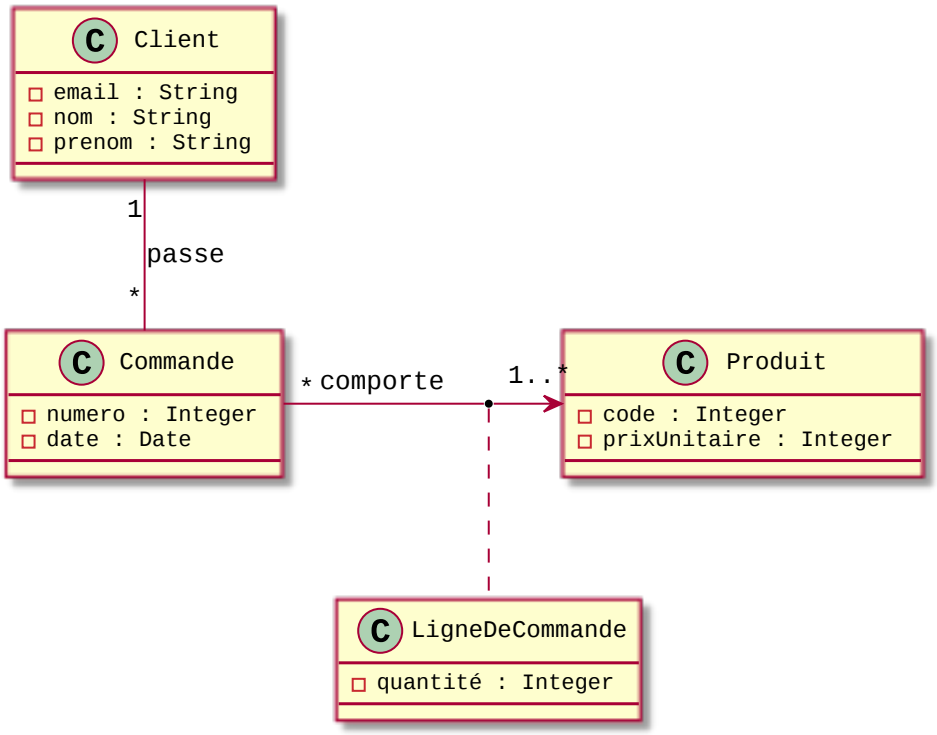
Mettez en place des tests unitaires au fur et à mesure du développement en utilisant Junit

L'exécution des tests sera faite par maven.

### Classes, Instances et Héritage

### Hiérarchie

Mettez en place les classes Java nécessaire pour représenter un système de gestion de commandes en vous inspirant du diagramme de classe UML ci-dessous :




**Un package `api` sera dédié à l'API, c'est à dire que des des interfaces seront définies pour chaque classe essentielle du modèle**

Vous veillerez :

- A ce que les visibilités soient correctes
- A utiliser correctement si besoin l'héritage et les interfaces
- A protéger les constructeurs par des factory ou des builder patterns.
- Mettez en places les méthodes nécessaire pour que l'égalité canonique et la comparaison soient correctes.

### Collections

Mettez en place une implantation de votre api un package spécifique qui utilise les collections. Ecrivez une DAO minimale qui utilise la programmation fonctionnelle pour accéder et metter à jour les données.

**Un package `collection` sera dédié à l'implantation du modèle avec les colelctions**

### Persistance avec JPA

Dans un autre package que celui de la question précédente et en utilisant JPA, mettez en place la persistance de l'application dans une base de données relationnelles de votre choix.

**Un package `jpa` sera dédié à l'implantation du modèle avec JPA**

### Rest/Websocket

Mettez en place des ressources REST au dessus de

### IHM

Créer une IHM simple permettant au moins de consulter et de mettre à jour l'une des entités en utilisant l'API REST/Websocket.

### Docker

Préparer une livraison de votre application avec Docker (base de donnée, API et implantation hors IHM).