Programmation Objet
Concepts Avancés

(mais on vous parlera surtout du métier de développeur)

Séance 1 – 6 octobre 2022

Mathieu Gandin – Michel Blancard

Objectifs et organisation

Qu’est-ce qu’on entend par maîtriser la POO ?

- Maîtrise des techniques et des pratiques de développement
- Prendre du recul : comprendre le problème avant de sauter sur la solution
- Comprendre le fonctionnement d’une équipe de développement

Objectifs et organisation

Qu’est-ce qu’on peux vous apprendre ?

- Approche par la pratique
- Prioriser ce qui va vous servir souvent
- Vous préparer à la transition entre formation initiale et formation continue

Objectifs et organisation

Pour qui ?

- Habitude d'être guidé : apprentissage de l'autonomie
- Déjà l'habitude de chercher par soi-même : fouiller les sujets pour aller plus loin

Le métier de développeur

Différents statuts :

- En interne (chez un éditeur logiciel, startup, grosse entreprise, etc.)
- Comme consultant en ESN (aka SSII)
- Comme consultant freelance

Le métier de développeur

Ses missions

- Vu comme un exécutant : il transcrit dans l'urgence des spécifications détaillées sans participer à la conception

- Vu comme un expert / créatif : il prend part en équipe à la construction d'un produit et assure une partie de la réalisation technique

Le métier de développeur

Sa rémunération : quelques ordres de grandeur

(à Paris, pour de la prestation)

- Développeur débutant : ~320€ HT par jour
- Développeur avec de l'expérience : ~600-650€ HT par jour
- Profil spécialisé / tech lead : ~800-900€ HT par jour
- Cabinet plutôt haut de gamme : ~1200€ HT par jour

Le métier de développeur

Ses conditions de travail

Le métier de développeur

Ses conditions de travail

De tout !

Le métier de développeur

Il y a d'autres possibilités que juste les ESN

ESN (aka SSII) :

- risque de perte de niveau si vous restez dans votre domaine d'expertise
- pas toujours joyeux ni très humain
- peu d'évolution à part chef de projet / directeur de mission

Négocier son salaire, comprendre la différence entre net, brut, super brut, fixes et primes, avantages et vrai salaire, etc

https://www.youtube.com/watch?v=y6h0jSwkLZA

Le métier de développeur

Son vocabulaire

- Le « domaine métier »
- « Expérience utilisateur » (UX) / « interface utilisateur » (UI)
- « Valeur client »

Bien programmer

Bien programmer

Règle n°1 : ???

Bien programmer

Règle n°1 : Low coupling

Bien programmer

Règle n°1 : Low coupling

Exemple : DRY (Do not Repeat Yourself)

Bien programmer

Règle n°1 : Low coupling

Exemple : DRY (Do not Repeat Yourself)

Formulation générale :

« Disposer ensemble ce qui
change fréquemment ensemble »

Observation capitale : ça dépend du contexte !

Bien programmer

Règle n°1 : Low coupling

Exemple : DRY (Do not Repeat Yourself)

Formulation générale :

« Disposer ensemble ce qui
change fréquemment ensemble »

Observation capitale : ça dépend du contexte !

Objectif : rendre le code évolutif

Bien programmer

Règle n°2 : ???

Bien programmer

Règle n°2 : KISS (Keep It Simple Stupid)

Exemple : bien nommer ses variables et fonctions

Objectif :

- Rendre le code facilement compréhensible
- Toujours choisir la solution la plus simple (mais pas la plus simpliste) pour résoudre nos problèmes métiers et techniques

Bien programmer

L’outil à tout faire

Bien programmer

L’outil à tout faire

abstraction

Bien programmer

L’outil à tout faire

abstraction

= interface

Bien programmer

L’outil à tout faire

abstraction

= interface

= contrat

Bien programmer

L’outil à tout faire

abstraction

= interface

= contrat

= frontière

Bien programmer

L’outil à tout faire

abstraction

= interface

= contrat

= frontière

= niveau d’indirection

Bien programmer

L’outil à tout faire

abstraction

= interface

= contrat

= frontière

= niveau d’indirection

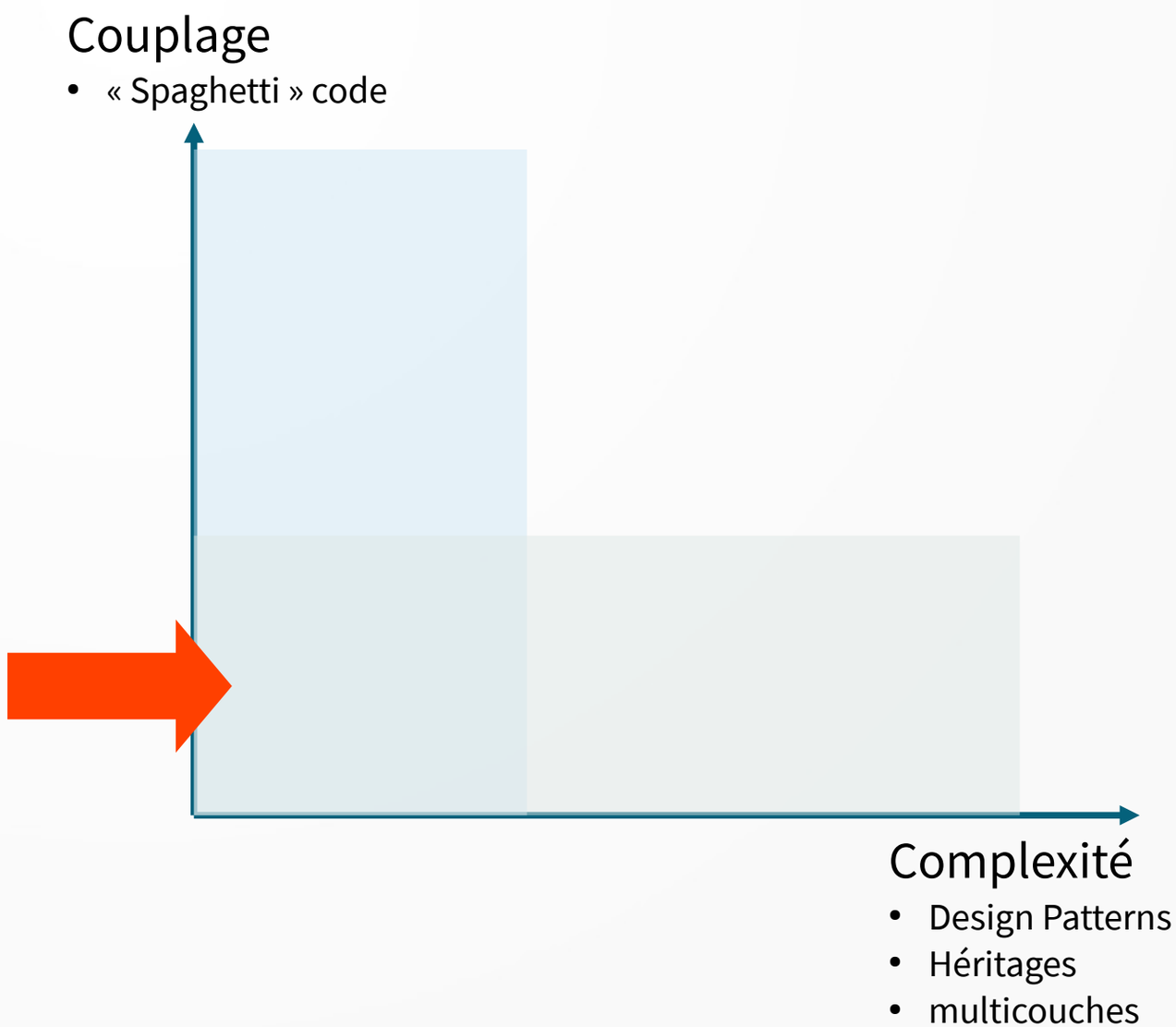
= découplage

Objectif : réduit le couplage (contre de la complexité)

Bien programmer

Low coupling / simplicité :
objectifs antagonistes

Bien programmer, c'est arbitrer
entre ces deux objectifs

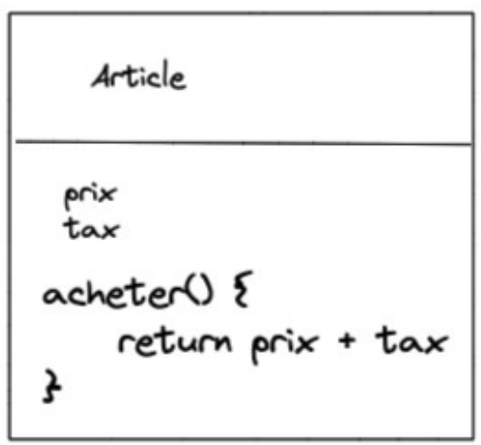


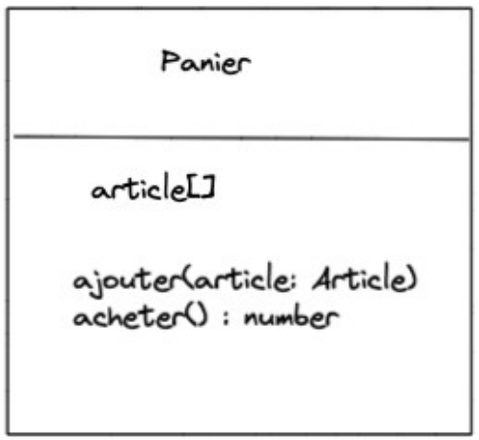
Bien programmer

Et l'Objet, dans tout ça ?

C'est une abstraction qui mêle données (membres) et comportement (méthodes).

Séparation des responsabilités






```
const playstation = new Article(500);
const guitare = new Article(700);

const panier = new Panier();
panier.ajouter(playstation);
panier.ajouter(guitare);

panier.acheter() => 1500 euros
```

Travailler à plusieurs

Découper le travail :

- « feature » ou « fonctionnalité » : elle apporte de la « valeur client »
- Une « feature » est spécifiée par une « user story » (US) :

  « As a …, when … I can … »

- Le « Product Owner » (PO) spécifie et priorise les US avec l'équipe de développement
- Régulièrement, on fait une démonstration au client

Travailler à plusieurs

Contribuer :

Travailler à plusieurs

Contribuer :

- S’assigner une User Story

Travailler à plusieurs

Contribuer :

- S’assigner une User Story
- Implémenter dans une branche dédiée

Travailler à plusieurs

Contribuer :

- S’assigner une User Story
- Implémenter dans une branche dédiée
- Demander une « code review » et prendre en compte les retours

Travailler à plusieurs

Contribuer :

- S’assigner une User Story
- Implémenter dans une branche dédiée
- Demander une « code review » et prendre en compte les retours
- Faire éventuellement un « rebase » pour garder un historique git linéaire

Travailler à plusieurs

Contribuer :

- S’assigner une User Story
- Implémenter dans une branche dédiée
- Demander une « code review » et prendre en compte les retours
- Faire éventuellement un « rebase » pour garder un historique git linéaire
- Faire un merge dans la branche d’intégration