Kata
Junit

Philippe Collet
d'après Sébastien Mosser et Simon Urli

Polytech Nice Sophia – SI3
2017-2018

Plan

- Game of Dices : specs
- Architecture (Maven, en avance…)
- Tache 1 : Lancer un dé
- Tester (Junit)
- Tache 2 : Ajouter un joueur
- Tester
- Tache 3 : Jouer aux dés !

Games of Dices : product backlog

Pas forcément de *stories* mais au moins un découpage vertical

1. Etre capable de jeter un dé
   - Critère d'acception : le dé a 6 faces, et retourne un nombre aléatoire en 1 et 6.
2. Associer un jet de dé à un joueur précis
   - Critère d'acception : un joueur a un nom, et expose la valeur obtenue de son propre dé
3. Le jeu de dés se joue à 2, et le joueur qui obtient la valeur maximale après un lancer gagne (ex-aequo entraîne une relance, pas de vainqueur après 5 matches ex-aequo)
   - Critère d'acception : le jeu expose un vainqueur en suivant les règles

Architecture

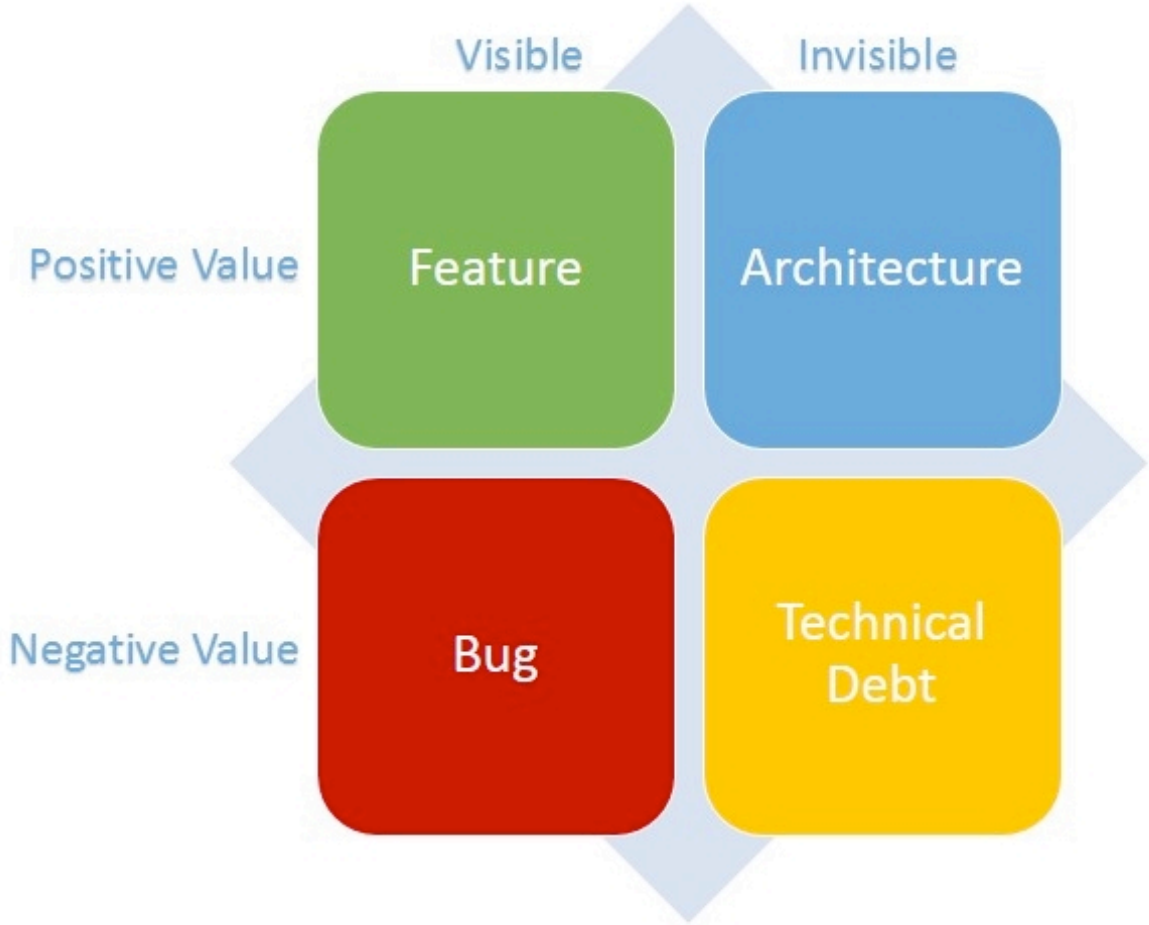
- Création des répertoires
  - Src, test
- Création d'un pom.xml minimal
- Pas forcément test-driven, mais testé unitairement

Tache 1 : Etre capable de jeter un dé

- Critère d'acception : le dé a 6 faces, et retourne un nombre aléatoire en 1 et 6.

- Package god (Game Of Dices)
- Classe Dice
  - Méthode roll
- Reproductibilité du jeu ?
  - Objet random fourni à la création
- Si random ne marche pas bien ?
  - RuntimeException…
  - C'est clairement de la dette technique !

Dette technique ?



Tester

- Modéliser les critères d'acceptation
  - Bonne propriété : quand les tests passent, la tâche est terminée !
- On a besoin de tester :
  - Lancer un dé rend une valeur ?
  - Une valeur hors de [1-6] lance une exception

Créer son propre Random: bof bof…

- Pas terrible de redéfinir une classe juste pour les tests
  - Si elle était plus compliquée, ce serait quasiment infaisable
- Au second semestre, en QGL, on pourra simuler des comportements de certaines classes à l'aide de Mock !

Tache 2 : Associer un jet de dé à un joueur précis

- Critère d'acception : un joueur a un nom, et expose la valeur obtenue de son propre dé

- Classe Player
  - Constructeur avec le nom du joueur et le dé
  - Attribut lastValue (et getter) pour la dernière valeur du dé (-1 sinon)
  - Méthode play lance le dé et stocke la valeur

- Test ?

Tache 3 : Le jeu de dés se joue à 2…

- et le joueur qui obtient la valeur maximale après un lancer gagne
  - Ex-aequo : on rejoue, jusqu'à 5 fois
  - Egalité si égalité après 5 jeux
- Critère d'acception : le jeu expose un vainqueur en suivant les règles

- Création d'une classe Game
  - Deux joueurs (left, right)
  - 1 si left gagne, -1 si right gagne, 0 sinon
- Tests ?
  - Cas « pas de vainqueur »
  - Cas « un vainqueur »