TP1 Nim
Programmation Java et composants

Jean-Claude Royer

12 novembre 2015

1 Le sujet

Il s'agit de partir de l'exemple simple dont une architecture a été vue en cours et de faire quelques tests en Java en utilisant une approche par composants. L'approche suggérée est celle vue en cours mais des améliorations personnelles sont les bienvenues. Cela permettra d'évaluer ce qu'il est possible de faire et ce qui est plus difficile à réaliser en Java. Pour des principes d'implémentation voir la fin du cours du module "architecture". Le type de schéma ci-dessus fait partie de la panoplie d'UML 2, certains outils ne le supporte pas, ArgoUML par exemple, par contre Dia, Visual Paradigm et d'autres le permettent.

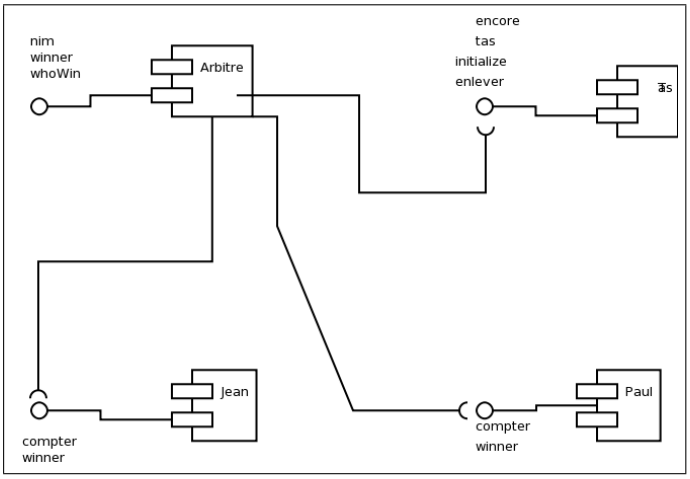

FIGURE 1 – Architecture avec deux joueurs

La description des services pourvus est la suivante :

- `Joueur`
  - `compter(int)` : un pourvu avec retour d'une valeur
  - `winner` : affirme qu'il a gagné
- `Tas`
  - `encore()` : indique si il reste des allumettes ou pas
  - `initialise()` : initialisation aléatoire du tas
  - `enlever(int)` : enlève des allumettes du tas
  - `getTas()` : retourne la valeur courante du tas

- `Arbitre`
  - `initialize(Joueur,Joueur)` : initialise une partie avec deux joueurs
  - `nim()` : déroulement d'une partie.
  - `winner()` : donne le nom du vainqueur

2 Le travail

Il s'agit d'implémenter les trois classes correspondantes à ces composants et d'écrire des configurations permettant de simuler des cas simples de coordination. Vous devez réfléchir aussi un peu à l'architecture car celle proposée n'est pas très détaillée. Pour l'implémentation vous serez guidez pendant le TP, nous réutiliserons des principes exposés à la fin du cours. Utiliser une approche à composants autant que possible, faire apparaître les interfaces pourvues et requises et assembler vos projets en .jar. Les interfaces requises ne seront pas utilisées mais seulement les pourvues.

1. Vérifier que cette architecture est réalisable dans le sens où elle ne définit pas des communications cycliques. De là en déduire un ordre de création et de packaging de vos composants.
2. Découper en trois projets `Tas`, `Joueur` et `Arbitre`.
3. `Arbitre` définira les configurations du jeu avec uniquement deux joueurs
4. Faire des tests à deux joueurs et un joueur et un intelligent et tests
5. Exporter votre `Arbitre` dans un jar et l'importer dans un autre projet pour le tester.
6. Configuration a n joueurs ($n \geq 2$), ce sera la classe `Nim`, donc définir ses interfaces et son implémentation.
7. Réutiliser le .jar `Nim` dans un autre projet avec génération « aléatoire » des joueurs.
8. Noter aussi que vous pouvez améliorer votre conception en définissant des singletons représentant les "vrais" composants.
9. Questions additionnelles
   (a) Séparer autant que possible `Joueur` et `Intel`
   (b) Ajouter un moyen d'obtenir la liste des interfaces requises et pourvues
   (c) Tester l'utilisation des interfaces requises, peut-être en les séparant du code
   (d) Essayer d'implémenter les services `bind` de Fractal (cf le cours)