ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

Projet Java ILO
2011

Composants v2.0

Objectif

On veut pouvoir créer des composants permettant de réaliser des fonctions. Le cadre d'application recherché dans ce projet est la réalisation de fonctions mathématiques, mais le paradigme des composants est applicable à la plupart des domaines.

Définitions

Un composant est une entité possédant un certain nombre d'entrées et un certain nombre de sorties. Les composant réalisent une opération à partir des valeurs présentes en entrée, puis produisent un résultat sur leurs différentes sorties.

- Un composant typique attend de nouvelles valeurs sur ses entrées afin de réaliser son ou ses opérations et produire des valeurs sur ses sorties.
- On veut pouvoir enchaîner des composants entre eux de manière à construire des fonctions plus complexes que les fonctions des composants eux-mêmes. On pourra ainsi construire des « macro-composants » constitués par un arrangement de composants plus simples.
- On veut que les composants fonctionnent de manière concurrente.
- Certains composants ne possèdent pas d'entrées mais juste des sorties : cela peut être le cas par exemple d'un composant faisant évoluer automatiquement une variable « x » entre « a » et « b » avec un pas « p », cette variable étant ensuite transmise à un composant calculant une fonction f(x) sur cette variable.
- D'autres composants ne possèdent que des entrées et pas de sorties : Cela peut être le cas par exemple d'un composant stockant les résultats successifs d'une fonction f(x) en vue de leur affichage sur la console, ou bien d'un composant traçant le résultat d'une fonction f(x).
- Une sortie d'un composant pourra être connectée à plusieurs entrées de plusieurs composants (Fan-out)
- En revanche une entrée d'un composant ne pourra être connectée qu'à un seul autre composant (pas de Fan-in).
- Les données échangées entre composants peuvent être de différentes natures :
    - Il peut s'agir uniquement d'événements déclencheurs.
    - Ou bien de données scalaires (entiers, flottants, booléens, ou autres) : à raison d'une seule valeur à la fois. On appelle souvent ce type de données SFxxx (pour Single Field).
    - Ou encore de données multiples : plusieurs valeurs à la fois. On appelle souvent ce type de données MFxxx (pour Multiple Field).
    - Les données scalaires ou bien multiples peuvent changer de valeur dans le cas de variables ou bien toujours conserver la même valeur s'il s'agit de constantes. Auquel cas, il serait intéressant pour le composant qui reçoit des valeurs constantes de ne pas attendre de nouvelles valeurs.

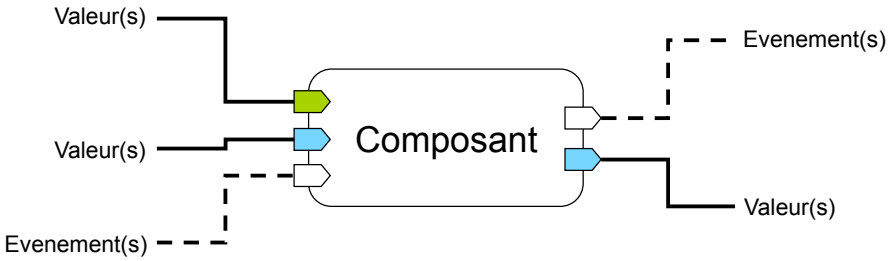

Figure 1: Composant à 3 entrées et 2 sorties

Fonctionnement d'un composant

- Entrées/Sorties de données
  - Tout composant qui reçoit des valeurs sur ses entrées possède (au moins) une fonction de calcul qui calcule un résultat ou effectue une opération d'après les valeurs en entrée et éventuellement produit un résultat sur une de ses sorties.
  - Lorsqu'un composant reçoit une nouvelle valeur sur une de ses entrées, il peut :
    - Soit effectuer un nouveau calcul et mettre à jour sa ou ses sorties
    - Soit attendre que certaines autres entrées reçoivent elles aussi de nouvelles valeurs avant d'effectuer son calcul et poster ses résultats.
- Entrées/Sorties d'évènements
  - Les événements ne contiennent pas de données
  - Lorsqu'un composant reçoit un événement en entrée, il effectue l'action correspondant à cet événement.
  - Lorsqu'un composant émet un événement en sortie le ou les composants qui recevront cet événement effectuerons chacun une action.
- Connections entre composants
  - Un composant doit posséder de quoi connecter ses sorties sur d'autres composants tout en contrôlant que la connexion est possible (on ne pourra pas connecter une sortie contenant un seul booléen sur une entrée d'un autre type).

Exemples de construction avec des composants

On souhaite afficher une famille de courbes $y = \cos^n(x)$ pour différentes valeurs de $n \in [1\ldots5]$ et de $x \in [0\ldots\pi/2]$. Une solution possible en termes de composants pourrait être celle-ci (La liste des entrées/sorties de ces composants n'est pas forcément exhaustive) :

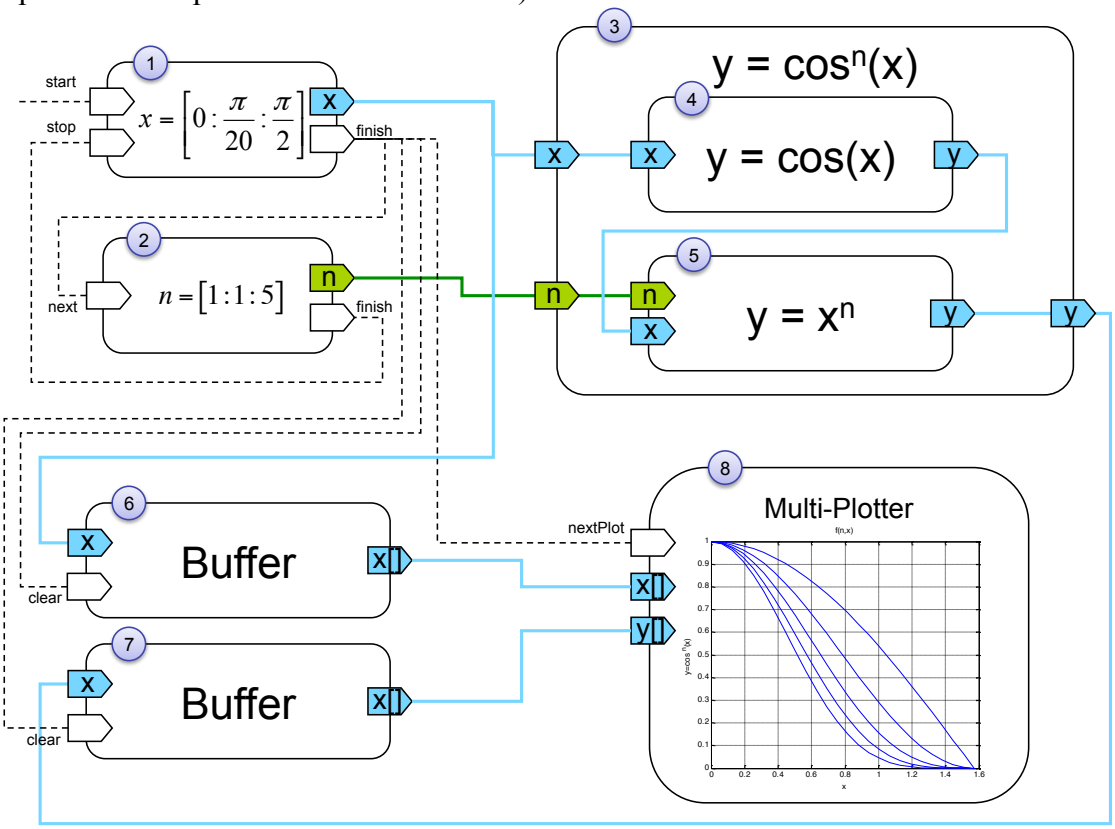

Figure 2 : exemple d'utilisation de composants

1. Composant itérant une variable x réelle comprise entre 0 et π/2 par pas de π/20
   - Lorsqu'il reçoit l'événement « start » il commence à itérer la valeur de x.
   - Chaque nouvelle valeur de x est envoyée sur la sortie.
   - Lorsque la valeur de x atteint π/2, l'événement « finish » est émit et la valeur de x est remise à 0.
   - Si le composant reçoit l'événement « stop », il s'arrêtera d'itérer la valeur de x lorsqu'elle aura atteint une nouvelle fois la valeur π/2.
2. Composant itérant une variable n entière comprise entre 1 et 5 par pas de 1.
   - Ce composant n'itère la prochaine valeur de n que lorsqu'il reçoit l'événement « next », auquel cas n est incrémenté et sa valeur est envoyée en sortie.
   - Lorsque la valeur de n atteint 5, l'événement « finish » est émis.
3. Macro composant composé de deux composants de calcul (4 & 5).
4. Composant de calcul de la fonction cosinus : $y = \cos(x)$
   - Effectue son calcul à chaque nouvelle valeur de x sur son entrée et envoie le résultat sur sa sortie.
5. Composant de calcul effectuant l'opération $y = x^n$:
   - Effectue son calcul si au moins une de ses entrées (x ou n) reçoit une nouvelle valeur et envoie le résultat sur sa sortie.
6. Buffer permettant d'accumuler plusieurs valeurs :
   - Lorsqu'un buffer reçoit une nouvelle valeur, celle ci est ajoutée à la collection de valeurs déjà reçues et cette collection est envoyée en sortie (on aurait pu aussi envisager de n'envoyer la collection de valeurs en sortie que lorsque le buffer reçoit un événement spécifique).
   - Lorsque le buffer reçoit l'événement « clear », il efface le contenu de sa collection de valeurs.
7. Identique à 6
8. Composant de dessin de plusieurs courbes

- Ce composant dessine la courbe composée de collections de coordonnées x et y à chaque fois que l'ensemble des deux entrées x[] et y[] sont mises à jour. Il dessine aussi l'ensemble des courbes qu'il a sauvegardées auparavant.
- Lorsque le composant reçoit l'évènement « nextPlot » il enregistre les données de la courbe dessinée à partir des entrées x[] et y[] dans sa collection de courbes.

Composants envisagés

Les opérations mathématiques envisageable sont nombreuses mais en voici quelques unes :

- Les opérateurs binaires : *, /, +, -
- Les opérateurs booléens:
  - Unaire : non
  - Binaires : et, ou, ou exclusif
- Les tests logiques:
  - Egalité, différence, inférieur (ou égal) à, supérieur (ou égal) à.
- Les fonctions mathématiques usuelles
  - Les puissances : x^n ($n \in N$ ou $n \in R$).
  - Les fonctions exponentielles, logarithmes et factorielle.
  - Les fonctions trigonométriques : cos, sin, tan, acos, etc.
  - Les fonctions d'arrondi (en dessous, au dessus, en moyenne), ainsi que min et max.
  - La valeur absolue et le signe.
- D'autres fonctions mathématiques
  - La dérivée (numérique)
  - L'intégration (numérique).
  - …

Il vous faudra par ailleurs construire un certain nombre de composants pour alimenter ces opérations (des producteurs de données de différentes natures) et des composants pour stocker, afficher et dessiner ces données (des consommateurs de données) comme ceux mentionnés dans la Figure 2 page 2 :

- Macro composants : composés d'autres composants
- Producteurs de données :
  - Composants permettant de produire des constantes de types différents.
  - Composants permettant de produire des variables de types différents variant entre deux bornes et avec un pas spécifiés.
- Consommateurs de données :
  - Composants permettant d'afficher une ou plusieurs valeurs dans la console ou de manière graphique
- Producteurs/consommateurs de données
  - Tous les composant permettant de transformer les données pour d'autres composants comme les Buffers mentionnés dans la Figure 2.
- Et de manière plus générale tous les composants dont vous pourriez avoir besoin pour réaliser des fonctionnalités particulières.

Travail demandé

Ce travail est à effectuer en binômes.

Concevez une hiérarchie de classes propre à représenter tous les éléments décrits précédemment (différents types de composants, valeurs, évènements, entrées, sorties, etc.) et suffisamment souple pour s'adapter si besoin à de nouveaux types de composants. Présentez le tout sous la forme d'un graphe de classes UML. Celui-ci devra présenter les caractéristiques suivantes :

- Contenir l'ensemble des membres des classes (attributs et méthodes) quelle que soit leur accessibilité.
- Présenter les types des arguments ainsi que le type de retour des méthodes.
- Présenter les relations d'héritage et d'implémentation ainsi que les relations d'agrégation ou de composition.
- Si besoin, Présenter les relations de dépendances ou des annotations textuelles qui explicitent le graphe de classe (comme celles utilisées dans le cours sur les Design Patterns).

Lorsque vous disposerez de vos classes vous pourrez dans un premier temps faire fonctionner les composants dans un simple programme java.

Puis vous implémenterez une interface graphique permettant de concevoir et d'exécuter un ou plusieurs arrangements de composants.

Il est important que ces deux parties soient bien distinctes de manière fonctionner avec ou sans interface graphique. Pour ce faire, respectez l'architecture Model/View/Controller (MVC) qui vous permettra d'utiliser un même Modèle avec plusieurs couples de View/Controller.

Interface Graphique

L'interface graphique doit permettre de concevoir un arrangement de composants ou de macro composants sélectionnés parmi ceux que vous aurez implémenté, puis de lancer l'exécution concurrente de ces composants, et s'il y a lieu, d'afficher les résultats de l'exécution de vos composants.

Il sera sans doute utile de consulter la documentation des widgets SWING qui proposent une large palette de widgets : http://java.sun.com/docs/books/tutorial/uiswing/index.html

Pour la partie dessin de courbes, nous vous conseillons d'utiliser l'API JFreeChart, disponible à l'adresse suivante : http://www.jfree.org/jfreechart/

Evaluation du projet

L'évaluation de votre projet ILO se déroulera en trois étapes :

1. La première étape consistera à nous présenter le graphe de classes que vous aurez conçu dans la première partie du projet deux séances après la présentation du projet (à savoir les 12/12 et 13/12). Vous mentionnerez dans vos graphes de classes les Design Patterns utilisés. Ces graphes de classes seront notés puis corrigés avant de vous être retournés avec d'éventuelles corrections.
2. La seconde étape de l'évaluation se fera sous forme de soutenances. Celles-ci auront lieu la semaine du 16 au 20 janvier 2012 et dureront 15 minutes. Vous nous présenterez le fonctionnement d'un arrangement de composants sans interface graphique, puis avec interface graphique.
3. Vous déposerez à la fin du projet vos **sources documentés** ce qui nous permettra de les vérifier en cas en problème et de générer la documentation de vos classes afin d'en vérifier la cohérence. Par ailleurs, vous déposerez un **dossier** qui vous permettra de commenter vos graphes de classes en termes de choix d'implémentation et/ou de design patterns implémentés. Vos choix devront être argumentés.