

Durée : 1h30

Documents autorisés : Polycopiés de cours, TDs et TP.

Exercice 1 : Constructeurs / Destructeurs (6 points)

- Qu'est ce qu'un constructeur : (2 points)
 - Quelle est son utilité ?
 - Peut-on l'invoquer comme une méthode normale ?
 - Quel est son type de retour ?
- Pour quelle raison ne définit-on pas toujours de constructeur par défaut lorsqu'on a défini d'autres constructeurs ? (1 point)
- En C++ on est parfois obligé de définir un constructeur par défaut, pour quelle raison ? (1 point)
- Expliquer les principes de création d'une instance en java et C++, ainsi que les mécanismes de gestion mémoire associés. Même question pour la destruction d'une instance. (2 points)

Exercice 2 : Généricité (4 points)

- Quelles sont les deux formes de généricités offertes par Java depuis l'introduction de « Java Generics » et que permet « Java Generics » par rapport à l'ancienne forme de généricité. (2 points)
- Quelle est la différence majeure entre Java Generics et les patrons de classes C++ en termes de classes génériques comme des Tableau<Compte> par exemple. (2 points)

Exercice 3 : Polymorphisme (2 points)

- Qu'est ce que le polymorphisme et à quoi s'applique la notion de polymorphisme ? (1 point)
- Expliquez la notion de lien dynamique. Cette notion est-elle disponible dans les langages comme Java et C++ et à quelles conditions ? (1 point)

Exercice 4 : Conception (8 points)

Le but de cet exercice est de vous faire construire un ensemble de classes permettant de représenter différents types de formes géométriques ainsi qu'un certain nombre d'opérations sur ces mêmes formes en utilisant un « Graphe de scène »

Un graphe de scène est un arbre représentant l'agencement des formes entre elles. Il est composé de deux types de nœuds :

- Les nœuds de groupement qui contiennent d'autres nœuds (en gris dans la Figure 1, page 2)
- Les nœuds terminaux qui contiennent les formes géométriques (en blanc dans la Figure 1, page 2)

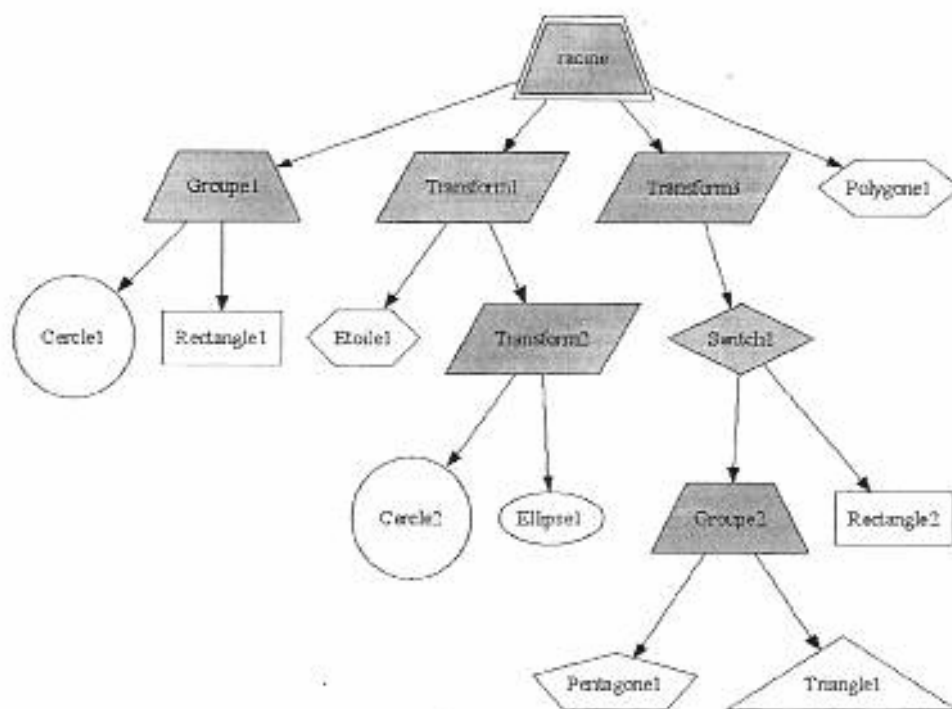


Figure 1 : Exemple de graphe de scène

Parmi les nœuds de groupement, on trouve

- Le « Groupes » qui ne font que contenir d'autres nœuds.
- Les « Transformations » qui contiennent d'autres nœuds et appliquent des transformations affines sur les nœuds qu'elles contiennent. Ces transformations peuvent être :
 - des translations,
 - des facteurs d'échelle,
 - ou des rotations.
- Les « Switch » qui permettent de n'afficher qu'un seul nœud à la fois parmi les différents nœuds contenus dans le « Switch ».

Parmi les formes géométriques contenues dans les nœuds terminaux on peut trouver

- Les cercles et les ellipses
- Les triangles : quelconques, isocèles ou équilatéraux
- Les rectangles et les carrés
- Les étoiles à n branches
- Et plus généralement les polygones : quelconques ou bien réguliers comme les pentagones ou les étoiles.

Question : Concevez une hiérarchie de classes Java propre à représenter les nœuds du graphe de scène décrit précédemment avec les transformations et les formes géométriques. Présentez le tout sous la forme d'un graphe de classes UML.

A savoir :

- L'héritage multiple n'existe pas en Java, néanmoins une classe peut implémenter plusieurs interfaces; on parle alors d'héritage multiple de comportements.
- Les membres publics sont précédés de : +
- Les membres protégés sont précédés de : #
- Les membres privés sont précédés de : -
- Les relations d'héritage sont indiquées par les flèches en traits pleins.
- Les relations d'implémentation sont indiquées par les flèches en traits pointillés.
- Les membres ou les classes concrète(s) sont indiqués en texte droit.
- Les membres ou les classes abstraite(s) sont indiqués en texte italique.
- On précisera pour les classes abstraites s'il s'agit d'interfaces ou de classes abstraites.

- Les compositions sont indiquées par un losange plein : par exemple la classe AB possède une instance de la classe « Element » et elle est responsable de la création et de la destruction de cette instance.
- Les agrégations sont indiquées par un losange vide : par exemple la classe ACD possède une collections d'instances de la classe « Element », mais elle n'est pas responsable de l'instanciation de cette collection (on supposera qu'elle lui est donnée).

Exemple :

