Ministère de l'Education Nationale
Conservatoire National des Arts et Métiers
Institut d' Informatique d' Entreprise

Examen ILO

5 janvier 2006

Durée : 1h30

Documents autorisés : Polycopié de cours, TDs et TPs.

Exercice 1 : Copie par référence ou copie profonde (2 points)

a) Quelle est la nuance entre la copie par référence et la copie profonde lorsque l'on veut copier des objets dans une collection par exemple ?
b) Quels sont les avantages et/ou inconvénients de chacune de ces méthodes ?

Exercice 2 : Constructeurs / Destructeurs (6 points)

a) Qu'est ce qu'un constructeur ? (2 points)
   - Dans quelles circonstances est-il utilisé ?
   - Peut-on l'appeler comme une méthode normale ?
   - Quel est son type de retour ?
b) Pour quelle raison ne définit-on pas toujours de constructeur par défaut lorsqu'on a défini d'autres constructeurs ? (1 point)
c) En C++ on est parfois obligé de définir un constructeur par défaut, pour quelle raison ? (1 point)
d) 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 3 : Polymorphisme (4 points)

a) Qu'est ce que le polymorphisme et à quoi s'applique la notion de polymorphisme ? (1 point)
b) Sur quels critères les méthodes surchargées sont-elles distinguées les unes des autres ? Quels sont les critères qui ne sont pas pris en compte ? (1 point)
c) Expliquez la notion de lien dynamique. Cette notion est-elle disponible dans les langages comme Java et C++ et à quelles conditions ? (2 points)

Exercice 4 : Conception (8 points)

On cherche à concevoir une hiérarchie de classes en Java propre à décrire différents documents (situés sur un ordinateur ou bien sur Internet). Les documents sont donc référencés par une URL[1] et caractérisés par une taille en octets. Les documents peuvent être ouverts, lus et fermés. La manière d'ouvrir ou de fermer une URL est la même pour tous les documents, seule diffère la manière de les lire.

Remarque importante : La description ci-dessous n'est sans doute pas la description la plus exhaustive ni la plus efficace. C'est néanmoins celle qu'il vous est demandé d'employer pour cet exercice.

Les documents sont de plusieurs natures :

- Texte (avec et sans mise en forme) : ceux-ci sont caractérisés par la langue employée et l'encodage utilisé pour les caractères.
- Graphique : ceux-ci sont caractérisés par une largeur, une hauteur.
  - Images (fixes et mobiles (vidéo)),
  - Graphiques vectoriels (2D ou 3D) : dessins 2D ou objets 3D.

  Les graphiques bidimensionnels y ajoutent une résolution pour l'impression et les graphiques tridimensionnels y ajouteront une profondeur et un facteur d'échelle pour le rendu sur l'écran.
- Audio (enregistrement et synthèse)

Comme on peut le constater dans la liste ci-dessus, certains documents sont le résultat d'un enregistrement de la réalité (par un appareil photo, une caméra ou un microphone) et auront besoin d'être

[1] URL : Uniform Resource Locator : chaîne de caractère utilisée pour repérer un document sur Internet ou bien localement.

décodés alors que d'autres sont le résultat d'un rendu effectué à partir d'instructions comme les graphiques vectoriels, le texte mis en forme ou la synthèse sonore. D'autre part, les documents résultant d'un enregistrement possèdent un nom interne (un titre), un auteur, et une date d'enregistrement.

Certains de ces types de documents se déroulent dans le temps, ceux-ci sont donc caractérisés par une durée ainsi que par une fréquence d'affichage ou de rendu qui pourra être changée : C'est le cas notamment des sons enregistrés ainsi que de la vidéo (que l'on peut considérer comme une suite d'images enregistrées).

Et enfin, les documents enregistrés et possédant un aspect temporel peuvent être le résultat d'un flux d'information plutôt que stockés dans un fichier, on parlera alors de document en streaming. Le flux sera alors caractérisé par le débit (en bit/s) nécessaire à son acheminement.

Concevez une hiérarchie de classes/d'interfaces Java précédemment décrites en utilisant la notation UML vue en cours.

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 concrèt(e)s sont indiqués par un nom en texte droit.
- Les membres ou les classes abstrait(e)s sont indiqués par un nom en texte italique.
- On précisera pour les classes abstraites s'il s'agit d'interfaces ou de classes abstraites.

Exemple :