TD/TP n° 5

Pour le TP : on utilisera le logiciel Umbrello, sous Linux. À l'issue du TP, la production devra être envoyée sur EPREL : `M3105-CPOO` > `Travaux` > `Soumission du TP n° 5...`

Exercice 1. L'administration d'un *cloud*

On se propose de modéliser le fonctionnement d'un *cloud*. Un cloud est un système de fichier décentralisé devant garantir la permanence des données et leur disponibilité. Notre cloud ne se contente pas de stocker les données, puisqu'il vend également du temps de calcul aux clients.

Il fonctionne de la manière suivante. Un cloud dispose d'une IP et est constitué de grappes de machines. Une grappe pouvant elle-même être composée d'autres grappes, et ainsi de suite. À la base sont les machines, qui sont de deux types : des machines à grande vitesse et peu de stockage — appelons-les `MachineCPU` — et les machines à faible vitesse et grande capacité de stockage — appelons-les `MachineData`.

Chaque machine a un nom, une vitesse, et une capacité de stockage. De plus, le pourcentage de processeur et l'espace disque disponibles sont également stockés. En outre, elles connaissent la grappe à laquelle elles appartiennent. De même, une grappe connaît ses machines ou grappes filles, ainsi que sa grappe mère. Enfin, il est possible pour une grappe de connaître sa capacité de calcul et de stockage (libre ou théorique).

1. Construire le diagramme de classes de ce système.
2. Quelles opérations faudrait-il programmer pour calculer la capacité totale de stockage libre ? Donner le pseudo-code (l'algorithme) de ces opérations.

On souhaite maintenant pouvoir manipuler les machines en en créant de nouvelles pour les ajouter à une grappe. Cependant, créer une nouvelle machine est en fait une opération complexe. En effet, lorsqu'on crée une machine, le système ne permet que d'entrer ses caractéristiques de base : nom, espace de stockage, vitesse. Le système décide alors s'il considère que cette machine est faite pour le calcul ou le stockage.

3. Modifier le diagramme de classes pour pouvoir ajouter de nouvelles machines. On précise que les grappes ont la possibilité d'ajouter elle-même une nouvelle machine.
4. Construire le diagramme de séquences de l'ajout d'une nouvelle machine par une grappe.

Suite à un changement de fournisseur dans nos machines, nous allons devoir gérer un cloud dont les machines sont de deux marques différentes. Incompatibles entre-elles, les grappes devront ne comporter qu'une seule marque de machines.

5. Nous allons modifier le diagramme pour résoudre ce problème, tout en nous assurant qu'un autre changement de fournisseur ne nous dérangerait pas plus que ça.
   a) Modifier le diagramme pour prendre en compte les différents types de grappes et de machines existant.
   b) Comment alors ajouter une nouvelle machine ?
   c) Comment alors ajouter une nouvelle grappe ?
   d) Que faut-il abstraire pour que l'arrivée d'un autre fournisseur soit anodine ?
6. *(Bonus)* Quels patrons de conception ont été utilisés pour résoudre ce problème ?

Exercice 2. Du marketing mobile

On souhaite concevoir une application mobile de tests marketing. Le fonctionnement est le suivant : l'écran de l'utilisateur est divisé en deux cases qui affichent chacune une image, la championne, et la challenger. Lorsqu'il clique sur une image (sa préférée), l'image cliquée devient (ou reste) la championne, l'autre est remplacée par une nouvelle image (aléatoire). L'image championne, qui reste, change cependant de place : d'en haut, elle est affichée en bas, et vice-versa.

D'autre part, on souhaite compter à chaque changement de championne combien de temps la précédente l'est restée. On garde ainsi une suite de paires (entier,image) qui maintient le palmarès.

Périodiquement, un module des serveurs de l'application demandent le palmarès d'une image donnée. Comme ces serveurs choisissent le moment où la connexion paraît fiable, il est possible qu'ils envoient plusieurs demandes (presque) en même temps. L'image dont il est question étant choisie aléatoirement, il est également possible que plusieurs demandes de palmarès interviennent concomitamment.

Lorsque l'utilisateur en a assez et quitte le jeu, le palmarès global est envoyé aux serveurs.

1. Construire un diagramme de cas d'utilisation de ce système.
2. Construire un diagramme de séquence système de l'utilisation normale de l'application.
3. On veut construire un diagramme de classes de l'application. On suppose que l'on travaille avec un *framework* graphique qui fournit des `CaseImage`s, implémentant l'interface `Observer`, et auquel on peut attacher des `Cliqueur` par l'opération `addCliqueur`. Un cliqueur est une classe qui implémente une opération `onClick()`.
   a) Construire le diagramme des classes et interfaces mentionnées ci-dessus.
   b) Ajouter les classes correspondant au modèle des images et des cases championne et challenger.
   c) Ajouter les classes nécessaires au stockage du palmarès.
   d) Ajouter les classes nécessaire au parcours du palmarès.
   e) Ajouter les classes de communication avec les serveurs.
4. On souhaite à présent construire différent diagrammes de séquence. À cette occasion, on ajoutera au diagramme de classes les opérations (constructeurs, accesseurs, ...) dont on a besoin.
   a) Construire le diagramme de séquences de l'ouverture de l'application : il consiste principalement à créer l'affichage. (On suppose que les modèles des images existent déjà à l'ouverture.)
   b) Construire le diagramme de séquence d'un clic sur une image qui n'était pas championne.
   c) Construire le diagramme de séquence de la demande d'un palmarès particulier par les serveurs de l'application.

Exercice 3.

On reprend l'exercice 2 de la feuille de TD/TP n° 4. Voici quelques indices pour la première question.

— Tout d'abord, la base de données : disposant d'une unique instance, on voit vite quel patron de conception utiliser...

— Il s'agit avant tout désormais d'obtenir une classe `Personne` qui va contenir quelques informations. Parmi celles-ci, des liens de parenté relient les personnes. Or, si l'on veut définir un objet `Personne`, toutes ses associations doivent être définies (ou du moins si un lien est `null`, il doit l'être car le parent n'est pas référencé). On doit donc *abstraire* les personnes pour limiter l'accès à certains attributs qu'il contient. L'accès à ces attributs nécessitera d'abord des opérations (coûteuses) pour instancier réellement notre personne partielle en véritable personne.