Travaux Dirigés de Java n°2

Cours de Java

—Maitrise d'informatique—

octobre 2000

On souhaite manipuler des adresses physiques (dites "MAC") de cartes réseaux. Un fabricant de cartes réseaux est identifié par un nom, ou par un entier. Quand un fabricant construit une carte réseau, il lui attribue une adresse MAC. Une adresse MAC est un couple (i, n) où i est l'entier identifiant de façon unique le fabricant, et n le nombre de cartes déjà construites par le fabricant au moment de l'attribution de l'adresse MAC.

Le but du TD est l'implémentation de la classe `Vendor` (un vendor est un fabricant de cartes), de la classe `MacAddress`, puis d'une classe `Hexa` permettant de faire de l'affichage de valeurs entières positives en hexadécimal (cette classe sera utilisée pour l'affichage des adresses MAC).

▸ **Exercice 1.** *Ecrire une classe* `Vendor` *ayant pour attributs le nom du fabricant, un entier (représentant son numéro l'identifiant), et une méthode* `String toString()`. *Créez une instance de* `Vendor`, *et appliquez dessus la méthode* `toString()`. *Que constatez-vous ? Ajoutez à votre classe un constructeur* `Vendor(String nom, int identifiant)`. *Critiquez, et proposez une solution.*

▸ **Exercice 2.** *Modifiez la classe* `Vendor` *de telle manière qu'un numéro identifiant le fabricant lui soit automatiquement attribué à la création d'une nouvelle instance de* `Vendor`. *Lui ajouter une méthode* `same` *réalisant la comparaison de deux instances de* `Vendor`.

▸ **Exercice 3.** *Ecrire une classe* `MacAddress` *ayant pour attributs un fabricant, un numéro identifiant la carte pour le fabricant, et une méthode* `String toString()`, *retournant un objet de la forme* (i, n) *où* i *identifie le fabricant, et* n *est le numéro de carte. Lui ajouter une méthode* `same` *réalisant la comparaison de deux instances.*

▸ **Exercice 4.** *En Java, un* `long` *a la taille de deux* `int`. *Ajouter à la classe* `MacAddress` *une méthode* `long getAddress()` *retournant l'adresse MAC sous forme d'un long, dans lequel l'*`int` *de poids fort identifie le fabricant, et l'*`int` *de poids faible le numéro de série de la carte.*

▸ **Exercice 5.** *On souhaite maintenant que la méthode* `toString()` *de la classe* `MacAddress` *retourne un objet de la forme* $i : n$, *où* i *et* n *ont la même signification que dans l'exercice* *précédent, mais cette fois-ci sont exprimés en hexadécimal. Ecrire une classe* `Hexa` *comportant deux méthodes* `String toString(int)` *et* `String toString(long)`*, qui renvoient la représentation hexadécimale de leur argument, qui doit être positif (si l'argument est négatif la méthode lève une exception de la classe* `IllegalArgumentException`*). Comment faire pour que la classe* `Hexa` *ne soit pas instanciable ? Utilisez ces méthodes dans la méthode* `MacAddress.toString()`*.*