Université de Nice – Sophia Antipolis — Deug MASS MIAS SM 1re année
UEF1 : Informatique & Programmation — 2000–2001

TP N° 4

Objets – Méthodes d'instance

Ça va mieux en le révisant ... Un objet est un regroupement de valeurs qui modélisent une réalité quelconque (ex : machine, employé, point...). Une fois que la classe a décrit la structure de ses objets, on peut en construire autant que nécessaire. La classe fixe :

- le nom et le type des *données privées*[1]
- les *constructeurs*
- les *méthodes d'instance.*

Définir une classe d'objets revient à définir un nouveau type à partir des types existants. Pour créer un objet de cette classe, on alloue la place nécessaire en mémoire (en utilisant le mot-clef `new`) et en appellant un constructeur de la classe afin initialiser les données privées de l'objet.
Une fois créé, un objet dispose, comme tous les objets de la classe, des méthodes d'instance de la classe. On distingue trois types de méthodes d'instance :

- *les accesseurs* : fonctions retournant une des données privées
- *les modificateurs* : procédures mettant à jour une ou plusieurs des données privées
- les autres, au comportement varié (calcul, affichage, ...).

Les méthodes d'instance sont celles que l'on peut appliquer à un objet. Elles se différencient des méthodes de classe par l'absence du mot-clé `static`. Comme ces dernières, on les utilise au moyen de la notation pointée, mais elles permettent d'envoyer un message à un objet plutôt qu'à sa classe, comme dans le cours :

```
comp.changeNom("Pokémon");
```

Ceci représente un *message* envoyé à l'objet `comp` et pourrait se lire :

"Hé, comp, veux-tu bien changer ton nom en Pokémon !"

1 Chaînes de caractères

Et oui, vous le savez à présent, les chaînes de caractères sont des objets d'une classe prédéfinie JAVA : la classe `String` qui est automatiquement accessible. Par exemple, la chaîne de caractères :

```
"Mais 10 au carré donne-t-il 100 ?"
```

a pour longueur 33 : elle comporte 33 *caractères*, comprenant des lettres (majuscules et minuscules), des espaces, des ponctuations... Les caractères qui composent une chaîne sont numérotés de gauche à droite à partir de 0, donc dans cet exemple de 0 jusqu'à 32. Voici quatre méthodes

1. Notez que quelquefois, les données privées seront appelées des *attributs* ou même des *champs*...

d'instance *prédéfinies*[2] sur les chaînes de caractères :

`length()`	retourne la longueur d'une chaîne
`substring(i,j)`	retourne la sous-chaîne du rang i au rang j-1 inclus
`toUpperCase()`	retourne une copie de la chaîne tout en majuscules
`toLowerCase()`	retourne une copie de la chaîne tout en minuscules

On pourrait ajouter l'opérateur de concaténation `+` qui, dès lors qu'une de ses deux opérandes est une chaîne, transforme l'autre également en chaîne de caractères[3] puis les concatène (i.e. les met bout à bout). Rappelez-vous :

```
System.out.println("Volume de la sphère : V = " + volume);
```

Bien que les chaînes de caractères soient des objets, il y a assez bizarrement deux façons de définir une variable de type chaîne de caractères, selon que l'on utilise ou non le constructeur `String`(...) de la classe `String` :

```
String ch = "Java bien ?"; String ch = new String("Java bien ?");
```

Rappel : On peut lire un nombre entier avec `Console.readInt`(...), un nombre approché avec `Console.readDouble`(...), et une chaîne de caractères avec `Console.readLine`(...).

Exercice 1) Ecrivez un programme `TP4Exo1.java` calculant un mot de passe. Il commence par lire au clavier le nom, le prénom (deux chaînes) et l'âge de l'utilisateur (un entier). Ensuite, il fabrique et affiche un mot de passe i.e. une chaîne constituée de la première lettre du prénom, suivie de celle du nom (les deux réduites en minuscules) puis des chiffres de l'âge en sens inverse (on supposera[4] que l'utilisateur a 10 ans ou plus et moins de 100 ans !), d'un trait d'union, et enfin (on ne saurait être trop prudent) d'un entier aléatoire de $[1000, 9999]$.
Par exemple, Anne-Marie Foubarre qui a 19 ans pourrait avoir comme mot de passe `af91-5329`. ◇

N'oubliez pas de commenter un minimum vos programmes afin de les rendre encore plus lisibles. Outre le commentaire délimité par `/*...*/`, il existe aussi en JAVA des commentaires de fin de ligne, introduits par les caractères `//`. Petit exemple :

```
ch = ch + Numerik.randomInt(1000,9999); // ajout d'un entier aléatoire
```

2. Dans le jargon des informaticiens, on les appelle parfois *primitives*.

3. Pour obtenir automatiquement la représentation d'un objet sous forme de chaîne, l'opérateur + utilise la fonction `toString()` de la classe de cet objet.

4. C'est ce genre de supposition qui conduit à des problèmes comme le bug de l'an 2000...

2 Fabriquons des objets

Commençons donc par définir une nouvelle classe d'objets :

Exercice 2) a. Ecrivez une classe `Employé` destinée à modéliser chaque employé(e) d'une entreprise. Voici quelle doit être la structure de cette classe :

- deux données privées : la chaîne d'identification, formée du nom et du prénom de l'employé(e) ; un nombre de type `double` pour son salaire.
- contructeurs : un seul constructeur avec 2 arguments correspondant aux deux données privées.
- les méthodes d'instance : une méthode accesseur qui retourne la chaîne d'identification, une autre méthode accesseur pour le salaire et une méthode modificateur qui hausse le salaire de 10 %.

b. Passons à la création d'employés... Ecrivez une autre classe `TP4Exo2` munie d'une méthode `main`(...) dans laquelle vous aller créer l'employée Alice Dupont, au salaire de 12000 francs. Faites afficher ces informations, puis, après avoir opéré une hausse de salaire de 10%, faites afficher le nouveau salaire d'Alice, en francs puis en euros (utilisez une constante `EUROS = 6.55957`). ◇

Remarquez ces quatre usages de la notation pointée qui n'ont pas la même signification :

```
Math.random()
comp.décrisToi()
Math.PI
unsa.Console
```

Le premier point témoigne du fait que la méthode de classe `random()` s'adresse à la classe `Math`. Le second point indique que la méthode d'instance `décrisToi()` sollicite l'instance `comp` de la classe `Machine`. Le troisième permet d'accéder directement à la constante publique `PI` de la classe `Math`, sans passer par une méthode accesseur. Nous laisserons cette possibilité aux classes primitives de JAVA, nous interdisant de le faire nous-mêmes (de toutes façons, ce ne serait pas possible puisque nos données sont *privées* !). Enfin, le quatrième signifie que la classe `Console` fait partie de la bibliothèque de classes `unsa`[5].

Exercice 3) Que dire alors de chacun des deux points dans l'instruction suivante ?

```
System.out.println("Ciao!");
```

◇

5. Une telle bibliothèque se nomme en JAVA *package* (*paquetage* en français).

3 Approfondissement

Exercice 4) a. Créez une classe `Point` pour modéliser les points $\texttt{p}\binom{x}{y}$ du plan dans un repère orthonormé. Cette classe devra posséder deux constructeurs :

- l'un `Point(x,y)` qui initialise les données privées `px, py` avec les entiers x et y passés en arguments
- l'autre `Point()`, sans argument, qui initialise le point créé à l'origine du repère.

b. On vous propose ensuite d'écrire les accesseurs aux coordonnées `getX()`, `getY()` par exemple, les modificateurs `setX(x)`, `setY(y)`, `translate(dx,dy)`, la méthode `distance0()` qui retourne la distance du point à l'origine.

c. Pour utiliser toutes les fonctionnalités de la classe `Point`, nous vous proposons d'écrire une petite classe-test `TP4Exo4`. Vous pouvez par exemple tirer un point au hasard à proximité de l'origine, afficher ses coordonnées et sa distance à l'origine.

d. Toujours dans `TP4Exo4`, utilisez un `System.out.print(p)` pour afficher par exemple un point $\texttt{p}\binom{2}{-3}$ de votre classe `Point`, et notez l'affichage sur papier. Ajoutez ensuite une méthode d'instance `toString()` à la classe `Point` retournant une chaîne de caractères de la forme `"<2,-3>"` pour représenter ce même point. Votre méthode `toString()` a pris la main sur la méthode `toString()` invoquée implicitement par défaut pour tous les objets JAVA ! Constatez l'effet produit à présent par l'instruction :

```
System.out.print(p);
```

◇

Morale : A partir de maintenant, lorsque vous définirez une nouvelle classe d'objets, on vous demandera d'y inclure une méthode d'instance `toString()` retournant une chaîne de caractères. Celle-ci devra contenir les informations relatives à l'objet sous la forme de votre choix.