Séquence 1

Utiliser des objets

1.1 Qu'est-ce qu'un objet ?

Un objet est un composant logiciel qui contient des données ainsi que des traitements qui manipulent ces données. Les données sont stockées dans des variables privées de l'objet qui sont appelées variables d'instance ou attributs de l'objet. Les traitements sont appelées méthodes d'instance (souvent abrégé en méthodes tout court).

Un objet a une vie qui commence avec la création de l'objet au moyen d'une opération appelée instanciation. Cette création consiste à réserver un emplacement dans la mémoire de l'ordinateur dans lequel seront stockées les données de l'objet. À partir de cette création, l'objet restera toujours à cette place et sera seul à pouvoir utiliser la mémoire en question.

Une fois créé, l'objet peut être utilisé pour réaliser les traitements qu'il contient. Ces traitements peuvent consister en consultations des données, dans leur modification ou dans leur utilisation.

En Java, il n'y a pas d'instruction pour mettre fin à la vie d'un objet. On peut considérer que l'objet existe jusqu'à la fin de l'exécution du programme. Dans la réalité, il peut être détruit par le système d'exécution Java si celui-ci a la certitude que cet objet ne sera plus utile et qu'il a besoin de mémoire pour créer d'autres objets.

1.2 Exemple des dates

Nous allons voir comment créer des objets de la classe Date et les utiliser. La classe Date (java.util.Date pour donner son nom complet) est une classe prédéfinie de Java qui sert à représenter des dates. La classe Date est une classe Java qui sert à créer des objets, chacun des objet représentant une date.

Les données d'un objet date sont un tantième, un mois, une année. On peut supposer qu'il y a ces trois attributs dans la classe et qu'ils sont de type `int`.

Nous allons utiliser dans ce chapitre des éléments qui sont notés `deprecated` par Java, c'est-à-dire déconseillés et obsolètes, à ne pas utiliser. Nous allons quand même les utiliser parce qu'ils sont simples à comprendre. Cela produira des avertissements (*warnings*) lors de la compilation. Ce ne sont pas des erreurs et cela n'empêchera pas la compilation de se faire.

Voyons d'abord comment créer un objet au moyen d'une instanciation.

---

new Date(-111,6,14);

Dans cette instanciation, il y a d'abord l'opérateur `new` que l'on connaît déjà pour les tableaux, puis le nom de la classe et une liste de paramètres. Cette liste (ici trois entiers), est spécifique à la

classe `Date`.

Cela a pour effet de créer un objet dans la mémoire, correspondant à une certaine date. Pour savoir quelle date, on peut utiliser une méthode qui s'appelle *toString* et qui renvoie une chaîne résumant le contenu de l'objet. Par ailleurs, une fois l'objet créé, si l'on veut pouvoir l'utiliser, il faut le mettre dans une variable, soit directement (une variable du type `Date`) soit indirectement (par exemple en le stockant dans un `ArrayList` de dates).

Voyons donc un exemple de programme complet permettant de créer l'objet puis d'afficher la chaîne renvoyée par `toString`.

```
package useDate;
import java.util.Date;
public class CreerDate{
    public static void main(String[] args){
        Date unedate = new Date(-111,6,14);
        System.out.println(unedate.toString());
    }
}
```

Commentons ce code.

— Ligne 2 : pour utiliser la classe java.util.Date, il faut faire cette clause d'importation (comme par exemple pour un `ArrayList`).
— Ligne 5 : création de l'objet et stockage de cet objet dans la variable `unedate`.
— Ligne 6 : pour appeler la méthode toString, il faut d'abord donner l'objet, puis mettre un point, le nom de la méthode et ses paramètres entre parenthèses. Comme toString n'a pas de paramètres, il n'y a rien entre parenthèses.

Si on compile puis exécute ce programme, on a l'affichage suivant :

```
Tue Jul 14 00:00:00 CET 1789
```

La date est donc le 14 juillet 1789. Quels sont les paramètres donnés à l'instanciation : -111,6,14 ? Le premier est l'année moins 1900 (1789-1900=-111). Le second est le mois compté à partir de 0 (janvier est codé par 0 et juillet par 6). 14 est le tantième, c'est-à-dire le jour du mois.

Est-ce qu'il n'est pas étrange de compter les années -1900 et les mois à partir de 0 ? Si assurément et c'est probablement une des raisons pour lesquelles ce code est `deprecated`.

Si l'on regarde maintenant l'affichage produit, on voit qu'il n'y a pas que la date, mais aussi une suite de 0. C'est l'heure, car la classe représente non seulement un jour, mais une heure précise de ce jour et CET pécise le fuseau horaire utilisé (Central European Time).

1.3 Utilisation de l'objet

Les méthodes d'un objet ont toujours un rapport avec les données qu'il contient. On peut les répartir en plusieurs catégories :

— les méthodes qui utilisent les données.
— les méthodes qui modifient les données.
— les méthodes qui permettent de comparer plusieurs objets.

1.3.1 Utilisation des données

La méthode `toString` utilise les données pour créer une chaîne de caractère qui résume le contenu de l'objet.

Il existe d'autres méthodes qui accèdent à une seule donnée. Par exemple, la méthode `getMonth` renvoie un entier qui est le numéro du mois (compté à partir de 0). Il y a de même `getYear` qui renvoie l'année moins 1900 et la méthode `getDate` qui renvoie le jour du mois.

Voici un programme utilisant ces méthodes :

```
package useDate;
import java.util.Date;
public class MethodesAccesDate{
    public static void main(String[] args){
        Date unedate = new Date(-111,6,14);
        System.out.println("Année:␣" + unedate.getYear());
        System.out.println("Mois:␣" + unedate.getMonth());
        System.out.println("Jour:␣" + unedate.getDate());
    }
}
```

Exécution :

```
> java useDate.MethodesAccesDate
Année: -111
Mois: 6
Jour: 14
```

On voit que les trois méthodes appelées permettent de retrouver les trois paramètres donnés à la création de l'objet.

1.3.2 Modification des données

Il existe dans la classe `Date` trois méthodes permettant de changer respectivement l'année, le mois et le jour. Elles s'appellent `setYear`, `setMonth` et `setDate`. Voyons l'exemple de l'exécution de l'une d'entre elles.

```
package useDate;
import java.util.Date;
public class MethodeSetDate{
    public static void main(String[] args){
        Date unedate = new Date(-111,6,14);
        System.out.println(unedate.toString());
        unedate.setYear(33);
        System.out.println(unedate.toString());
    }
}
```

Exécution :

```
> java useDate.MethodeSetDate
Tue Jul 14 00:00:00 CET 1789
Fri Jul 14 00:00:00 CEST 1933
```

On voit que l'appel à la méthode `setYear` a effectivement changé l'année de l'objet.

Un autre effet de la méthode a été de changer le fuseau horaire qui est devenu CEST : Central Europe Summer Time. C'est dû au fait qu'il y avait en 1933 une heure d'été et pas en 1789.

Une chose importante distingue la méthode `getYear` de la méthode `setYear` : la première renvoie un résultat, un nombre entier et ne prend pas de paramètre. Un entier sort de l'objet et rien n'y rentre. Pour la méthode set, c'est le contraire : elle ne renvoie pas de résultat, mais elle a un paramètre. La première est utilisée dans une expression, la seconde est seule sur une ligne de code. La méthode `setYear` a pour effet de modifier la mémoire privée de l'objet sur lequel elle est appelée.

1.3.3 Comparaison de dates

Parmi les opérations sur les dates, il y a en plusieurs qui permettent de comparer deux dates pour savoir si l'une est avant l'autre ou pas. Ce sont des opérations qui prennent en compte deux dates donc deux objets différents. Or une méthode ne peut être appelée que sur un seul objet. Ces méthodes de comparaison vont comparer l'objet sur lequel elle sont appelées et un deuxième objet passé en paramètre.

Il y a la méthode `before` qui teste si la date sur laquelle elle est apelée est avant celle passée en paramètre. Elle renvoie un résultat booléen et elle peut donc être utilisée comme condition d'une boucle ou d'un `if`.

```
package useDate;
import java.util.Date;
public class ComparaisonDate{
    public static void main(String[] args){
        Date revolution = new Date(-111,6,14);
        Date decouverte_amerique = new Date(1492-1900,9,12);
        System.out.print("revolution.before(decouverte_amerique):␣");
        System.out.println(revolution.before(decouverte_amerique));
        System.out.print("decouverte_amerique.before(revolution):␣");
        System.out.println(decouverte_amerique.before(revolution));
        System.out.print("revolution.before(revolution):␣");
        System.out.println(revolution.before(revolution));
    }
}
```

Exécution :

```
> java useDate.ComparaisonDate
revolution.before(decouverte_amerique): false
decouverte_amerique.before(revolution): true
revolution.before(revolution): false
```

1.4 Récapitulation et terminologie

- Un objet est créé par l'opérateur `new` suivi du nom d'une classe suivi d'une liste de paramètres entre parenthèses.
- Le nombre et le type des paramètres dépend de la classe.
- Les paramètres servent à initialiser l'objet.
- Un objet peut être stocké dans une variable ayant pour type le nom de la classe utilisée pour créer l'objet.
- Une méthode est appelée en précisant d'abord l'objet à laquelle elle appartient, puis le nom de la méthode, puis les paramètres entre parenthèses.

— Une méthode peut renvoyer un résultat ou pas.
— Certaines méthodes consultent les données de leur objet, d'autres les modifient, d'autres concernent plusieurs objets.
— Quand une méthode utilise plusieurs objets (par exemple pour comparer deux objets), un des objet est celui sur lequel la méthode est appelé (l'objet situé avant le point), les autres sont passés en paramètre.

Terminologie :

— Instanciation : création d'un objet à partir de sa classe.
— Instance d'une classe : objet créé en utilisant cette classe. Par exemple, `undedate` est une instance de `Date`.
— Attribut : variable locale à une objet, aussi appelé variable d'instance, qui permet de stocker une donnée qui est une valeur Java.
— Invocation de méthode : portion de programme qui consiste à appeler une méthode sur un objet avec des paramètres.
— Accesseur : méthode qui permet de consulter la valeur d'un attribut. Ce genre de méthode a souvent pour nom `get` suivi du nom de l'attribut (par exemple : `getYear`). On les appelle aussi *getter*.
— Modificateur : méthode qui permet de modifier la valeur d'un attribut. Ces méthodes ont souvent pour nom `set` suivi du nom de l'attribut (par exemple : `setYear`). On les appelle aussi *setter*.