Résumé

Déclaration de classes

Déclaration simple

On écrit le mot-clé class suivi du nom de la classe puis d'un bloc entre accolades contenant les déclarations de champs et de méthodes.

```
class NomDeLaClasse {
 ...;
}
```

Toute classe hérite de façon implicite de la classe Object qui est donc à la racine de toute arborescence de classes.

Champs

Les données d'une classe sont contenues dans des champs qui sont des variables. La déclaration se fait en donnant le type du champ suivi de son nom.
Par exemple :

```
class ajout {
 double raison;
}
```

Méthodes

La déclaration se fait en donnant le type de retour suivi du nom et de parenthèses entourant les paramètres. Attention, les parenthèses doivent être présentes même s'il n'y a pas de paramètres.
La déclaration est suivie d'un bloc d'instructions entre accolades qui définit ce que fait la méthode. La valeur de retour est introduite par le mot-clé `return`. Celui-ci peut être omis si le type de retour est `void`.
Par exemple, si nous ajoutons la méthode image à la classe ajout :

```
class ajout {
 double raison;
 double image(double d) {
  return d+raison;
 }
}
```

Les paramètres sont toujours transmis par valeur pour les types de données de base. Par contre les instances passées en paramètres sont transmises par référence et sont donc susceptibles d'être modifiées si leur contrôle de visibilité le permet.

Constructeurs

Les constructeurs permettent d'initialiser les champs d'une classe. On les déclarent en utilisant le nom de la classe suivi de parenthèses entourant des paramètres.
Par exemple :

```
class ajout {
 double raison;
 ajout(double d) {
  raison=d;
 }
 double image(double d) {
```

```
    return d+raison;
  }
}
```

Héritage

Une classe peut être dérivée d'une classe préexistante qui est alors sa classe parente. La déclaration utilise alors le mot-clé `extends`. Par exemple :

```
class NomDeLaClasse extends ClasseParente {
 ...;
}
```

La classe dérivée hérite des champs et des méthodes visibles de la classe parente. Elle peut aussi surcharger certaines méthodes.

Contrôle de visibilité

Il existe 4 niveaux de protection des champs et des méthodes d'une classe.

1. `package`
   C'est le niveau par défaut (pas de déclaration à faire). La visibilité est limitée aux classes du package courant. Les champs sont accessibles en lecture et écriture.
2. `public`
   La déclaration est introduite par le mot-clé public. Il n'y a aucune restriction d'accès.
3. `protected` La déclaration est introduite par le mot-clé protected. Les champs et méthodes déclarés protected sont visibles par toutes classes du package courant et toutes les sous-classes qu'elles fassent partie ou non du package.
4. `private`
   La déclaration est introduite par le mot-clé private. Seule la classe peut voir les champs (encapsulation des données) et les méthodes déclarés private. En général on prévoit des méthodes publiques d'accès aux données.

Mots-clés final et static

Mot-clé static

Il permet d'introduire des *variables et des méthodes de classe* par opposition aux *variables et aux méthodes d'instance*.
Les variables et méthodes de classe peuvent être utilisées sans création d'une instance de classe.
La méthode main d'une classe est toujours static.

Mot-clé final

Il a l'effet suivant :

- pour une classe, il interdit d'en hériter
- pour une variable, il la rend constante
- pour une méthode, il interdit de la redéfinir dans une sous-classe

Note : les méthodes private sont déjà final.

Constantes

L'association des mots-clés static et final permet de définir des constantes de classe. Par exemple :

```
public static final int an=2000;
```

Création d'instances

Mot-clé new

La création d'une instance se fait au travers d'un new qui procède à l'allocation en mémoire et à l'appel d'un constructeur.
L'allocation en mémoire est dynamique et un garbage collector se charge de récupérer la mémoire allouée qui n'est plus référencée.

Quand on fait appel à un constructeur sans paramètres, une instance de base est créée avec des valeurs par défaut.
Quand le constructeur attend des arguments, ils déterminent les valeurs initiales des (tous ou certains) champs.

Accès aux champs et aux méthodes

On accède aux champs et aux méthodes en utilisant la notation pointée.

Pour les champs et les méthodes déclarés static on utilise le nom de la classe. Par exemple, avec la classe Math :

```
double mon_pi=Math.PI;
```

Dans les autres cas on utilise le nom d'une variable d'instance. Par exemple, en utilisant la classe ajout :

```
ajout a=new(12.5);
double d=a.raison;
double rep=a.image(5);
```

Mots-clés this et super

Le mot-clé this référence l'instance courante.

Le mot-clé super référence la classe parente de la classe courante. On l'utilise lorsqu'on surcharge des constructeurs ou des méthodes de la classe parente.

Retour au menu