Programmation Orientée Objet

Vincent Despré

Polytech Nancy
2ième année de Peip

Types

Objet

Héritage

- Mémoire: des tas d'octets.
- Un octet = un nombre?
- $00010010 = 1*16+0*8+0*4+1*2+0*1 = 18$.

Exercice

Écrire en binaire: 13, 121 et 2050.

Complément à 2

Si le premier bit est 1 alors le nombre est négatif et a la valeur manquante pour atteindre 2^k où k est le nombre de bit disponibles.

Exercice

On se place en complément à 0 en 8 bits combien valent: 00001110, 11001010 et 11111111.

Héxadécimal

On souhaite encoder les nombres en base 16 (32 pour le binaire et 10 pour le décimal). On a alors les chiffres suivants: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F.

Exercice

Écrire en hexadécimal: 13, 121 et 2050.

Taille des entiers

On peut encoder les entiers sur 32 ou 64 bits naturellement mais aussi sur d'autres nombres de bits en théorie.

Exercice

Quels sont les valeurs maximales pour des entiers codés en 32 bits et 64 bits.

Types

Objet

Héritage

- Mantisse: le nombre en lui-même.
- Exposant: le décalage.
- Lien:

https://fr.wikipedia.org/wiki/IEEE_754

Types

Objet

Héritage

- En général codés sur un ou deux octets.
- Code Ascii:
<https://en.wikipedia.org/wiki/ASCII>
- Différents encodages, UTF-8:
<https://en.wikipedia.org/wiki/UTF-8>

Un objet est un "super type" qui définit une notion avec les éléments suivants:

- **Attributs:** les variables sauvegardées qui définissent les instances de l'objet.
- **Constructeurs:** la fonction qui va créer des instances de l'objet (typiquement en initialisant les attributs).
- **Méthodes:** des fonctions qui sont appelées depuis une instance concrète et qui correspondent à l'utilisation de l'objet).

Une classe n'est qu'un plan!

Une classe présente un concept mais ne le matérialise pas. Pour cela il faut créer des instances de la classe dans la partie du code qui s'exécute à proprement parler (méthode main en java).

Pour créer une instance d'une classe X , il faut faire la manipulation suivante:

$$X \text{ mon}X = \text{new } X(12);$$

- Le premier X est le type de la variable.
- $\text{mon}X$ est le nom de la variable. On parle aussi d'une instance de la classe X .
- new annonce un constructeur.
- Le second X est le constructeur qui prend en entrée un int dans l'exemple.

Principe d'encapsulation

Types

Objet

Héritage

- Plusieurs niveaux de portée: private, public, protected...
- Les attributs toujours en privé!
- Accès aux attributs: getter (souvent) et setter (rarement).

Spécificités Java:

- Portée par défaut: package private.
- Méthode d'affichage *String toString()*.

Plusieurs méthodes (en particulier les constructeurs) peuvent partager un même nom dans les conditions suivantes:

- Le type de retour est le même.
- Le type/nombre des arguments est différent.
- La sémantique est similaire.

Cas des constructeurs:

- Il est souvent utile d'avoir un constructeur par défaut sans argument.
- Il ne faut pas hésiter à créer de nombreux constructeurs car c'est une partie importante de l'interface pour les utilisateurs.

Principe de l'héritage

Types

Objet

Héritage

Une classe peut hériter d'une autre classe. Ce faisant, elle obtient naturellement une certaine quantité de propriétés:

- Les attributs de la classe mère.
- Les méthodes de la classe mère hors constructeurs.
- Les constructeurs doivent être **redéfinis** mais ont accès au code de la classe mère grâce au mot clef super.

Redéfinition et surcharge

On peut surcharger une méthode à l'intérieur d'une classe mais on redéfinit une méthode de la classe mère dans la classe fille (on parle souvent d'override).

Pourquoi hériter:

- Éviter la duplication de code.
- Donner la responsabilité d'une tâche à une classe pour laquelle ça semble naturelle.
- Clarifier la structure du code.

Exemple:

- Véhicule <- 2Roues <- Vélo <- VéloCross ...
- Matrice <- MatriceSymmetric, MatriceSparse, MatriceNilpotente ...

Principe

On peut créer une instance avec un type grâce à un constructeur d'un type qui hérite du type principal.

Exemple:

```
Vehicule monVehicule = new Moto();
```

- La variable a pour type *Vehicule*.
- Seules les méthodes déclarées dans la classe *Vehicule* peuvent être utilisées.
- Quand une méthode est appelée, elle est appelée avec la redéfinition de la classe fille *Moto*.

Interface:

- Seules les déclarations des méthodes sont autorisées.
- Pas de code, pas de constructeur, pas d'attribut.
- Une interface ne peut pas être instanciée directement.

Classe abstraite:

- On peut ajouter des attributs mais pas de constructeurs.
- Les méthodes peuvent être abstraites ou implémentées mais au moins une doit être abstraite.
- Une classe abstraite ne peut pas être instanciée directement.

Pas d'instance mais un type valable!

On peut créer une instance avec le type d'une interface ou d'une classe abstraite en utilisant le polymorphisme.