Humanbooster

Algorithmie

07 – Les objets partie 2 – Les méthodes

DWWM PE6 LYON

Barthélémy DELUY – CC-BY-NC-SA 4.0

Introduction 1/2

Après l'introduction des objets simples (parfois appelés « structures »), un nouveau paradigme de programmation est apparu : la programmation objet.

En programmation impérative, on déclare les types, puis les actions possibles sous formes de procédures et de fonctions.

Cela force à passer en paramètre toutes les informations dont la routine a besoin. Cela complique également le développement : si j'ai une fonction « peindreVoiture(Voiture v, var couleur) », je dois aussi déclarer peindreAvion(Avion a, var couleur) , peindreMoto(Moto m, var couleur), etc. Je vais donc devoir multiplier les fonctions suivant les différents types de variable à prendre en charge.

Introduction 2/2

La programmation objet « renverse la charge de la preuve » : dorénavant, on va déclarer les actions possibles sur un objet directement dans celui-ci.

J'aurai donc une fonction changerCouleur(var couleur) dans mon type Voiture, et dans mon type Avion, et dans mon type Moto.

Ainsi, au lieu d'écrire :

```
Déclarer v : Voiture
v ← peindreVoiture(v, blanc)
```

Je vais écrire :

```
Déclarer v : Voiture
v.changerCouleur(blanc)
```

Sommaire

1. Les méthodes 1/4

On appelle « méthode » une routine déclarée à l'intérieur d'une classe. Elle pourra être appelée comme les attributs, à l'aide du . en général, ou de la flèche -> en PHP.

```
Type Voiture
    Déclarer couleur
    PROCÉDURE changerCouleur(var c)
    DÉBUT
        couleur ← c
    FIN
FinType
```

1. Les méthodes 2/4

En PHP :

```
class Voiture {
    public $couleur ;

    public function changerCouleur($c) {
        $couleur = $c ;
    }
}
```

Utilisation :

```
Déclarer v ← nouvelle Voiture()
v.couleur ← rouge
v.changerCouleur(blanc)
Afficher v.couleur          # affiche « blanc »
```

1. Les méthodes 4/4

La méthode « changerCouleur » est très simple, et on peut changer facilement la couleur via l'attribut public couleur.

Mais une méthode peut réaliser un traitement plus complexe :

```
Type Apprenant
    Déclarer notes[]
    FONCTION calculerMoyenne()
        ...
    FIN
FinType
```

2. Le mot-clé this

En programmation, le mot-clé this permet d'accéder aux attributs de l'objet, pour lever toute ambiguité :

```
Type Voiture
    Déclarer couleur
    PROCÉDURE changerCouleur(var couleur)
    DÉBUT
        this.couleur ← couleur
    FIN
FinType
```

Par défaut, quand on ne précise pas this et qu'une variable a le même nom qu'un attribut, alors l'ordinateur prend la valeur de la **variable**.

3. Le constructeur

Le constructeur est une méthode spéciale, qui est appelée automatiquement quand on utilise « new ».
Il permet de « pré-construire » l'objet avec des valeurs par défaut :

```
Type Voiture
    Déclarer couleur
    Déclarer nombresDeRoues
    PROCÉDURE constructeur(var couleur)
    DÉBUT
        this.couleur ← couleur
        this.nombreDeRoues ← 4
    FIN
FinType
Afficher (nouvelleVoiture("blanc")).nombreDeRoues       # Affiche « 4 »
```

Mise en application

1. Écrire la classe Apprenant qui a comme attribut un nom et un liste de notes, et comme méthode une méthode calculerMoyenne() qui retourne sa moyenne, et une méthode afficher() qui affiche son nom et sa moyenne.
2. Écrire la classe Carte qui a comme attribut une matrice de cases, et un trésor qui a des coordonnées [x,y] sur la carte (on suppose qu'il est bien sur la carte). Écrire la classe Aventurier qui a un nom et des coordonnées, et qui a une méthode « déplacer(x, y) qui permet de le déplacer sur la carte.
3. Modifier la classe Aventurier pour vérifier qu'on essaie de déplacer l'aventurier sur une case adjacente (haut, gauche, bas, droite), sinon afficher une erreur
4. Modifier la classe Aventurier pour lui permettre de se déplacer en diagonale
5. Créer un programme principal qui permet de déclarer une carte avec un trésor placé aléatoirement, un aventurier sur une position aléatoire, et de le faire chercher le trésor.

Conclusion