digitalent
FORMATION

Le langage Java

Méthode & Constructeurs

Message et méthode

- Message
  - précise le comportement à exécuter d'un objet
  - doit «appartenir» à l'interface de l'objet
  - est envoyé à un objet par un autre objet
- Méthode
  - définit comment le comportement est exécuté
  - peut accéder aux données de l'objet
  - à **un** message envoyé correspond **une** méthode

Message

- Composé
  - d'un **sélecteur**
  - d'**arguments** (optionnels)
    - les arguments sont des objets
- Envoyé à un receveur
- L'objet receveur retourne souvent un résultat
- Le message ignore le détail de l'exécution

Méthode

- Exécutée en réponse à un message
- Script définissant la réponse
  - code
  - séquence d'expressions exécutables
- Possède le même nom que le message qui l'a déclenché

Exécution d'une application

- Une classe peut être exécutée
  - si elle comporte une méthode *main*
  - ou si c'est une sous-classe d'**Applet**

```
public static void main(String[]
  args){
  // code à exécuter
}
```

Déclaration des méthodes (1)

```
[modificateurs] typeRetour nom(typeArg nomArg…){
   code…
}
```

- Méthodes d'instance
  - appliquées aux instances
  - **`float calculerTaux(int n) {…}`**
- Méthodes statiques
  - appliquées à la classe ou à ses instances
  - **`static String transformer() {…}`**

Déclaration des méthodes (2)

digitalent FORMATION

- Arguments
  - passés par *valeur* pour les types primitifs
  - Passés par référence pour les objets
- Pas de retour attendu
  - type de retour **void**
  - l'expression **return;** peut être utilisée pour une sortie explicite avant la fin de la méthode
- Retour d'un objet à la fin d'une méthode
  - mot-clé **return** suivi de l'objet
  - éventuellement **return null**

Comment envoyer un message ?

- Utilisation du caractère **.** (point)
- Méthodes d'instance
  - envoi à une instance (via sa variable)
  - **`obj.calculerTaux(5);`**
  - **`this.envoyer();`** (this sert à référencer l'objet courant)
- Méthodes statiques
  - envoi à une classe ou une instance
  - **`ExempleDeClasse.transformer();`**
  - **`obj.transformer();`**

Constructeur

- Méthode *spéciale* sans type de retour
- Initialisation automatique d'un objet
- Appelé lors de la création d'un objet

Définition d'un constructeur

digitalent FORMATION

- Même nom que la classe
- Nombre quelconque d'arguments
- Plusieurs constructeurs par classe
  - par surcharge
  - signatures différentes
- Pas de valeur de retour

Constructeur par défaut

- Constructeur sans argument
- Défini implicitement dans chaque classe
  - sans traitement spécifique autre que la création d'instance
  - peut être redéfini pour ajouter des traitements spécifiques
- « Disparaît » dès qu'un autre constructeur est défini dans la classe
  - il faut, si nécessaire, le redéfinir explicitement

- Assistant de création avec Eclipse
  - le constructeur par défaut peut être défini explicitement (paramétrable lors de la création de la classe)

Exemple de constructeurs

```
class CartePostale {
    Photo image;
    String texte;
    Adresse adresse;

    CartePostale () { /* redéfinition */
      texte = "Nous passons de bonnes vacances";
  }

  CartePostale(String texte) { /* surcharge */
    this.texte = texte;
  }
}
```

Appel d'un constructeur

- Pour créer un objet

```
CartePostale carte = new CartePostale();
CartePostale carteRusse = new
  CartePostale("Bons baisers de Russie");
```

- Par un autre constructeur (mot-clé this)

```
CartePostale(String texte) {
   this();
   this.texte = texte;
}
```

Destruction d'objet

- Automatique en Java
  - une fois que l'objet n'est plus référencé
  - pas de destructeur (C++)
- Effectuée par le *garbage collector*
  - libère la mémoire
  - processus de faible priorité
- Envoi du message *finalize()* lorsque l'objet est sur le point d'être détruit