JAVA pour les Kids

**public** class BonjourMonde {

**public** **static** void main(String[] args) {

System.out.println("Bonjour Monde »);

}

}

La classe BonjourMonde ne possède qu'une méthode main(), qui est le point d'entrée d'une *application* (programme) Java. On peut dire que main est une méthode parce qu'il y a des parenthèses après le mot main. Les méthodes peuvent *appeler (call*), c'est-à-dire utiliser, d'autres méthodes. Par exemple, notre méthode main() *appelle la méthode* println() pour afficher le texte "Bonjour Monde" à l'écran.

Toutes les méthodes commencent par une *ligne de déclaration* appelée *signature de la méthode* :

• Qui a accès à la méthode - public. Le mot-clé public signifie que la méthode main() peut être utilisée par n'importe quelle autre classe Java ou par Java lui-même.

• Comment utiliser la méthode - static. Le mot-clé static signifie qu'il n'est pas nécessaire de créer une instance (une copie) de l'objet BonjourMonde en mémoire pour pouvoir utiliser cette méthode. Nous reviendrons sur les instances de classe dans le prochain chapitre.

• La méthode *retourne-t-elle* des données ? Le mot-clé void signifie que la méthode main() ne retourne aucune donnée au programme appelant, qui en l'occurrence est Eclipse. Mais si on prenait l'exemple d'une méthode effectuant des calculs, celle-ci pourrait retourner un résultat à son appelant.

• Le nom de la méthode est main.

• La liste des arguments - des données qui peuvent être fournies à la méthode - String[] args. Dans la méthode main(), String[] args signifie que la méthode peut recevoir un *tableau (array)* de chaînes de caractères (String) qui représentent du texte. Les valeurs qui sont passées à la méthode sont appelées *arguments.*

*Les noms de méthodes Java sont toujours suivis par des parenthèses. S'il n'y a rien entre les parenthèses, cela signifie que la méthode ne prend pas d’arguments.*

*La plupart du temps, un programme ne pourra utiliser une classe Java qu'après qu'une instance en ait été créée.*

Les *variables* Java peuvent représenter les attributs d'une classe, les arguments d'une méthode ou être utilisées à l'intérieur d'une méthode pour stocker temporairement certaines données. Les variables doivent être déclarées avant de pouvoir être utilisées.

Si tu n'initialises pas les variables, Java le fera pour toi en donnant la valeur 0 à chaque variable numérique, la valeur false aux variables de type boolean.

Il y a aussi le mot-clé spécial final ; s'il est utilisé dans la déclaration d'une variable, on ne peut affecter de valeur à cette variable qu'une fois, cette valeur ne pouvant plus être modifiée par la suite. Dans certains langages, les variables invariantes sont appelées *constantes*. En Java les noms des variables invariantes sont généralement écrits en lettres majuscules :

**final** String CAPITALE\_ETAT = « Washington";

Voyons maintenant la signature de la méthode dormir():

**public** void dormir()

Elle nous indique que cette méthode peut être appelée depuis n'importe quelle autre classe Java (public) et qu'elle ne retourne aucune donnée (void). Les parenthèses vides signifient que cette méthode ne prend pas d'argument, parce qu'elle n'a besoin d'aucune donnée du monde extérieur - elle affiche toujours le même texte.

La signature de la méthode dire() est celle-ci :

**public** String dire(String unMot)

Cette méthode peut aussi être appelée depuis n'importe quelle autre classe Java, mais elle doit retourner un texte, ce qu'indique le mot-clé String devant le nom de la méthode. Par ailleurs, elle attend une donnée textuelle de l'extérieur, d'où l'argument String unMot.

Si une méthode retourne une valeur, tu dois l'appeler d'une autre façon. Déclare une variable du même type que la valeur retournée par la méthode. Tu peux maintenant appeler cette méthode :

String réactionAnimal;

réactionAnimal = monAnimal.dire("Cui !! Cui !! »);