UNIVERSITE VIRTUELLE DU SENEGAL

foo nekk foofu la

Langage Java

Mama AMAR

Séquence 1 : Présentation du langage

ANALYSE DE L'APPLICATION JAVA HELLOWORLD

Maintenant que vous avez vu le programme ***HelloWorld*** (et peut-être même compilé et exécuté), vous demandez peut-être comment cela fonctionne. Voici encore son code:

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Bonjour le monde !"); // Affiche le message.
    }
}
```

Le programme java ***HelloWorld*** se compose de trois parties principales: les commentaires de code source, la définition de classe ***HelloWorld*** et la méthode principale (main). L'explication suivante vous fournira une compréhension de base du code, mais les implications plus profondes ne deviendront apparentes qu'après avoir complété le reste du cours.

I. Commentaires sur le code source

Le texte en gras suivant définit les commentaires de l'application ***HelloWorld*** :

```
/**
 * La classe HelloWorld implémente une application qui
 * imprime simplement "Hello World!" à la sortie standard.
 */
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Bonjour le monde !"); // Affiche "Bonjour le monde".
    }
}
```

Les commentaires sont ignorés par le compilateur mais sont utiles aux autres programmeurs. Le langage de programmation Java prend en charge trois types de commentaires:

```
/* texte */
```

Le compilateur ignore tout de / * à * /.

```
/** Documentation */
```

Cela indique un commentaire de documentation (commentaire doc, pour faire court). Le compilateur ignore ce genre de commentaire, tout comme il ignore les commentaires qui utilisent / * et * /. L'outil javadoc utilise des commentaires doc lors de la préparation de la documentation générée automatiquement. Pour plus d'informations sur javadoc, consultez la documentation de l'outil Javadoc ™.

```
// texte
```

Le compilateur ignore tout // de la fin de la ligne.

II. La définition de classe HelloWorld

Le texte en gras suivant commence le bloc de définition de la classe pour l'application HelloWorld application:

```
/**
 * La classe HelloWorld implémente une application qui
 * imprime simplement "Bonjour le monde !" à la sortie standard.
 */
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Affiche "Bonjour le monde".
    }
}
```

Comme indiqué ci-dessus, la forme la plus basique d'une définition de classe est:

```
class nomClasse {
    . . .
}
```

Le mot clé ***classe*** de commence la définition pour un nom de classe nommé nomClasse, et le code de chaque classe apparaît entre les accolades d'ouverture et de fermeture marquées en gras ci-dessus. Le chapitre 3 donne un aperçu des classes en général et le chapitre 4 discute des classes en détail. Pour l'instant, il suffit de savoir que chaque application commence par une définition de classe.

III. La méthode principale (main)

Le texte en gras suivant commence la définition de la méthode principale:

```
/**
 * La classe HelloWorld implémente une application qui
 * imprime simplement "Bonjour le monde !" à la sortie standard.
 */
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Affiche "Bonjour le monde".
    }
}
```

Dans le langage de programmation Java, chaque application doit contenir une méthode principale dont la signature est:

```
public static void main(String[] args)
```

Les modificateurs ***public*** et ***static*** peuvent être écrits dans n'importe quel ordre (public static ou static public), mais la convention est d'utiliser public static comme indiqué ci-dessus. Vous pouvez nommer l'argument comme vous voulez, mais la plupart des programmeurs choisissent "args" ou "argv".

La méthode principale est similaire à la fonction principale en C et C ++; ***C'est le point d'entrée de votre application*** et, par la suite, invoque toutes les autres méthodes requises par votre programme.

La méthode principale accepte un seul argument: un tableau d'éléments de type String.

```
public static void main(String[] args)
```

Ce tableau est le mécanisme par lequel le système d'exécution transmet les informations à votre application. Par exemple:

```
java MyApp arg1 arg2
```

Chaque chaîne du tableau s'appelle un argument de ligne de commande. Les arguments de ligne de commande permettent aux utilisateurs d'affecter le fonctionnement de l'application sans la recompiler. Par exemple, un programme de tri peut permettre à l'utilisateur de spécifier que les données doivent être triées dans l'ordre décroissant avec cet argument de ligne de commande:

```
-descending
```

L'application ***HelloWorld*** ignore ses arguments de ligne de commande, mais vous devez être conscient du fait que ces arguments existent.

Enfin, la ligne:

```
System.out.println("Bonjour le monde !");
```

utilise la classe ***System*** de la bibliothèque principale pour afficher le message "Bonjour le monde !" à la sortie standard. Des parties de cette bibliothèque (également appelée "Application Programming Interface", ou "API") seront discutées tout au long du reste du cours.