Introduction à la programmation en JAVA

technofuturTIC

WAVENET
Intelligent IT Solutions

Table des matières

technofuturTIC

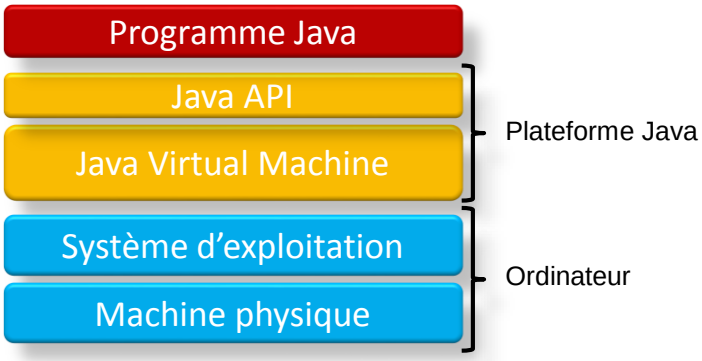
© Wavenet 2015

WAVENET
Intelligent IT Solutions

API Java

Aperçu du chapitre

I . Rappel



I . Rappel

L'API Java (Application Programming Interface) est constitué de librairies (packages) qui reprennent des ensembles fonctionnels de composants:

- data type, objets, string, I/O, date, ….
- Applet
- Abstract Window Toolkit (AWT)
- Sécurité
- Networking
- Internationalisation
- …

Il s'agit d'**outils** mis à la disposition du programmeur pour lui permettre de développer ses applications à base de Java

Aperçu du chapitre

II . Le package java.lang

Le **package java.lang** contient toutes les classes et les interfaces **fondamentales** en **Java**.

Cela inclut les classes de bases qui constituent la **hiérarchie** de classes, les **exceptions** , les fonctions **mathématiques**, les collections, …

Ce package est **par défaut importé** dans chaque nouveau fichier source même s'il n'est pas fait mention de la clause d'import (celui-ci étant implicite).

Object	Classe mère de toutes les classes.
Enum	Classe de base des énumérations
Class	Classe de base utilisée pour faire de la reflection en Java
Throwable	Classe de base de la hiérarchie des exceptions en Java.
Error, *Exception*, *RuntimeException*	Classes de base pour chacun des types d'exception
Thread	Classe permettant les opérations sur les threads.
String	Classe des chaînes de caractères.
StringBuffer et *StringBuilder*	Classes servant à réaliser des manipulations sur les chaînes de caractères
Comparable	Interface pour réaliser des comparaisons et des tris sur les objets
Math	Classe fournissant les outils mathématiques de base

Aperçu du chapitre

III . La classe Class

Classe de description d'une classe.
Elle permet de **manipuler une classe** dans un programme Java.

Deux manières de procéder pour instancier un objet de type classe, voici deux :

1. A partir d'un objet:

```
MaClasse objetMaClasse = new MaClasse();
Class objetClass = objetMaClasse.getClass();
```

2. A partir du nom de la classe:

```
Class objetClass = Class.forName(" MaClasse ");
```

Une fois un objet de type Class obtenu, il est possible de faire de l'introspection (*reflection*) sur les classes, c'est-à-dire trouver dynamiquement les membres ou constructeurs de la classe et y accéder.

Aperçu du chapitre

IV . La classe Math

La classe **Math** contient toute une série de fonctions statiques pour effectuer, entres autres, les calculs suivants :

- Valeur absolue
- Sinus, cosinus, tangente, arc sinus, arc cosinus, arc tangente
- Exponentielle
- Logarithme
- Arrondissement
- Exposant
- Conversion radians > degrés (et inversement)
- …

Elle contient également des constantes statiques tel que le nombre PI.

Aperçu du chapitre

V . La classe String et les classes « primitives »

La classe **String** représentent les chaînes de caractères en Java.

Les **conversions de tout type primitif vers String** se fait au moyen de la méthode statique surchargée **String.valueOf**.

```
int numero = 1;
String num = String.valueOf(numero);
```

Dans le cas des **objets**, c'est leur méthode **toString()** qui est appelée.

Un ensemble de méthodes prédéfinies permet les manipulations classiques sur les chaînes de caractères:

- charAt(int) : retourne le caractère numéro i
- substring(int, int) : retourne une sous-chaîne de caractères
- replaceAll(String, String) : remplacement de caractères
- toLowerCase() : rendre minuscule tous les caractères
- toUpperCase() : rendre majuscule tous les caractères
- trim() : retire les espaces situés au début et à la fin du String
- …

V . La classe String et les classes « primitives »

Chaque type primitif possède une classe qui lui correspond.

On parle de classe « wrapper ».

Ces classes permettent, entre autres, la conversion d'un String en un type primitif.

```
int i = Integer.parseInt("123");
```


Aperçu du chapitre

VI . La classe StringBuilder

La classe **StringBuilder** représentent également les chaînes de caractères en Java.

Elle permet d'**éviter les allocations et les désallocations de chaînes temporaires** en permettant d'étendre ou de modifier une chaîne de caractères en place.

```
String prenom = "Jean";

StringBuilder buffer = new StringBuilder ("Bonjour , ");
buffer.append(prenom);
buffer.append(". Ca va ?");
String message = buffer.toString();

StringBuilder buffer 2= new StringBuilder ("Bonjour , ");
buffer2.insert(4,prenom);
String message2 = buffer2.toString();
```


Aperçu du chapitre

VII . La classe Calendar

La classe **Calendar** est utilisée pour récupérer des infos sur la date actuelle, l'heure en fonction de la zone horaire, …

Exemple:

```
// Afficher la date actuelle
System.out.println(Calendar.getInstance().getTime());
```

L'affichage de cet exemple sera semblable à celui-ci :

```
Wed Jul 16 14:05:31 CEST 2014
```

Il est cependant possible de le formater pour avoir un autre format (16/07/2014 par exemple)