Java

CM3-4 : Java, outils

Mickaël Martin Nevot

V1.1.1

Cette œuvre de Mickaël Martin Nevot est mise à disposition sous licence Creative Commons Attribution - Utilisation non commerciale - Partage dans les mêmes conditions.

 BY-NC-SA

Java

I. Prés.
II. POO
III. Objet
IV. Java
V. Types
VI. Héritage
VII. Outils
VIII. Exceptions
IX. Polymorphisme
X. Thread
XI. Avancé

 (cc) BY-NC-SA

Classe Object / méthode main

- `Object` :
  - **Classe de plus haut niveau** dans la hiérarchie d'héritage
  - Toute classe autre que `Object` possède une super-classe
  - Toute classe hérite directement ou pas de `Object`
  - Toute classe qui n'a pas de clause `extends` hérite de `Object`
- `main( … )` :
  - Point de commencement d'exécution du code
  - Au moins une par application (**classe principale**) :

```
public static void main(String[] args) {
    // Le code va commencer par s'exécuter ici.
}
```

 BY-NC-SA

A savoir

- Classe `String` :

```
String myString = "Hello!";
myString += "How are you?";
```

- Classe `File` :

```
File myFile = new File("file.txt");
```

- Affichage (y compris types primitifs/références) :

```
System.out.println("a = " + a);
```

 (cc) BY-NC-SA

Javadoc

- Outil standard pour créer une **documentation** d'API
- Génération automatique en HTML
- Utilisation (≠ commentaire `/* */`) :
  - Première ligne : uniquement `/**`
  - Lignes suivantes : un espace suivi de `*`
  - Dernière ligne : un espace suivi uniquement de `*/`
  - L'entité documentée est précédée par son commentaire
  - Tags prédéfinis

Bonne utilisation : expliquer n'est pas traduire !

 (cc) BY-NC-SA

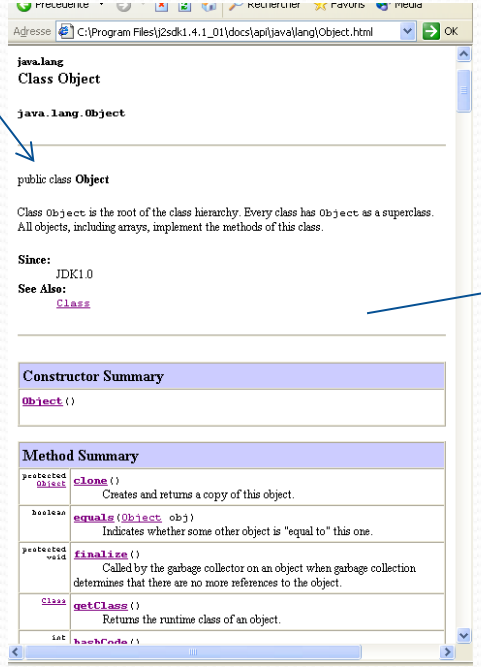
Javadoc : principaux tags

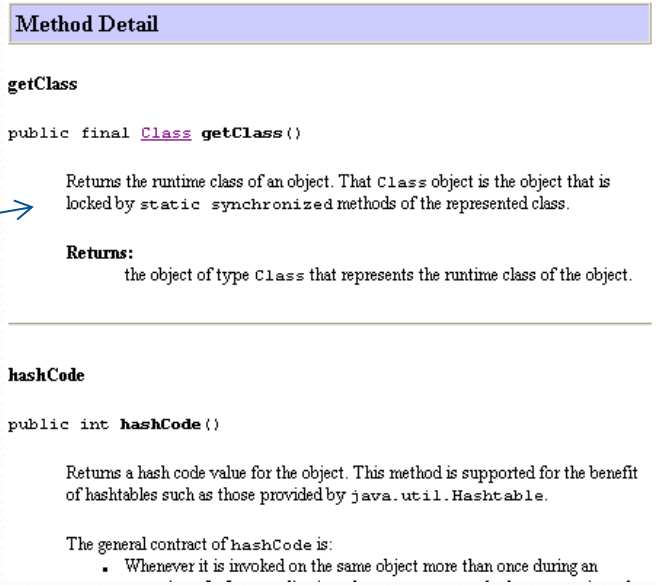
- `@author` : nom du développeur
- `@version` : version d'une classe/méthode
- `@param` : définit un paramètre de méthode :
  requis pour chaque paramètre
- `@since` : version du JDK de l'apparition de la classe/méthode
- `@return` : valeur de retour
- `@throws` : classe de l'exception et conditions de lancement
- `@deprecated` : marque la méthode comme dépréciée
- `@see` : référence croisée avec un autre élément

BY-NC-SA

Exemple de Javadoc

```
/**
 * Valide un mouvement de jeu d'Échecs.
 * @param beginCol Colonne de la case de départ
 * @param beginRow Ligne de la case de départ
 * @param endCol   Colonne de la case de destination
 * @param endRow   Ligne de la case de destination
 * @return vrai(true) si le mouvement d'échec est valide ou faux(false) sinon
 */
```





(cc) BY-NC-SA

API

Paquetages

Classes

Description
Attributs
Méthodes

Java™ 2 Platform
Std. Ed. v1.4.2

All Classes

Packages
java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.geom
java.awt.im
java.awt.im.spi
java.awt.image

All Classes
ARG_IN
ARG_INOUT
ARG_OUT
AWTError
AWTEvent
AWTEventListener
AWTEventListenerProxy
AWTEventMulticaster
AWTException
AWTKeyStroke
AbstractAction
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeContext
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDimensions
AbstractList
AbstractListModel
AbstractMap
AbstractMethodError
AbstractPreferences
AbstractSelectableChannel
AbstractSelectionKey
AbstractSelector
AbstractSequentialList
AbstractSet
AbstractSpinnerModel
AbstractTableModel
AbstractUndoableEdit
AbstractWriter
AccessControlContext
AccessControlException
AccessController
AccessException
Accessible
AccessibleAction
AccessibleBundle
AccessibleComponent

Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Java™ 2 Platform
Std. Ed. v1.4.2

Java™ 2 Platform, Standard Edition, v 1.4.2
API Specification

This document is the API specification for the Java 2 Platform, Standard Edition, version 1.4.2.

See:
Description

Java 2 Platform Packages	
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing *beans* -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (BigInteger) and arbitrary-precision decimal arithmetic (BigDecim
java.net	Provides the classes for implementing networking applications.
java.nio	Defines buffers, which are containers for data, and provides an overview of the other NIO packages.
java.nio.channels	Defines channels, which represent connections to entities that are capable of performing I/O operations, such as files and sockets; defines selectors, for multiplexed, non-blocking I/O operations.
java.nio.channels.spi	Service-provider classes for the `java.nio.channels` package.
java.nio.charset	Defines charsets, decoders, and encoders, for translating between bytes and Unicode characters.
java.nio.charset.spi	Service-provider classes for the `java.nio.charset` package.
java.rmi	Provides the RMI package.
java.rmi.activation	Provides support for RMI Object Activation.
java.rmi.dgc	Provides classes and interface for RMI distributed garbage-collection (DGC).
java.rmi.registry	Provides a class and two interfaces for the RMI registry.
java.rmi.server	Provides classes and interfaces for supporting the server side of RMI.

 (cc) BY-NC-SA

Outils

- Éditeur :
  - Eclipse : http://www.eclipse.org
- Ressources Java :
  - API : http://download.oracle.com/javase/1.5.0/docs/api
  - Convention de nommage : http://java.sun.com/docs/codeconv/CodeConventions.pdf
  - Mots clefs réservés : http://download.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

(cc) BY-NC-SA

Bonnes pratiques

- Penser à l'initialisation pour éviter une erreur
- Penser à construire les objets avant de les utiliser
- Penser à l'utilisation de `break` dans un `switch`
- Attention à l'encapsulation
- Utiliser le mot clef `this` autant de fois que possible
- Pas de mot clef `then` (en relation avec un `if`)
- Pas d'utilisation de variable d'instance ni du mot clef `this` dans une méthode de classe
- Pas d'héritage multiple

BY-NC-SA

Crédits

Auteur

Mickaël Martin Nevot
mmartin.nevot@gmail.com

Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com