

# Java Refcard 1.1 (Auteur C.Dutoit)

## 1 histoire

Mis au point par Sun en 1991. Conçu pour être petit (raté). 1994 : Invention des applets.  
1995 : Java intégré dans Netscape → début de la success story (hem).

## 2 Java vs C++

Java=C++ moins : caractéristiques critiques (pointeurs, surcharge d'opérateurs, héritage multiple, plus de destructeurs=libération de la mémoire transparente, meilleure gestion des erreurs, Chaînes et tableaux sont des objets faisant partie du langage). Java est plus lent que C++, mais plus portable. Bytecode. Tout est objet.

## 3 Versions

- 1.0 : première version stable. La totalité des browsers le supporte.
- 1.1 : (1997) Améliorations syntaxiques, meilleur interface utilisateur, gestion des exceptions.
- 2.0 : améliorations multimédias. Utilisation d'interface graphique avancée avec Swing.

## 4 Programmes

**javac** : Compilateur java; 'javac moncode.java'; option -g inclus les infos de débogage

**java** : Interpréteur java (machine virtuelle); 'java moncode'

**applet viewer** : interpréteur d'applets; 'appletviewer moncode.class'

**jdb** : débogueur

**javap** : Décompilateur Bytecode-code source

**javadoc** : générateur de documentation html

**jar** : compresseur de classes java

Note : case-sensitive entre le nom de la classe et le nom du fichier. Le même nom est impératif !

Le JDK est téléchargeable sur java.sun.com ou javasoft.com taille : 20MO + 12MO de doc'

## 5 Syntaxe

**Commentaires** : /\* ... \*/ (non imbriquables) et //

**Commentaires pour JavaDoc** : /\*\* ... \*/

## 6 Canevas d'applications

### 6.1 Applications

```
public class Nom_du_prg {
    public static void main (String args[]) {
        System.out.println("Hello World");
    }
}
```

### 6.2 Applets

```
public class Nom_applet extends java.applet.Applet {
    public void paint (Graphics g) {
        g.drawString("Hello World", 20, 40);
    }
}
```

## 7 Applet Java

Code Java :

```
public class MyApplet extends java.applet.Applet {
    public void init(){
        add(new Label("Hello World"));
    }
}
```

Code HTML :

```
<html><body>
    <APPLET code="MyApplet.class" width=100 height=100>
    Msg si l'applet non supportée par le navigateur
</APPLET>
</body></html>
```

## 8 Types

Conventions :

**Variable** : commence par une minuscule (?)

**Constante** : Tout en majuscule. Ex : final double PI=3.1415926

**Instance d'objet** : commence par une minuscule

### 8.1 Primitives

Primitive	Signification	Bytes	Plage de valeurs valides
char	Caractère	2	...
byte	entier très court	1	-128 à 127
short	entier court	2	-32'768 à 32'767
int	entier	4	-2'147'483'648 à 2'147'483'647
long	entier long	8	-9'223.10 <sup>15</sup> à 9'223.10 <sup>15</sup>
float	flottant(réel)	4	-1, 4.10 <sup>-45</sup> à 3, 4.10 <sup>38</sup>
double	flottant double	8	4, 9.10 <sup>-324</sup> à 1, 7.10 <sup>308</sup>
boolean	booléen	1	0 ou 1 (si >1, considéré comme 1)

Note : Tableaux = int tab[] = new ...[...]; ...tab.length()...

### 8.2 Wrappers (enveloppeurs)

Enveloppeur	Primitive associée
BigDecimal	-
BigInteger	-
Character	char
Byte	byte
Short	short
Integer	int
Long	long
Float	float
Double	double
Boolean	boolean
Void	void

\*\*\*String s="\*\*\*;

### 8.3 Variables

Déclaration n'importe où dans le code : type Ma\_Var1=donnée, Ma\_Var2, ...;

Visibilité :

- globale si déclaration à l'extérieur de toute fonction, classe...

- locale=limitée au bloc dans laquelle la var. est déclarée

Constantes : final int Variable=12; cast : x=(int)3.1415;

Conversion de types

**Integer** → **int** : (int)myInt

**Integer** → **String** : myInt.toString() ou myInt+""

**String** → **Integer** : myInt.valueOf("18");

Affectation :

**Partage** : P=X

**Copie** : C=X.clone(); → public Object clone()... (!Copie superficielle-profonde)

## 9 Opérateurs

Arithmétiques : + - \* / = += -= \*= /= ++ --

Comparatifs : == < > >= <= !=

Booléens : || && !

Bit-à-bit : & | ~^(et, ou inclusif, ou exclusif)

Rotation de bits : << >> >>> (bit shift left, right, right+0)

## 10 Priorités

0	[]	.		-(un opérande)	(casting)	new
-	++	!				
*	/	%				
+	-					
<<	>>	>>>				
<	<=	>=	>	instanceof		
==	!=					
&						
^						
-						
&&						
~						
?	:					
=	+=	-=	*=	/=	%=	&=
<<=	>>=	>>>=	△	—=		
.						

## 11 Structures conditionnelles

```
if (condition) { ... } else { ... }
(condition)?instruction_si_vrai:instruction_si_faux;
```

```
switch(Variable) {
    case Valeur1: ... break;
    case Valeur2: ... break;
    default: ... break;
}
for (int i=0; i<10; i++) { ... } (Instruction break dans for sort du bloc for...)
while (condition) { ... } (Instruction continue dans while passe revient au début du bloc)
```

## 12 Classes

```
class Nom_classe{...}
Instanciation de classe: MaClasse toto=new MaClasse();
```

Données membres :

```
public char maChaine[20]; private float monFloat; protected int monInt;
```

Utilisation : NomClasse.NomDonneeMembre=...;

this : objet dans lequel on se trouve.

Méthodes : int getX(){ ... }

Fonctions : float calcF() { ... }

Protection : public int getX; protected... private... Surcharge : d'opérateurs uniquement. (pas sur le type de retour)

Constructeurs :

- Même nom que la classe
- Pas de type de retour
- Peut avoir des arguments
- Pas obligatoire. (Constructeur par défaut fournit par Java)
- Plusieurs constructeurs possible.

Exemple :

```
class Chat51{ // ;-)
    int age;
    float poids;
    Chat(int age, float poids) {
        this.age=age;
        this.poids=poids;
    }
}
```

### 12.1 Héritage

```
class truc extends chose { ... } Accès à la classe parent(superclasse) : super.propriété=3; super.setX(3);
Héritage simple uniquement!
```

## 13 interfaces

Code : interface myInterface { ... public void x(); ... }

Décrit : seulement des opérations

Utilisation : class ... implements monInterface{ ... } (doit implémenter tout ce qui est dans l'interface)

## 14 Multitâche

```
class ... implements runnable { Thread activite = new Thread(Param); public void run() try thread.sleep(1000);
catch(interruptException e) ..... }
```

## 15 paquetages

Le dossier du paquetage doit avoir le même nom que le paquetage.

Classpath :

- Java recherche le paquetage dans le dossier courant et dans 'classpath'.
- Linux : export CLASSPATH=/home/bgates/java:/usr/lib/jdk/lib:...
- Windows : SET CLASSPATH=c:\java;d:\donnees\ltorvalds\java;...

Utilisation :

- MonPackage.MaClasse
- import Monpackage.MaClasse
- import Monpackage.\*

Convention de nom : Donner un nom au package et l'allonger par le nom de la société, ou du concepteur des classes qu'il contient. Exemple : graph.eivd.spider.

Note : Toutes les classes du dossier courant sont visibles !

Fichiers jar :

But : S'assurer que les app sont intacts, que tous les packages de l'app soient présents dans le cadre d'un transfert sur le net. Moins gourmand en espace.

Création : jar cvf MonArchive.jar \*.class

Décompression : jar xvf MonArchive.jar \*.class

Utilisation : Web : appel du jar uniquement

Soft : Mojar permet la manipulation des jar en interface graphique. (www.opengroup.com/moajar)

## 16 Exception

```
try{ ... }
catch (TypeException e) { ... }
Note : pas d'instructions entre try et catch...
Propagation : throw(TypeException) e;
```

## 17 JDBC

Java DataBase Connectivity : Ensemble de classes permettant de développer des apps capables de se connecter à des serveurs de bases de données (SGBD).  
...(more to come)

## 18 Divers

Mémoire : gérée par le garbage collector.

Portabilité : grâce au ByteCode

## 19 web

ma Source : <http://www.commentcamarche.net/java>