#### **Boucles**

while Crée une boucle avec une condition d'arrêt.

**for** Crée une boucle avec une initialisation de variable, une condition d'arrêt et une règle d'incrémentation.

do Crée une boucle, associée à un « while », dont la condition d'arrêt est évaluée après l'itération.

**continue** Stoppe l'exécution d'une boucle et reprend à l'itération suivante (ou au niveau d'un label).

## **Exceptions**

throw Lève une exception.

throws Indique qu'une méthode peut lever une ou plusieurs exceptions.

try Ouvre un bloc pour intercepter des exceptions.

**catch** Ouvre un bloc pour gérer une exception survenue dans le « try ».

finally Ouvre un bloc toujours exécuté après le « try ».

```
FileReader reader = null;

try {
          reader = new FileReader(myFile);
          ...
} catch (Exception e) {
          e.printStackTrace();
} finally {
          reader.close();
}
```

#### **Branchements**

**if** Exécute un bloc lorsqu'une condition booléenne est satisfaite.

**else** Exécute un bloc lorsque la condition du « if » n'est pas satisfaite.

**switch** Exécute un bloc de code, désigné par un label « case », en fonction d'une valeur.

case Définit un label cible dans un bloc « switch ».

default Définit un label par défaut quand aucun label d'un bloc « switch » ne correspond.

**break** Saute hors d'une boucle, hors d'une instruction (« switch ») ou vers un label.

assert Vérifie qu'une condition est remplie, sinon lève une erreur avec le message spécifié.

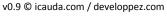
**instanceof** Teste si une variable est une instance d'un type spécifique.

**return** Stoppe l'exécution d'une méthode et renvoie la valeur (facultative) spécifiée.

**synchronized** Indique qu'un seul thread à la fois peut accéder à une méthode ou à un bloc.







Nous offrons ce mémento pour le diffuser au maximum. N'hésitez pas à nous indiquer vos retours afin que nous puissions l'améliorer.

Scannez le QR code ci-contre pour lire l'article détaillé et télécharger gratuitement ce mémento ou rendez-vous sur http://icauda.com/articles



abstract	else	interface	switch
assert	enum	long	synchronized
boolean	extends	native	this
break	false**	new	throw
byte	final	nu11**	throws
case	finally	package	transient
catch	float	private	true**
char	for	protected	try
class	goto*	public	void
const*	if	return	volatile
continue	implements	short	while
default	import	static	
do	instanceof	strictfp	
double	int	super	

(\*) réservé mais pas utilisé

(\*\*) réservé mais pas clé



## **Objets**

class Définit un objet de type « classe ».

**interface** Définit un objet de type « interface », qui spécifie un comportement mais pas son code.

enum Définit un objet de type « énumération ».

**extends** Indique qu'une classe ou une interface étend respectivement une autre classe ou d'autres interfaces.

implements Indique qu'une classe implémente une ou plusieurs interfaces.

import Référence des classes, des packages entiers ou des méthodes pour les utiliser dans le programme sans écrire leurs noms complets.

this Référence l'objet courant.

super Référence la classe mère de l'objet courant.

package Indique le package auquel appartient l'objet.

**abstract** Indique qu'une classe ou une méthode est abstraite (devra être implémentée par héritage) .

**native** Indique qu'une méthode est écrite de manière native, dans un autre langage que Java et dans un autre fichier.

### **Modificateurs**

**private** Indique qu'un élément n'est accessible que dans la classe où il est définit.

protected Indique qu'un élément n'est accessible que dans la classe où il est définit, ou dans les classes filles, ou encore dans les classes du même package.

public Indique qu'un élément est accessible partout.

# États

false\*\* Représente une valeur booléenne négative.

true\*\* Représente une valeur booléenne positive.

new Opérateur permettant d'instancier un objet.

null\*\* Valeur indiquant qu'une variable ne référence rien.

transient Interdit qu'une variable soit sérialisée.

**static** Indique qu'une variable, une méthode ou un bloc n'appartient pas à une instance particulière d'une classe.

**strictfp** Garantit la précision et les arrondis des calculs flottants pour assurer la portabilité, conformément à la spécification « IEEE 754 ».

final Interdit qu'un élément soit modifié ou étendu.

**volatile** Garantit la synchronisation d'une variable dans un contexte multithread.

### **Type**

void Indique qu'une méthode ne renvoie pas de valeur.

boolean	Booléen	true (vrai) et false (faux)	
char	Caractère	16 bits	
byte	Entier	8 bits signés	
short	Entier	16 bits signés	
int	Entier	32 bits signés	
long	Entier	64 bits signés	
float	Décimal	32 bits signés	
double	Décimal	64 bits signés	