concept	définition	pseudo-code	Java
affectation	attribution d'une valeur (par exemple 5) à une variable (par exemple l'entier a)	$a \leftarrow 5$	`a=5;`
affichage	présentation à l'écran d'une valeur entière, d'un flottant, d'une chaîne de caractères ou d'un booléen	**Affiche**("toto")	`System.out.println("toto");`

algorithme	suite d'instructions pour résoudre un problème, c'est-à-dire fournir un résultat en sortie à partir de variables en entrée	Algorithme **Algo1**
		Entrées : *entree*1 de type type1, *entree*2 de type type2.
		Sortie : variable de type type3 (dépend des valeurs de *entree*1 et *entree*2).
		Variables : *var*1 de type type3.
		Début
		*var*1 ← **Algo2**(*entree*1,*entree*2)+*entree*2
		renvoyer *var*1
		Fin
			`public static type3 Algo1(type1 entree1, type2, entree2){`
			`type3 var1;`
			`var1=Algo2(entree1,entree2)+entree2;`
			`return var1;`
			`}`

booléen	valeur VRAI ou FAUX	**Variable :** booléen *resultat*
		resultat ← VRAI
		resultat ← FAUX
			`boolean resultat;`
			`resultat=true;`
			`resultat=false;`

boucle	structure de programmation qui permet d'exécuter plusieurs fois un suite d'instructions	**Tant que** ... **faire** :
		...
		Fin TantQue
			`while(...){`
			`...;`
			`}`

chaîne de caractères	ensemble de caractères consécutifs, toujours notée entourée de guillemets droits	**Variable :** chaîne de caractères *chaine* *chaine* ← "toto"	`String chaine;` `chaine="toto";`
commentaire	ligne d'un code source qui n'est pas exécutée	//Commentaire	`//Commentaire`
compilation	création du code exécutable d'un programme à partir de son code source. On compile du code source Java avec le programme `javac`.	-	-
concaténation	opération qui réunit en une seule chaîne de caractères deux chaînes de caractères	**Concatène**("toto","1") renvoie "toto1"	`"toto"+"1"` est la chaîne de caractères `"toto1"`
déclaration	définition du type d'une variable	**Variables :** entier *var*1 et *var*2 de type type2.	`int var1; type2 var2;`
égalité	être de même valeur	1=1 / "chaine"="chaine"	`1==1 / "chaine".compareTo("chaine")==0`
entier	en pseudo-code, "nombre entier" ; en Java, "entier compris entre -2 147 483 648 et 2 147 483 647"	**Variable :** entier n $n \leftarrow 42$	`int n;` `n=42;`
flottant	nombre à virgule	**Variable :** flottant *pi* *pi* ← 3.14156	`double pi;` `pi=3.14156;`
initialisation	première affectation dans une variable	(voir *affectation*)	(voir *affectation*)
instruction	ordre donné à un ordinateur, élément d'un algorithme	une instruction par ligne	instruction toujours terminée par ";"
longueur	pour un tableau t, nombre de cases ; pour une chaîne de caractères c, nombre de caractères	**Longueur**(t), **Longueur**(c)	`t.length; c.length();`

modulo	(a modulo b) est le reste dans la division de a par b. On peut utiliser modulo pour tester si un entier n est pair.	**Modulo**(a,b)
		//Test de parité de n :
		Si(**Modulo**(n,2)=0)...
			`a%b`
			`//Test de parité de n :`
			`if(n%2==0)...`

renvoyer	donner le résultat d'un algorithme	**renvoyer** 5	`return 5;`

tableau	ensemble de variables de même type. Le nombre de valeurs stockées (la *longueur*) ne peut pas être changée après l'initialisation du tableau. La numérotation des cases en Java commence à 0.	**Variable :** tableau d'entiers t
		$t \leftarrow$ **NouveauTableau(2)**
		Case(t,1)←3
		Case(t,2)←6
			`int[] t;`
			`t=new int[2];`
			`t[0]=3;`
			`t[1]=6;`
			équivalent à
			`int[] t;`
			`t=new int[]{3,6};`

test	structure de programmation qui permet d'exécuter une instruction de manière conditionnelle	**Si** ... **alors** :
		...
		Sinon :
		...
		FinSi
			`if(...){`
			`...;`
			`} else {`
			`...;`
			`}`

trace	suite des valeurs prises par chaque variable tout au long de l'algorithme	-	-
type	ensemble de valeurs possibles pour une variable	entier, flottant (nombre à virgule), chaîne de caractères, booléen, couleur	`int`, `double`, `String`, `boolean`, `Color`
valeur	contenu d'une variable, par exemple entier 2, chaîne de caractères "2", ou tableau à deux cases numérotées 1 et 2	2 "2" 1 | 2	2 "2" {1,2}
variable	élément ayant un nom fixé et un type fixé, qui permet de stocker une valeur, qui peut changer suite à une affectation	(voir *déclaration*, *initialisation*, *affectation*, *type*, *valeur*)	(voir *déclaration*, *initialisation*, *affectation*, *type*, *valeur*)