Lexique des types de base

Types entiers	byte, short, int, long
Types réels	float, double
Type caractère	char
Type booléen	boolean

Syntaxe de déclaration de variable

type nomDeVariable ;	int age ; double rayon ;

Affectation (opérateur =)

nomDeVariable = expression ;	prix = 15 ; surface = PI * rayon * rayon ;

Déclaration et affectation – raccourcis d'écriture

type nomDeVariable = expression ;	double prix = 16.5 ; double surface = PI * rayon * rayon ;

Opérateurs arithmétiques

Addition	+
soustraction	-
division (entière et réel)	/
multiplication	*
Modulo (reste de la division entière)	%

Opérateurs booléens et logiques

Plus petit	<
Plus petit ou égal	<=
Plus grand	>
Plus grand ou égal	>=
égalité	==
différent de	!=
Et	&&
Ou	||
N'est pas	!

Changement de type (type cast)

(type) expression ;	int arrondi = ((int)(x*10+0.5))/10d ;
Raccourcis pour les constantes littérales	double res = 10 / 7**d** ; //division dans les réels

Déclaration de constantes

static final type NOM_CONSTANTE = valeur;	static final int AGE_MAJORITE = 18;

Commentaires

Commentaire sur une ligne	// ceci est un commentaire
Commentaire sur plusieurs lignes	/* Commentaire sur plusieurs lignes. */
Java doc (documentation avec génération automatique d'aide pour le programmeur)	/** * */

Affichage sur la sortie standard (et opérateur de concaténation +)

Affichage avec retour à la ligne	System.out.println("Hello World ! ");
Affichage sans retour à la ligne	System.out.print("Hello World ! ");
Affichage avec retour à la ligne et concaténation	System.out.println("Masse : " + masse + " [kg] ");

Lecture au clavier

Initialisation du scanner	Scanner clavier = new Scanner(System.in);
Affichage d'un message de saisie	System.out.println("Entrer un nombre: ");
lecture de la donnée	double nb = clavier.nextDouble() ;
Particularité :	
lecture d'un caractère	char c = clavier.next().charAt(0) ;

Déclaration de fonction (rajouter public devant static si la fonction est dans une bibliothèque)

static type nomDeFonction(type parametre, …){
bloc d'instructions
return expression
}
	static boolean estMinuscule(char c) {
	return c >= 'a' && c <= 'z';
	}

static void nomDeFonction(type para, …){
bloc d'instructin
}
	static void afficherResultats(double n1, int n2) {
	System.out.println("n1 vaut " + n1);
	System.out.println("n2 vaut " + n2);
	}

Les formants algorithmiques de sélection

Si condition alors bloc 1 sinon bloc 2	if (nb % 2 == 0) {
	…
	} else {
	…
	}

Si condition alors bloc 1	if (nb >= 0 && nb <= 10) {
	…
	}

si condition alors bloc 1 sinon si condition alors bloc2 sinon bloc3	if (nb > 6) {
	… //ici nb est plus grand que 6
	} else if (nb > 4) {
	… //ici nb est entre 4 non compris et 6 compris
	} else {
	… //ici nb est strictement plus petit que 4
	}

Les formants itératifs

Initialisation de la condition
Tant que condition faire
bloc d'instruction
mise à jour de la condition
Fin tant que
	Int i = 0 ;
	while (i < 10) {
	//bloc d'instruction
	i = i + 1;
	}

Identique mais avec le raccourcis **for**	for (int i = 0; i < 10; i++) {
	//bloc d'instruction
	}

Raccourcis d'écritures : affectation et opérateur arithmétique

var = var + 1 ;	Post incrémentation: var++; Pré incrémentation: ++var;
var = var - 1 ;	Post décrémentation: var--; Pré décrémentation: --var;
var = var + nb ;	var += nb;
var = var * nb ;	var *= nb;
var = var / nb ;	var /= nb;
var = var % nb ;	var %= nb;
Différence entre post et pré incrémentation	
int var = 7 ; int nb = var ; var = var + 1 ;	int var = 7; int nb = var++; //var => 8, nb => 7
int var = 7 ; var = var + 1 ; int nb = var ; Idem pour post et pré décrémentation	int var = 7; int nb = ++var; //var => 8, nb => 8

Quelques algorithmes itératifs classiques

Boucle de validation d'une donnée lue

```
Lire la donnée
Tant que donnée invalide
   afficher un message d'erreur
   relire la donnée
Fin tant que
```

```
int n = Read.readInt("Entrez un nombre positif");
while (n<0) {
   System.out.println("le nb n'est pas positif ! ")
   n = Read.readInt("Entrez un nombre positif");
}
```

Boucle de répétition

```
Pour itération allant de 0 à 9 par pas de 1
   Bloc d'instructions
Fin pour
```

```
for (int i = 0; i < 10; i++) {
  //bloc d'instructions
}
```

Boucle de traitement de flot

```
Pour chaque membre du flot
   lire le membre du flot
   bloc d'instructions
Fin pour
```

Exemple avec le calcul de la somme

```
int somme=0; int nb;
for (int nbTraite=0; nbTraite<10; nbTraite++) {
   nb = Read.readInt("Entrez un nombre: ");
   somme += nb
}
```

Boucle de traitement de flot avec exception

```
Lecture et gestion du(des) premier(s) membre(s)
du flot comme cas particulier
Pour chaque membre du flot restant
   lire le membre du flot
   bloc d'instructions
Fin pour
```

Exemple avec la recherche du maximum

```
int nb = Read.readInt("Entrez un nombre: ");
int max = nb;
for (int nbTraite=1; nbTraite<10; nbTraite++)  {
   nb = Read.readInt("Entrez un nombre: ");
   if (nb > max)  {
      max = nb;
   }
}
```

Nombres (pseudo)aléatoires

Nombre **double** dans l'intervalle [0 ;1[

```
double nb = Math.random();
```

Nombre **int** dans l'intervalle [min ;max]

```
public static int random(int min, int max) {
   double n = Math.random()*(max-min+1)+min;
   return (int)n;
}
```

Exemple d'utilisation de la function:

```
int nb = random(10, 100);
```

Les tableaux et quelques algorithmes de traitement classique

Déclaration : type[] nomVar ;

```
int[] ages;
```

Initialisation : nomVar = new **type**[taille]

```
ages = new int[10];
```

Affectation d'une valeur à un élément :
nomVar [indice] = valeur
remarque : indice est un int dans [0 ;taille-1]

```
ages[0] = 30;
```

Algorithme de parcours classique (parcours de tous les éléments du tableau) :

```
pour chaque indice entre [0 ;taille-1] faire
   bloc d'instructions
fin pour
```

Exemple avec l' initialisation aléatoire d'un tableau d'entier:

```
for (int i = 0; i < ages.length; i++){
  age[i ]=MyMath.random(AGE_MIN,AGE_MAX);
}
```

Algorithme de recherche de la position(indice) d'un élément d'un tableau (version de base)

```
chercher l'élément depuis la position 0
tant que la position est valide ET l'élément n'est
          pas à cette position
   aller à la position suivante
fin tant que
si la position est valide alors
   retourner cette position
sinon
   retourner « non trouvé » sous la forme -1
fin si
```

Exemple avec un tableau de char:

```
public static int search(char[] tab, char ele){
   int pos = 0;
   while(pos < tab.length && tab[pos] != ele){
      pos++;
   }
   if (pos < tab.length){
      return pos;
   } else {
      return -1;
   }
}
```

Algorithme de parcours Gauche / Droite

```
placer la borne gauche sur la 1er case
placer la borne droite sur la dernière case
Tant que les bornes ne se sont pas croisées
   (bloc dinstructions)
   incrémenter gauche
   décrémenter droite
Fin tant que
```

Exemple avec l'inversion d'un tableau

```
   int gauche  = 0; int droite  = array.length-1;
   while (gauche<droite) {
         int temp = array[gauche];
         array[gauche] = array[droite];
         array[droite] = temp;
         gauche++;
         droite--;
   }
```

Algorithme de tri par recherche du minimum

```
   int tamp; int j; int petit;
   for (int i = 0; i < array.length - 1; i++) {
      petit = i;
      for (j = i + 1; j < array.length;  j++) {
         if (array[j] < array[petit]) {
            petit = j;
         }
      }
      tamp = array[i];
      array[i] = array[petit];
      array[petit] = tamp;
}
```