Structures de base [bs]
Résumé de cours

Université de Haute Alsace

Unisciel algoprog Version 12 mai 2018

Table des matières

1 Java - Résumé de cours

1.1 Le langage

Identifiant
Séquence de lettres (A...Z, a...z), de chiffres (0...9), de lettres accentuées ou du caractère souligné (_) ou `$`, ?, μ. Le premier caractère doit être différent d'un chiffre.

La casse
Le langage est *case-sensitif* et les accentués sont autorisés.
Ceci signifie que `cout`, `Cout` et `COUT` réfèrent trois mots différents et `coût` est licite.

1.2 Variables, types et valeurs

Variable
Élément informatique qu'un programme peut manipuler.
Décrite par :

- Un **identifiant** unique qui la désigne.

- Un **type** qui définit de quel « genre » est l'information associée.
- Une **valeur** qui doit respecter le type.

Types intégrés

Domaine	Algorithmique	Équivalent Java
$\mathbb{Z}$	`Entier`	`int`
$\mathbb{R}$	`Réel`	`double`
$\mathbb{B}$	`Booléen`	`boolean`
$\mathbb{A}$	`Caractère`	`char`
$\mathbb{T}$	`Chaîne`	`String`

Littéraux

- `Entier` : Suite de chiffres éventuellement préfixé par un signe (+ ou −).
- `Réel` : S'écrit en notation décimale ou en notation scientifique.
- `Booléen` : Ils identifient le `Vrai` (mot-clé `true`) et le `Faux` (mot-clé `false`).
- `Caractère` : Se place entre quotes (').
- `Chaîne` : Se place entre guillemets (").

1.3 Déclarations

Déclaration de variables

```
TypeVar nomVar;
TypeVar nomVar1, nomVar2, ...;
```

Initialisation d'une variable

```
TypeVar nomVar = valeur;
TypeVar nomVar(valeur);
```

Définition de constante

```
final static TypeConst nomConst = expression; // notation impérative
final static TypeConst nomConst(expression); // notation objet
```

1.4 Structure générale

Commentaire orienté ligne

```
... // rend le reste de la ligne non-exécutable (hérité du C++)
```

Commentaire orienté bloc

```
/*
rend le code entouré non exécutable...
(hérité du C)
*/
```

Bloc

```
{
  instruction1;
  instruction2;
  ...
}
```

Structure générale

```
import des_trucs_utiles;
public class nomAlgo
{
  déclaration_des_objets_globaux
  déclarations_et_définitions_de_fonctions_utiles
  public static void main(String[] args)
  {
    corps_du_programme
  }
}
```

1.5 Interactions avec l'extérieur

Pour utiliser la saisie

```
import java.util.Scanner;
```

Saisie de données

```
Scanner input = new Scanner(System.in);
nomVarI = input.next();       // lecture d'une chaîne
nomVarI = input.nextLine();   // lecture d'une ligne
nomVarI = input.nextInt();    // lecture d'un entier
nomVarI = input.nextDouble(); // lecture d'un réel
nomVarI = input.nextChar();   // lecture d'un caractère <=> input.next().charAt(0);
```

Affichage de résultats

```
System.out.print(expr1+expr2+...+exprN);   // SANS retour de ligne
System.out.println(expr1+expr2+...+exprN); // AVEC retour de ligne
```

1.6 Expressions algébriques

Expression, opérandes, opérateurs

Éventuellement accompagnés de parenthèses, une **expression** est une séquence « bien formée » (au sens de la syntaxe) d'**opérandes** (valeurs littérales, variables ou expressions) et d'**opérateurs** destinée à l'évaluation.

Opérateurs arithmétiques

Opérateur Mathématique	Signification	Équivalent Java
+	(unaire) valeur	`+a`
-	(unaire) opposé	`-a`
+	addition	`a + b`
-	soustraction	`a - b`
*	multiplication	`a * b`
/	division décimale	`a / b`
div	division entière	`a / b`
mod	modulo (reste de la division entière)	`a % b`

Ordre de priorité des opérateurs arithmétiques

Comme en mathématique :

1. Les opérateurs unaires (+, −) (priorité la plus élevée)
2. L'opérateur d'exponentiation () (s'il existe)
3. Les opérateurs multiplicatifs (∗, /, div, mod)
4. Les opérateurs additifs (+, −) (priorité la plus basse)

La règle d'associativité s'applique en cas d'ambiguïté entre opérateurs du même ordre de priorité.

Règle de promotion

Pour qu'une opération numérique binaire $(+, -, *, /)$ puisse s'effectuer, il faut que ses deux opérandes soient du **même type** ou d'un **type compatible**. Lorsque ce n'est pas le cas, il y a **promotion** de l'opérande de type le plus faible vers le plus grand.

Fonctions mathématiques

Elles agissent sur des paramètres à valeurs réelles et donnent un résultat réel.

Pour les utiliser

```
import java.lang.Math;
```

1.7 Affectation interne

Affectation interne

```
nomVar = expression;
```