Département d'Informatique
DUT Informatique S1
Année 2014/2015
Algorithmique et programmation

Algorithmique et programmation

TP 3 : Le langage Java, conditionnelles

1 Structures de contrôle conditionnelles

1.1 Expressions booléennes

Les expressions booléennes sont des expressions ne pouvant prendre que deux valeurs : *vrai* et *faux*. Les opérations courantes entre expressions booléenne sont

- la *négation* d'une expression (notée ¬), qui inverse la valeur de l'expression ;
- la *conjonction* ou *et logique* de deux expressions (notée ∧), qui vaut *vrai* si les deux expressions valent vrai, et qui vaut *faux* sinon ;
- la *disjonction* ou *ou logique* de deux expressions (notée ∨), qui vaut *vrai* si au moins une des deux expressions vaut *vrai*, et qui vaut *faux* sinon.

En Java, le type des expressions, des variables et des constantes booléennes est **boolean**. Les constantes booléennes sont notées **true** pour la valeur *vrai* et **false** pour la valeur *faux*. Les opérateurs de comparaisons == (égalité), != (différence), > (supérieur à), < (inférieur à), >= (supérieur ou égal à) et <= (inférieur ou égal à) produisent des expressions booléennes. Les opérateurs booléens sont notés ! (négation), && (conjonction) et || (disjonction).

Par exemple, après exécution du fragment de code suivant, le booléen `b` vaut **true** si et seulement si la valeur entrée par l'utilisateur pour `x` est comprise entre 0 et 10 (exclus).

```
int x;
x = input.nextInt();
boolean b = x > 0 && x < 10;
```

NB : l'évaluation d'une expression booléenne s'effectue de gauche à droite et s'arrête dès que la valeur de l'expression est connue. Par exemple dans `A && B`, si `A` vaut *faux*, B n'est pas évaluée car, quelle que soit la valeur de `B`, la valeur de l'expression `A && B` est *faux*. De même dans `A || B`, `B` n'est pas évaluée si `A` vaut *vrai*, car l'expression `A || B` vaut alors *vrai*.

1.2 La structure if

La structure de contrôle **if** permet d'effectuer des traitements conditionnels. Sa syntaxe est la suivante :

```
if (expression booléenne) {
    // bloc d'instructions A
} else {
    // bloc d'instructions B
}
```

Lors de l'exécution du programme, si l'évaluation de l'expression booléenne donne *vrai*, alors le bloc d'instructions A est exécuté, sinon le bloc d'instructions B est exécuté.

Si le bloc d'expression B est vide, le mot clef **else** et le bloc qui suit peuvent être omis :

```
if (expression booléenne) {
    // bloc d' instructions
}
```

Lors de l'exécution du programme, le bloc d'instructions ne sera exécuté que si l'évaluation de l'expression booléenne donne *vrai*.

Enfin, si le bloc d'expression se résume à une seule expression, les accolades peuvent être enlevées. Par exemple :

```
int x, y;
// ...
if (x > y)
    System.out.println("x est plus grand");
else
    System.out.println("y est plus grand");
```

1.3 La structure switch

La structure de contrôle **switch** permet d'effectuer des traitements suivant des valeurs possibles d'une expression entière. Sa syntaxe est la suivante :

```
switch (expression) {
case constante1:
    // ...
    break;
case constante2:
    // ...
    break;
// ...
default:
    // ...
    break;
}
```

L'expression est évaluée, puis l'exécution du programme continue à partir de l'étiquette **case** dont la constante est égale à la valeur de l'expression. L'exécution continue alors jusqu'à la première instruction **break** rencontrée. Si aucune étiquette ne correspond, l'exécution continue à partir de l'étiquette spéciale **default** si elle existe. L'étiquette **default** peut en effet être omise.

Attention, si une instruction **break** n'est pas mise, l'exécution peut continuer au cas suivant.

Par exemple, le fragment de code suivant affichera `A` si x vaut 0, `BC` si x vaut 1, `C` si x vaut 2, et `D` dans tous les autres cas.

```
int x;
// ...
switch (x) {
case 0:
    System.out.print("A");
    break;
case 1:
    System.out.print("B");
case 2:
    System.out.print("C");
    break;
default:
```

```
        System.out.print("D");
        break;
    }
```

1.4 L'opérateur ternaire

L'opérateur ternaire est un opérateur conditionnel permettant de construire des expressions dont la valeur sera déterminée par l'évaluation d'une expressions booléenne. Sa syntaxe est la suivante :

```
expression booléenne ? expression A : expression B
```

La valeur d'une telle expression est celle de l'expression A si l'expression booléenne vaut *vrai*, et celle de l'expression B sinon. NB : Le type des expressions A et B doivent être compatibles.

Par exemple, après l'exécution du fragment de code suivant, la variable z est égale à plus grande valeur entre celle de x et celle de y :

```
int x, y, z;
// ...
z = (x > y ? x : y);
```

c'est (dans ce cas) équivalent à faire :

```
int x, y, z;
// ...
if (x > y)
    z = x;
else
    z = y;
```

2 Exercices

2.1 Bonjour

Écrire un programme qui demande à l'utilisateur son prénom, son année de naissance et son sexe et qui affiche le message suivant :

Bonjour Toto, tu es âgé de 10 ans.

Il faudra bien sûr mettre le prénom de l'utilisateur à la place de « Toto » ainsi que son âge réel. Les mots « âgé » et « an » devront être correctement accordés.

Indications

- Une valeur numérique peut être utilisée pour coder le sexe. Par exemple 0 pour « masculin » et 1 pour « féminin ».
- L'expression suivante permet de récupérer l'année courante (type **int**) :

```
Calendar.getInstance().get(Calendar.YEAR)
```

2.2 Années bissextiles

Écrire un programme qui demande une année à l'utilisateur, et qui affiche si l'année entrée est bissextile ou non. Une année est bissextile si elle est divisible par 4, mais pas par 100. Les années divisibles par 400 sont cependant bissextiles.

Testez votre programme avec les années 1900, 1984, 2000 et 2010. Vous pouvez vous servir de la commande système `cal` pour vérifier vos résultats.

2.3 Équations du 2nd degré

Écrire un programme permettant de résoudre des équations du second degré de la forme

$$a \times x^2 + b \times x + c = 0.$$

L'utilisateur entrera les valeurs de a, b et c. Le programme donnera le nombre de solutions, leur type (réel ou complexe) et les affichera.

Rappels Pour résoudre de telles équations, il faut d'abord calculer la valeur du discriminant

$$\Delta = b^2 - 4 \times a \times c.$$

On distingue alors trois cas, suivant la valeur du discriminant :

- Cas $\Delta = 0$: une seule solution réelle

$$x = \frac{-b}{2 \times a}$$

- Cas $\Delta > 0$: deux solutions réelles

$$x = \frac{-b + \sqrt{\Delta}}{2 \times a} \qquad \text{et} \qquad x' = \frac{-b - \sqrt{\Delta}}{2 \times a}$$

- Cas $\Delta < 0$: deux solutions complexes

$$x = \frac{-b + \sqrt{-\Delta}i}{2 \times a} \qquad \text{et} \qquad x' = \frac{-b - \sqrt{-\Delta}i}{2 \times a}$$

Extensions possibles

- Faites en sorte que votre programme affiche par exemple « `1 - 2i` » plutôt que « `1 + -2i` ».
- Complétez votre programme pour traiter le cas où $a = 0$.

2.4 Calculatrice simple

Écrire un programme lisant une expression arithmétique simple du type *opérande opérateur opérande* et qui affiche le résultat. L'opérateur pourra être n'importe laquelle des quatre opérations de base : $+$, $-$, $\times$, $/$.

Indications

- L'opérateur est représenté par un simple caractère. Il pourra être lu dans une variable de type caractère (**char** en Java) en utilisant `input.next().charAt(0)`.
- Faites deux variantes de votre programme : l'une utilisant des structures de contrôle **if** ... **else** ... , et l'autre utilisant une structure **switch** ... **case** ...

Extensions possibles Essayez d'ajouter d'autres opérateurs.