IFT 1010 - Programmation 1

Conditions 1

Professeurs:
Sébastien Roy & Balázs Kégl

B. Kégl
Département d'informatique et de recherche opérationnelle
Université de Montréal

automne 2004

Au programme

- Instructions conditionnelles: `if - else` et `switch`
- Blocs d'instruction
- Expressions relationnelles et booléennes
- [Tasso:3], [Niño:6]

Les instructions de sélection

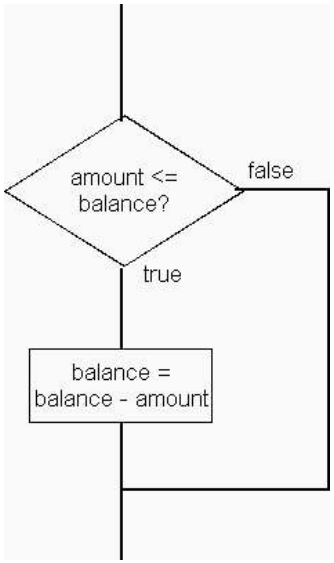
- Objectif
  - permettre au programme de choisir entre les directions
  - l'ordre d'exécution peut dépendre de l'information qui n'existe que pendant l'exécution
- Instructions
  - `if` et `switch`
- Type primitif logique
  - `boolean`: `true` ou `false`

L'instruction `if`

- Exemple
  - Retirer l'argent seulement s'il y en a assez sur le compte

```
double balance = 1345.89;
System.out.println("How much to withdraw?");
double amountToWithdraw = Keyboard.readDouble();
if (balance > amountToWithdraw)
    balance = balance - amountToWithdraw;
System.out.println("New balance is " + balance);
```

L'instruction `if`

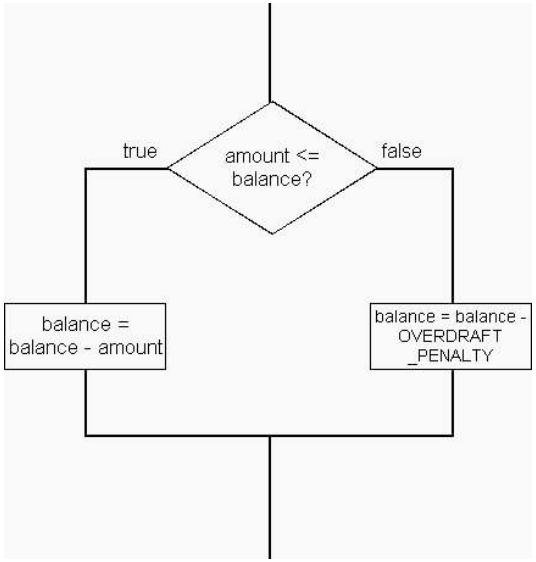


L'instruction `if ... else`

- Exemple
  - Sinon, ajouter une pénalité

```
double balance = 1345.89;
final double OVERDRAFT_PENALTY = 25;
System.out.println("How much to withdraw?");
double amountToWithdraw = Keyboard.readDouble();
if (balance > amountToWithdraw)
    balance = balance - amountToWithdraw;
else
    balance = balance - OVERDRAFT_PENALTY;
```


L'instruction `if ... else`



Les instructions `if` et `if ... else`

- Syntaxe de `if`

```
if (<condition>)
   <instruction>
```

- Syntaxe de `if ... else`

```
if (<condition>)
   <instruction1>
else
   <instruction2>
```

L'instruction `if ... else`

- Exécution de `if ... else` (la sémantique)
  - `<condition>` est évaluée
  - si elle est vraie, `<instruction1>` est exécutée
  - si elle est fausse, `<instruction2>` est exécutée
  - après l'instruction `if ... else`, l'exécution se poursuit à partir de la prochaine instruction

Le bloc d'instructions

- Exemple

```
if (balance > amountToWithdraw)
{
    balance = balance - amountToWithdraw;
    System.out.println(
        "Succesful withdraw, new balance is "
         + balance);
}
else {
    balance = balance - OVERDRAFT_PENALTY;
    System.out.println(
        "Withdraw refused, penalty applied, "
        + "new balance is " + balance);
}
```


Le bloc d'instructions

- Syntaxe

```
{
   <instruction_1>
   . . .
   <instruction_n>
}
```

- alignez les accolades
- utilisez toujours les blocs, même s'il n'y a qu'une instruction

Comparaisons

- Opérateurs relationnels
  - pour comparer des nombres

opérateur de Java	notation en math	description
`>`	$>$	supérieur à
`>=`	$\geq$	supérieur ou égal à
`<`	$<$	inférieur à
`<=`	$\leq$	inférieur ou égal à
`==`	$=$	égal à
`!=`	$\neq$	n'est pas égal à

Comparaisons

- Nombres en point flottant
  - erreurs d'arrondi

```
double r = Math.sqrt(2);
double d = r * r - 2;
if (d == 0)
   System.out.println("sqrt(2)^2 minus 2 is 0");
else
   System.out.println(
      "sqrt(2)^2 minus 2 is not 0 but " + d);
```

  - résultat:

```
sqrt(2)^2 minus 2 is not 0 but 4.440892098500626E-16
```


Comparaisons

- Nombres en point flottant
  - tester s'ils sont proches

comparaison absolue: $|x - y| \leq \epsilon$

comparaison relative: $\dfrac{|x - y|}{\max(|x|, |y|)} \leq \epsilon$

```
final double EPSILON = 1E-14;
if (Math.abs(x - y) <= EPSILON)
   . . .
if (Math.abs(x - y)/Math.max(Math.abs(x),Math.abs(y)) <=
    EPSILON)
   . . .
```


Comparaisons

- Chaînes
  - `if (str1.equals(str2)) ...`,
    PAS `if (str1 == str2) ...`!!!
  - ignorer les minuscules/majuscules:
    `if (str1.equalsIgnoreCase(str2)) ...`
  - comparer en ordre alphabétique: `str1.compareTo(str2)`

Le type `boolean` et les opérateurs conditionnels

- Le type `boolean`
  - stocker les valeurs logiques
  - littéraux: `true`, `false`
- Opérateurs conditionnels

opérateur de Java	notation en math	vrai si
`a && b`	$a \wedge b$	les deux, a et b sont vrais
`a || b`	$a \vee b$	soit a soit b est vrai
`!a`	$\overline{a}$ ou $\neg a$	a est faux

Tables de vérité

`a && b`

	b est `true`	b est `false`
a est `true`	`true`	`false`
a est `false`	`false`	`false`

`a || b`

	b est `true`	b est `false`
a est `true`	`true`	`true`
a est `false`	`true`	`false`

Le type `boolean` et les opérateurs conditionnels

- Exemples
  - `if (0 < amount && amount < 1000) . . .`
  - `if (input.equals("S") || input.equals("M")) . . .`
  -
```
boolean married;
. . .
if (married) // PAS if (married == true)
   . . .
else
   . . .
```


La préséance des opérateurs

Nom de l'opérateur	Type d'opérateur	opérateur
in/décrémentation	numérique	`++ --`
négation	booléen	`!`
signe unaire	numérique	`+ -`
cast	numérique	`(<type>)`
multiplication, division, reste	numérique	`* / %`
addition (concaténation), soustraction	numérique	`+ -`
comparaison	relationnel	`== != < > <= >=`
ET logique	booléen	`&&`
OU logique	booléen	`||`
affectation	numérique	`=`
affectation-opérateur	numérique	`+= -= /= *=`