Algorithmique et programmation par objets

Inf F3
Licence 2 MIASHS
Université Grenoble Alpes

Jerome.David@univ-grenoble-alpes.fr

2022-2023

http://miashs-www.u-ga.fr/~davidjer/inff3/

Cours 3 – Les opérateurs

- Affectation : différences entre types simples et des objets
- Opérateurs arithmétiques
- Opérateurs relationnels
- Opérateurs logiques
- Opérateurs sur représentation binaire
- Opérateur de concaténation

Compléments : les commentaires

- 2 syntaxes pour les commentaires
  - Sur plusieurs lignes
    - Commencent par `/*` et finissent par `*/`

```
/* Un commentaire
 * sur plusieurs
 * lignes
 */
```

  - Sur une seule ligne
    - Commencent par `//` et finissent à la fin de la ligne

```
// un commentaire sur une seule ligne
```

Attention, il existe aussi les commentaires javadoc qui commencent par /** et finissent par */ (voir TP1)

Les opérateurs

- Une expression est construite à partir de littéraux, d'identificateurs (variables ou constantes), d'opérateurs et de parenthèses
  - Certains opérateurs sont plus prioritaires que d'autres
    - http://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html
    - Si vous ne les connaissez pas, utilisez des parenthèses
      - C'est plus sûr et surtout plus lisible !

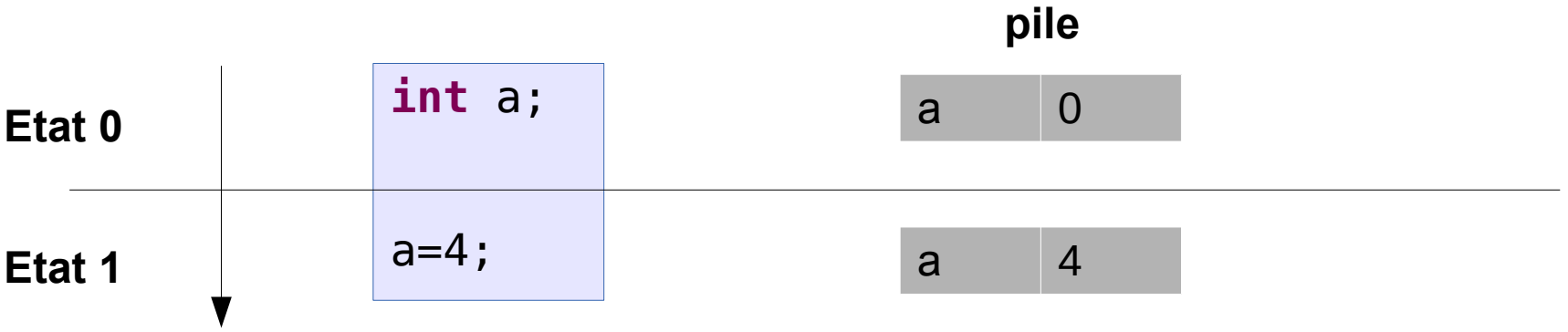
L'opérateur d'affichage

- Dans la classe System, il y a des variables de classe représentant les consoles d'affichage :
  - System.out : la sortie « standard »
  - System.err : la sortie d' « erreur »
- Ces variables sont de type PrintStream
  - Cette classe définie les méthodes :
    - println(… data) : affiche data puis commence une nouvelle ligne
    - print(... data) : affiche data
    - Vous pouvez voir que les méthodes println et print sont définies pour plusieurs types d'arguments

```
System.out.println("coucou");
System.out.println(1);
System.out.println(true);
```

L'affectation

- L'affectation est réalisée via l'opérateur =
  - « prend la valeur de la partie de droite et copie la dans la partie de gauche »
    - La partie de droite peut être une constante, une variable ou une expression qui produit une valeur
    - La partie de gauche doit être une variable
- Pour les types simple l'affection est triviale



L'affectation pour les objets

- Les objets sont manipulés par référence
- L'affectation ne copie pas l'objet d'un endroit à un autre mais sa référence

```
class Ampoule {
    boolean allumee;
    int intensite;
}
```

```
public class ExampleAffectation {
    public static void main(String[] args) {
        Ampoule a = new Ampoule();
        a.allumee=true;
        a.intensite=50;

        Ampoule b = a ;
        b.intensite=30;

        System.out.println(a.intensite);
    }
}
```

Question :
Qu'affichera ce programme ?

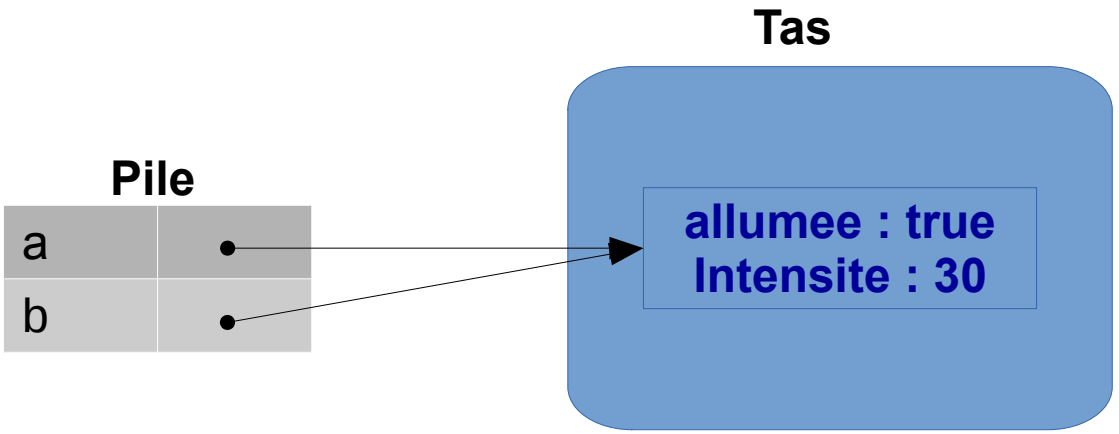
L'affectation pour les objets

Réponse : 30

```
public class ExampleAffectation {
    public static void main(String[] args) {
        Ampoule a = new Ampoule();
        a.allumee=true;
        a.intensite=50;

        Ampoule b = a ;
        b.intensite=30;

        System.out.println(a.intensite);
    }
}
```



L'instruction `b=a` fait que `b` et `a` référencent le même objet

Opérateurs arithmétique

- Ils prennent un ou plusieurs arguments et retournent une nouvelle valeur
- Les opérateurs arithmétique classiques

  `+` (addition ou plus unaire), `-` (soustraction ou opposé unaire), `*` (multiplication), `/` (division), `%` (modulo)
- Les raccourcis (opération + affectation)

```
x+=4; // équiv. à x=x+4;
x-=5; // etc.
x*=3;
x/=2;
x%=2;
```

Incrémentation et décrémentation

- Opérateurs d'incrémentation et de décrémentation

  ++ incrémente d'une unité son opérande

  - préincrément : ++x et postincrément x ++

```
i = 1;
j = ++i; // j et i valent 2
```

```
i = 1;
j = i++; // j vaut 1, i vaut 2
```

  -- décrémente d'une unité son opérande

  - prédécrément : --x et postdécrément x--

```
i = 1;
j = --i; // j et i valent 0
```

```
i = 1;
j = i--; // j vaut 1, i vaut 0
```

Opérateurs relationnels

- Ils évaluent un relation entre les opérandes et retournent un booléen
  - `true` si la relation entre opérandes est satisfaite
  - `false` sinon
- Les opérateurs

  < (inférieur), <= (inférieur ou égal), == (égal),

  != (différent), > (supérieur), >= (supérieur ou égal)

Opérateurs relationnels cas spécial des objets

- == et != sont définis sur les objets
  - Mais ils testent les références et non le contenu des objets !

```
public class Egalite {
    public static void main(String[] args) {
        String a=new String("toto");
        String b=new String("toto");
        System.out.println(a==b);
    }
}
```

Affiche false !

  - Pour tester l'égalité entre objets, il faut utiliser :

```
a.equals(b);
```

Attention cela ne fonctionne pas automatiquement sur des classes que vous créez

Opérateurs logiques

- Les opérateurs logiques :
  - ET → &&
  - OU → ||
  - NON → !
  - Ils retournent une valeur booléenne qualifiant la relation entre les opérandes (qui sont eux aussi des booléens)
- Ils ont la propriété de « court-circuiter » l'évaluation
  - Si l'opérande de gauche du && est évaluée à false, alors la partie de droite n'est pas évaluée
  - Si l'opérande de gauche du || est évaluée à true, alors la partie droite n'est pas évaluée

Opérateurs logiques

```
class Ampoule {
    boolean allumee;
    int intensite;

    boolean allumeeAPleinePuissance() {
            System.out.println("hello");
            return allumee && intensite==100;
    }
}
```

```
public class Egalite {
    public static void main(String[] args) {
        Ampoule a1 = new Ampoule();
        Ampoule a2 = new Ampoule();

        a1.allumee=true;
        a2.allumee=true;
        a2.intensite=100;

        boolean res = a1.allumeeAPleinePuissance()&&a2.allumeeAPleinePuissance();
        System.out.println(res);
    }
}
```

Qu'affiche ce programme ?

Les opérateurs orientés bits

- Ils permettent de manipuler les bits dans les types de données primitifs
  - Bas niveau, et donc pas très utilisés

op	description	exemple	résultat
~	Complément à 1	~1	
&	et	5&3	1
|	ou	5|3	7
^	Ou exclusif	5^3	6
<<	Décalage à gauche	5<<1	10
>>	Décalage à droite	5>>1	2
>>>	Décalage à droite non signé		

Les opérateurs de concaténation

- + et += sont également utilisés sur les chaînes de caractères
  - Ils se comportent alors comme la concaténation
  - Les chaînes sont immuables
    - Quand on concatène deux chaînes, alors une autre instance de chaîne est créée.
  - Le symbole + est interprété comme la concaténation si au moins une des opérandes est une chaine

```
String s="";
System.out.println((s+1)+3); // affiche 13
```

```
String s="";
System.out.println(s+(1+3)); // affiche 4
```

```
String s="";
System.out.println(s+1+3); // affiche 13
```