Erreurs de débutants

Erreurs classiques

Le test d'égalité s'écrit ==, et pas =

```
if (a = 1) // !!!
```

n'est pas accepté par le compilateur.

Erreurs classiques

```
if (a == 1); // !!!
  System.out.println("a vaut 1");
```

`a vaut 1` est toujours affiché quelle que soit la valeur de `a`!

Le point-virgule est considéré comme une instruction, qui ne fait rien.
Le code précédent est compris par le compilateur comme:

```
if (a == 1)
  ;
System.out.println("a vaut 1");
```

l'instruction `System.out.println("a vaut 1");` est donc située après le `if`.

Si on utilise des accolades même quand il n'y a qu'une instruction dans le bloc, et qu'on écrive le test de la façon suivante:

```
if (a == 1) {
  System.out.println("a vaut 1");
}
```

l'erreur précédente a beaucoup moins de chance d'arriver.

Erreurs classiques

Ne pas oublier les accolades, l'indentation ne suffit pas:

```
if (n < p)
  System.out.println("n est plus petit que p");
  max = p;
else
  System.out.println("n est plus grand ou egal a p");
```

génère à la compilation l'erreur:

```
error: 'else' without 'if'
```

Voici une meilleure présentation du code précédent:

```
if (n < p)
  System.out.println("n est plus petit que p");

max = p;
else
  System.out.println("n est plus grand ou egal a p");
```

```
System.out.println("Entrez le premier nombre:");
int n = scanner.nextInt();
System.out.println("Entrez le deuxieme nombre:");
int p = scanner.nextInt();

if ((n < p) && (2 * n >= p)) {
  System.out.print("1");
}

if ((n < p) || (2 * n >= p)) {
  System.out.print("2");
}

if (n < p) {
  if (2 * n >= p) {
    System.out.print("3");
  } else {
    System.out.print("4");
  }
}
System.out.println();
```

Qu'affiche ce programme quand l'utilisateur entre **1 et 2** ?

A: 2
B: 24
C: 123
D: 1234

?

```
System.out.println("Entrez le premier nombre:");
int n = scanner.nextInt();
System.out.println("Entrez le deuxieme nombre:");
int p = scanner.nextInt();

if ((n < p) && (2 * n >= p)) {
  System.out.print("1");
}

if ((n < p) || (2 * n >= p)) {
  System.out.print("2");
}

if (n < p) {
  if (2 * n >= p) {
    System.out.print("3");
  } else {
    System.out.print("4");
  }
}
System.out.println();
```

Qu'affiche ce programme quand l'utilisateur entre **1 et 3** ?

A: 2
B: 24
C: 123
D: 1234

?

```
System.out.println("Entrez le premier nombre:");
int n = scanner.nextInt();
System.out.println("Entrez le deuxieme nombre:");
int p = scanner.nextInt();

if ((n < p) && (2 * n >= p)) {
  System.out.print("1");
}

if ((n < p) || (2 * n >= p)) {
  System.out.print("2");
}

if (n < p) {
  if (2 * n >= p) {
    System.out.print("3");
  } else {
    System.out.print("4");
  }
}
System.out.println();
```

Qu'affiche ce programme quand l'utilisateur entre **2 et 1** ?

A: 2
B: 24
C: 123
D: 1234

?

Le type booléen (`boolean`)

Le type `boolean`

Le type `boolean` (pour booléen) est le type des **conditions**.

Il permet de déclarer des **variables contenant la valeur d'une condition**.

Une variable de type booléen est souvent appelée simplement un **booléen**.

Un booléen ne peut prendre que **deux valeurs possibles**: `true` ou `false`

Exemple:

```
int a = 1, b = 2;
boolean test1 = (a == b);
boolean test2 = (a < b);
```

Le type `bool`

Le type `boolean` (pour booléen) est le type des **conditions**.

Il permet de déclarer des **variables contenant la valeur d'une condition**.

Une variable de type booléen est souvent appelée simplement un **booléen**.

Un booléen ne peut prendre que **deux valeurs possibles**: `true` ou `false`

Exemple:

```
int a = 1, b = 2;
boolean test1 = (a == b);
boolean test2 = (a < b);
```

Le type `bool`

Le type `boolean` (pour booléen) est le type des **conditions**.

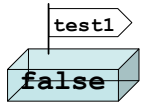
Il permet de déclarer des **variables contenant la valeur d'une condition**.

Une variable de type booléen est souvent appelée simplement un **booléen**.

Un booléen ne peut prendre que **deux valeurs possibles**: `true` ou `false`

Exemple:

```
int a = 1, b = 2;
boolean test1 = (a == b);
boolean test2 = (a < b);
```



Le type `bool`

Le type `boolean` (pour booléen) est le type des **conditions**.

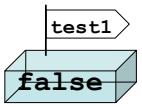
Il permet de déclarer des **variables contenant la valeur d'une condition**.

Une variable de type booléen est souvent appelée simplement un **booléen**.

Un booléen ne peut prendre que **deux valeurs possibles**: `true` ou `false`

Exemple:

```
int a = 1, b = 2;
boolean test1 = (a == b);
boolean test2 = (a < b);
```



Le type `bool`

Le type `boolean` (pour booléen) est le type des **conditions**.

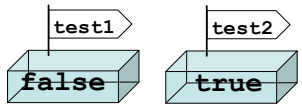
Il permet de déclarer des **variables contenant la valeur d'une condition**.

Une variable de type booléen est souvent appelée simplement un **booléen**.

Un booléen ne peut prendre que **deux valeurs possibles**: `true` ou `false`

Exemple:

```
int a = 1, b = 2;
boolean test1 = (a == b);
boolean test2 = (a < b);
```



On peut initialiser des booléens à l'aide des constantes `false` et `true`.

On peut utiliser des booléens comme des conditions. Par exemple:
- on peut utiliser des opérateurs logiques (`&&`, `||` et `!`) entre booléens;
- on peut utiliser un booléen comme condition dans un `if`.

```
int a = 1, b = 2;

boolean c = true;
boolean d = (a == b);
boolean e = (d || (a < b));

if (e) {
  System.out.println("e vaut true");
}
```

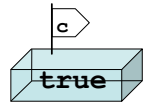
On peut initialiser des booléens à l'aide des constantes `false` et `true`.

On peut utiliser des booléens comme des conditions. Par exemple:
- on peut utiliser des opérateurs logiques (`&&`, `||` et `!`) entre booléens;
- on peut utiliser un booléen comme condition dans un `if`.

```
int a = 1, b = 2;

boolean c = true;
boolean d = (a == b);
boolean e = (d || (a < b));

if (e) {
  System.out.println("e vaut true");
}
```



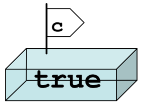
On peut initialiser des booléens à l'aide des constantes `false` et `true`.

On peut utiliser des booléens comme des conditions. Par exemple:
- on peut utiliser des opérateurs logiques (`&&`, `||` et `!`) entre booléens;
- on peut utiliser un booléen comme condition dans un `if`.

```
int a = 1, b = 2;

boolean c = true;
boolean d = (a == b);
boolean e = (d || (a < b));

if (e) {
  System.out.println("e vaut true");
}
```



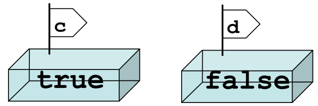
On peut initialiser des booléens à l'aide des constantes `false` et `true`.

On peut utiliser des booléens comme des conditions. Par exemple:
- on peut utiliser des opérateurs logiques (`&&`, `||` et `!`) entre booléens;
- on peut utiliser un booléen comme condition dans un `if`.

```
int a = 1, b = 2;

boolean c = true;
boolean d = (a == b);
boolean e = (d || (a < b));

if (e) {
  System.out.println("e vaut true");
}
```



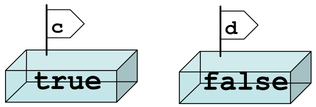
On peut initialiser des booléens à l'aide des constantes `false` et `true`.

On peut utiliser des booléens comme des conditions. Par exemple:
- on peut utiliser des opérateurs logiques (`&&`, `||` et `!`) entre booléens;
- on peut utiliser un booléen comme condition dans un `if`.

```
int a = 1, b = 2;

boolean c = true;
boolean d = (a == b);
boolean e = (d || (a < b));

if (e) {
  System.out.println("e vaut true");
}
```



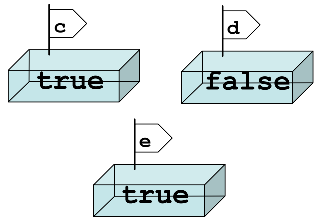
On peut initialiser des booléens à l'aide des constantes `false` et `true`.

On peut utiliser des booléens comme des conditions. Par exemple:
- on peut utiliser des opérateurs logiques (`&&`, `||` et `!`) entre booléens;
- on peut utiliser un booléen comme condition dans un `if`.

```
int a = 1, b = 2;

boolean c = true;
boolean d = (a == b);
boolean e = (d || (a < b));

if (e) {
  System.out.println("e vaut true");
}
```



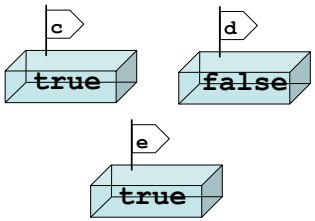
On peut initialiser des booléens à l'aide des constantes `false` et `true`.

On peut utiliser des booléens comme des conditions. Par exemple:
- on peut utiliser des opérateurs logiques (`&&`, `||` et `!`) entre booléens;
- on peut utiliser un booléen comme condition dans un `if`.

```
int a = 1, b = 2;

boolean c = true;
boolean d = (a == b);
boolean e = (d || (a < b));

if (e) {
  System.out.println("e vaut true");
}
```



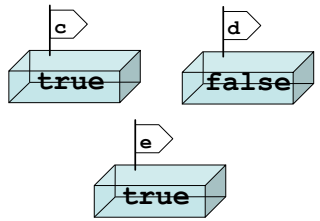
On peut initialiser des booléens à l'aide des constantes `false` et `true`.

On peut utiliser des booléens comme des conditions. Par exemple:
- on peut utiliser des opérateurs logiques (`&&`, `||` et `!`) entre booléens;
- on peut utiliser un booléen comme condition dans un `if`.

```
int a = 1, b = 2;

boolean c = true;
boolean d = (a == b);
boolean e = (d || (a < b));

if (e) {
System.out.println("e vaut true");
}
```



Les booléens sont utiles pour de nombreux problèmes, nous rencontrerons des exemples concrets dans la suite du cours.