

Introduction à la Programmation 1

Séance 1 de cours/TD

Université Paris-Diderot

Objectifs:

- Utiliser le langage JAVA comme une calculatrice.
- Identifier et donner un sens aux différentes constructions du langage (déclaration et utilisation des variables, boucles for, conditionnelles, procédures et fonctions).
- Apporter une modification mineure à un programme existant.

```
1  int stopSecs = (stopHour * 60 + stopMin) * 60;
2  int startSecs = (startHour * 60 + startMin) * 60;
3  int numberOfSecs = stopSecs - startSecs;
4  int alertCode = 0;
5  for (int i = 0; i < numberOfSecs; i++) {
6      waitUntilNextSecond ();
7      if (numberOfSecs - i < 30)
8          alertCode = 1;
9      drawInt (numberOfSecs - i, alertCode);
10 }
```

Listing 1 – Que fait ce programme ?

1 Le langage des expressions arithmétiques

Sous-langage des expressions arithmétiques [COURS]

- Le type `int` est l'ensemble des valeurs entières comprises entre -2^{31} et $2^{31}-1$.
- Une **expression arithmétique de type `int`** peut être :
 - (a) une constante entière (0, 1, 2, -1, -2, -2147483648, 2147483647 ...);
 - (b) deux expressions séparées par une opération arithmétique binaire ($1 + 2$, $1 + 2 * 3 / 4$, ...);
 - (c) une expression entourée de parenthèses ($(1 + 2)$, $(1 + 2 * 3 / 4)$, ...);
 - (d) une expression précédée du signe moins (-2 , $-(1 + 2)$, ...).
- Les opérateurs ont des priorités relatives, par exemple $\{*, /, \%\} \preceq \{+, -\}$ où \preceq signifie "être prioritaire sur".
- Les opérateurs binaires précédents sont associatifs à gauche.
- Par exemple, l'évaluation de l'expression $1 + 2 * 3 - 4$ se décompose en (i) $2 * 3$ donne 6; (ii) $1 + 6$ en 7 et (iii) $7 - 4$ donne 3.
- Les opérations (+, -, /, *, ...) ont le sens usuel *tant que l'on reste entre les bornes du type `int`*. Dans le type `int`, la division / est une *division entière*. Par exemple, $31/7$ vaut 4.
- L'opérateur % désigne le *modulo* : $a\%b$ est le reste dans la division entière de a par b. Par exemple, $31\%7$ vaut 3 car $31 = 7 \times 4 + 3$.

Exercice 1 (Java comme une calculatrice, ☆)

Prévoir l'évaluation des expressions arithmétiques suivantes :

```
1 6 * 7 + 3
2 6 * (7 + 3)
3 45 / 7
4 3 * 7 / 4
5 (3 * 7) / 4
6 (45 / 7) * 7 + 45 % 7
7 (1 + 2 - 3 + 4 - 5 + 6 - 7 + 8 - 9 + 10 - 11 + 12 - 13) / (1 - 2 + 3 - 4
  + 5 - 6 + 7 - 8 + 9 - 10 + 11 - 12 + 13)
```

Réponse

□

2 Utilisation des variables

Déclaration et utilisation des variables _____[COURS]

- Une **variable** a un *nom* et contient une *valeur*, c'est-à-dire le résultat d'un calcul.
- On déclare une variable `x` de type `int` qui contient initialement le résultat du calcul `6 * 7` ainsi :

```
1 int x = 6 * 7;
```

- Dans l'exemple précédent, la valeur de la variable `x` est 42.
- Exemples de noms de variable : `x`, `y`, `foo`, `foobar42`...
- On peut utiliser la valeur d'une variable dans une expression en faisant référence à son nom. Ainsi, l'expression "`x + 1`" vaut 43 si `x` vaut 42.
- On peut changer la valeur d'une variable qui a été déclarée auparavant en lui *affectant* le résultat d'un nouveau calcul. L'opérateur d'affectation est « `=` » :

```
1 x = 2 * 10;
```

Instructions _____[COURS]

- Une expression calcule une valeur tandis qu'une instruction a un effet sur la machine.
- On dit qu'une machine exécute une instruction.
- On peut utiliser une **procédure** en écrivant son nom suivi d'un ou plusieurs paramètres séparés par des virgules et entourés de parenthèses. Par exemple :

```
1 printInt(1 + 2);
```

affiche 3 à l'écran. Un autre exemple :

```
1 putPixel(0, 0, 255, 255, 255);
```

affiche un pixel blanc en position (0, 0) d'une image.

- Une procédure est définie en termes d'une suite d'instructions (les détails viendront plus tard).
- On peut écrire des suites d'instructions qui sont exécutées dans l'ordre par la machine.

Exercice 2 (Nommer, ☆)

Quelle est la valeur des variables, à la suite des instructions suivantes.

```
1 int x = 6 * 7;
```

```
2  int y = x + x;
```

Réponse ☐

Exercice 3 (Affectations, **)

Quelle est la valeur des variables, à la suite des instructions suivantes.

```
1  int x = 1;
2  int y = 4;
3  x = y + 2;
```

Réponse ☐

Exercice 4 (Affectations, **)

Quelle est la valeur des variables, à la suite des instructions suivantes.

```
1  int x = 1;
2  int y = x - 1;
3  x = 2;
```

Réponse ☐

Exercice 5 (Affectations, **)

Quelle est la valeur des variables, à la suite des instructions suivantes.

```
1  int x = 1;
2  x = x - 1;
```

Réponse ☐

3 Fonctions

Nommer un calcul comme une fonction _____ [COURS]

- La fonction qui attend en paramètre une valeur `x` de type `int` et calcule une valeur de type `int` qui est le triple de `x` s'écrit :

```
1 public static int triple (int x) {
2     return (x * 3);
3 }
```

- On peut utiliser une fonction en écrivant son nom suivi de son paramètre entouré de parenthèses, comme par exemple dans l'expression :

```
1 triple (2) * 6
```

- Dans cet exemple, `triple (2)` vaut 6 donc l'expression vaut 36.
- Voici des exemples de noms de fonction : `triple`, `add3`, `fooBar`, ...
- Une fonction peut prendre plusieurs paramètres.
- On peut utiliser des fonctions définies dans des bibliothèques de fonctions.

Exercice 6 (Utiliser une fonction, *)

Étant donnée la fonction suivante, que vaut l'expression `twice(3)` ?

```
1 public static int twice (int x) {
2     return (2 * x);
```

```
3 }

```

Réponse ☐

Exercice 7 (Utiliser une fonction, **)

On considère la fonction suivante.

```
1 public static int cube (int x) {
2     return (x * x * x);
3 }

```

Quelle est la valeur de la variable a à la suite des instructions suivantes ?

```
1 int b = 5;
2 int a = 3;
3 a = b - a;
4 a = cube(a);

```

Réponse ☐

Exercice 8 (Écrire une fonction, **)

Voici une fonction

```
1 public static int f(int x) {
2     return (x * x + 5);
3 }

```

Quel est son nom ? Que calcule-t-elle ? Modifier son corps pour qu'elle calcule la division entière par 3. Modifier son nom pour qu'elle se nomme third.

Réponse

☐

Exercice 9 (Écrire une fonction, **)

Écrire des fonctions effectuant les calculs suivants :

1. La puissance 5 d'un entier donné en paramètre
2. Le produit de deux entiers donnés en paramètre moins leur somme
3. Le produit de trois entiers donnés en paramètre au carré

Réponse

☐

4 Conditionnelle

Instruction conditionnelle [COURS]

- Une instruction conditionnelle permet d'exécuter des instructions en fonction d'une condition.
- On écrit par exemple

```
1  int x, y;  
2  x = ...;  
3  y = ...;  
4  if (x <= 0) {  
5      y = x;  
6  } else {  
7      y = -x;  
8  }  
9
```

pour affecter la valeur de x à y si $x \leq 0$ ou pour lui affecter la valeur $-x$ dans le cas contraire.

- Les conditions peuvent par exemple être des comparaisons entre deux expressions de type `int` ($e1 == e2$, $e1 != e2$, $e1 < e2$, $e1 <= e2$, $e1 > e2$, $e1 >= e2$).

Exercice 10 (Le max, ★)

À la suite des instructions ci-dessous, quelle est la valeur de la variable max ?

```
1  int x = 3;  
2  int y = 4;  
3  int max = 0;  
4  if (x > y) {  
5      max = x;  
6  } else {  
7      max = y;  
8  }
```

Transformer la suite d'instructions pour calculer le minimum de x et y et le mettre dans une variable min.

Réponse

□

Exercice 11 (Différents tests, ★★)

Quelle est la valeur des variables a, b et c après la suite d'instructions suivante :

```
1  int a = 2;  
2  int b = a * a + 3;  
3  int c = b - a;  
4  if (c == a) {  
5      a = 1;  
6  } else {  
7      a = a + 3;  
8  }  
9  if ((b + c) < a) {  
10     b = 2;  
11 } else {  
12     b = 4;  
13 }  
14 if (b != c * c) {  
15     c = 12;  
16 } else {  
17     c = -6;
```

Réponse

□

5 Boucles

Boucles [COURS]

- Une boucle permet de répéter plusieurs fois les mêmes instructions.
- Le numéro de l'itération est disponible dans une variable qui s'appelle le *compteur de boucle*.
- La boucle suivante affiche les entiers de 0 à 9 :

```
1  for(int i = 0; i <= 9; i++) {
2      printInt(i);
3  }
```

- L'*entête* est formé du mot clé `for` suivi d'une initialisation, d'une condition, et d'une incrémentation, séparées par des points-virgules et entourées de parenthèses ;
- L'*initialisation* `int i = 0` définit le type et la valeur initiale du compteur de boucle ;
- La *condition de boucle* `i <= 9` définit sous quelle condition l'exécution de la boucle continue ;
- L'*instruction d'incrément* `i++`. Cette instruction est équivalente ici à l'instruction `i = i + 1`.
- Le *corps de la boucle*, entre accolades, est exécuté à chaque itération de la boucle.
- La même boucle aurait pu être écrite comme suit :

```
1  for(int i = 0; i < 10; i++) {
2      printInt(i);
3  }
```

Exercice 12 (Afficher des suites d'entiers, ★)

1. Écrivez une boucle qui affiche les 100 premiers entiers, en commençant à 0. Quel est le dernier entier affiché ?
2. Écrivez une boucle qui affiche les 50 premiers entiers pairs, en commençant à 0. Quel est le dernier entier affiché ?
3. Écrivez une boucle qui affiche 1000 fois le nombre 3.

Réponse

□

6 Fonctions et procédures utilisées

```
1 /*  
2  * Affiche un entier sur l'écran.  
3  * x est l'entier à afficher.  
4  */  
5 public static void printInt (int x) {  
6     System.out.println(x);  
7 }
```

7 Réponses aux exercices

Exercice 1, page 2

Compiler et exécuter le programme suivant :

```
1 public class ExoCalc {
2
3     /*
4      * Affiche un entier sur la sortie standard du programme.
5      * x est l'entier à afficher.
6      */
7     public static void printInt (int x) {
8         System.out.println (x);
9     }
10
11     /* Fonction d'entrée du programme. */
12     public static void main (String[] args) {
13         printInt (6 * 7 + 3);
14         printInt (6 * (7 + 3) );
15         printInt (45 / 7);
16         printInt (3 * 7 / 4);
17         printInt ( (3 * 7) / 4);
18         printInt ( (45 / 7) * 7 + 45 % 7);
19         printInt ( (1 + 2 - 3 + 4 - 5 + 6 - 7 + 8 - 9 + 10 - 11 + 12 - 13)
20 /
21                 (1 - 2 + 3 - 4 + 5 - 6 + 7 - 8 + 9 - 10 + 11 - 12 + 13)
22     );
23     }
```

code/ExoCalc.java

Exercice 2, page 3

En exécutant le programme suivant la réponse sera affichée.

```
1 public class ExoValy {
2
3     /*
4      * Affiche un entier sur la sortie standard du programme.
5      * x est l'entier à afficher.
6      */
7     public static void printInt (int x) {
8         System.out.println (x);
9     }
10
11     /* Fonction d'entrée du programme. */
12     public static void main (String[] args) {
13         int x = 6 * 7;
14         int y = x + x;
15         printInt (x);
16         printInt (y);
17     }
18 }
```


Exercice 3, page 3

En exécutant le programme suivant la réponse sera affichée.

```
1 public class ExoValx {
2
3     /*
4      * Affiche un entier sur la sortie standard du programme.
5      * x est l'entier à afficher.
6      */
7     public static void printInt (int x) {
8         System.out.println (x);
9     }
10
11     /* Fonction d'entrée du programme. */
12     public static void main (String[] args) {
13         int x = 1;
14         int y = 4;
15         x = y + 2;
16         printInt (x);
17         printInt (y);
18     }
19 }
```

code/ExoValx.java

Exercice 4, page 3

En exécutant le programme suivant la réponse sera affichée.

```
1 public class ExoValx2 {
2
3     /*
4      * Affiche un entier sur la sortie standard du programme.
5      * x est l'entier à afficher.
6      */
7     public static void printInt (int x) {
8         System.out.println (x);
9     }
10
11     /* Fonction d'entrée du programme. */
12     public static void main (String[] args) {
13         int x = 1;
14         int y = x - 1;
15         x = 2;
16         printInt (x);
17         printInt (y);
18     }
19 }
```

code/ExoValx2.java

Exercice 5, page 3

En exécutant le programme suivant la réponse sera affichée.

```
1 public class ExoValx3 {  
2  
3     /*  
4     * Affiche un entier sur la s tr un un un
```

```

4      * Affiche un entier sur la sortie standard du programme.
5      * x est l'entier à afficher.
6      */
7      public static void printInt (int x) {
8          System.out.println (x);
9      }
10
11     public static int cube (int x) {
12         return (x * x * x);
13     }
14
15     /* Fonction d'entrée du programme. */
16     public static void main (String[] args) {
17         int b = 5;
18         int a = 3;
19         a = b - a;
20         a = cube (a);
21         printInt (a);
22     }
23 }

```

code/ExoCube.java

Exercice 8, page 4

Le nom de cette fonction est f. Elle calcule le carré de la variable donnée en paramètre auquel elle ajoute 5. Cette fonction après modification devient :

```

1      public static int third (int x) {
2          return (x / 3);
3      }

```

Exercice 9, page 4

Voilà une réponse possible pour ces fonctions :

```

1.
1      public static int power_five (int y) {
2          return (y * y * y * y * y);
3      }

```

```

2.
1      public static int product_minus_sum(int x, int y) {
2          return ((x * y) - (x + y));
3      }

```

```

1      public static int product_square(int x, int y, int z) {
2          return ((x * y * z) * (x * y * z));
3      }

```

Exercice 10, page 5

Voilà le programme, pour voir la valeur de max.

```
1 public class ExoMax {
2
3     /*
4      * Affiche un entier sur la sortie standard du programme.
5      * x est l'entier à afficher.
6      */
7     public static void printInt (int x) {
8         System.out.println (x);
9     }
10
11     /* Fonction d'entrée du programme. */
12     public static void main (String[] args) {
13         int x = 3;
14         int y = 4;
15         int max = 0;
16         if (x > y) {
17             max = x;
18         }
19         else {
20             max = y;
21         }
22         printInt (max);
23     }
24 }
```

code/ExoMax.java

et voilà la solution calculant le minimum.

```
1 public class ExoMin {
2
3     /*
4      * Affiche un entier sur la sortie standard du programme.
5      * x est l'entier à afficher.
6      */
7     public static void printInt (int x) {
8         System.out.println (x);
9     }
10
11     /* Fonction d'entrée du programme. */
12     public static void main (String[] args) {
13         int x = 3;
14         int y = 4;
15         int min = 0;
16         if (x > y) {
17             min = y;
18         }
19         else {
20             min = x;
21         }
22         printInt (min);
23     }
24 }
```

Exercice 11, page 6

Le programme suivant affiche la valeur des trois variables.

```

1 public class ExoValabc {
2
3     /*
4      * Affiche un entier sur la sortie standard du programme.
5      * x est l'entier à afficher.
6      */
7     public static void printInt (int x) {
8         System.out.println (x);
9     }
10
11     /* Fonction d'entrée du programme. */
12     public static void main (String[] args) {
13         int a = 2;
14         int b = a * a + 3;
15         int c = b - a;
16         if (c == a) {
17             a = 1;
18         }
19         else {
20             a = a + 3;
21         }
22         if ( (b + c) < a) {
23             b = 2;
24         }
25         else {
26             b = 4;
27         }
28         if (b != (c * c) ) {
29             c = 12;
30         }
31         else {
32             c = -6;
33         }
34         printInt (a);
35         printInt (b);
36         printInt (c);
37     }
38 }

```

code/ExoValabc.java

Exercice 12, page 6

```

3.
1 for(int i = 0; i < 100; i++) {
2     printInt (i);
3 }
4 \item
5 \begin{lstlisting}
6 for(int i = 0; i < 50; i++) {

```

```
7   printInt (2 * i);  
8 }
```

```
1  for(int i = 0; i < 1000; i++) {  
2      printInt (3);  
3  }
```