n ve é B D de – B
I t od t ona , nfo at q e t a` a p og a at on I 1)
B t e d déce e 00 — D ée : e e
Doc en a c tt e o d at e t t é ép one nona t o é .

On prendra garde à distinguer les programmes qui ont des entrées sorties (clavier/écran) des méthodes qui ont des paramètres. Les exercices sont indépendants.

Exercice 1.

1.1. Donner la représentation en base 2 de l'entier 108.
Quelles sont ses représentations en base 8 et en base 16?

1.2. Donner la représentation machine de l'entier -108 comme valeur de type `short` (codée sur 16 bits).

1.3. On veut stocker un entier compris entre 0 et 2^{60} (utilisés pour la cryptographie). Que se passe-t-il si on utilise le type `int` ? Les types `float` et `double` ? Quel autre type peut-on utiliser ?

Exercice 2.

Exécuter à la main la séquence d'instructions suivante et en particulier indiquer ce qui est affiché à l'écran.

```
int a , b -2, c 5
float x, y 2
c a * b            b a +
Deug println( c)  Deug println( b)  Deug println( a)
x a / 2
Deug println( x)  Deug println( a / 2)  Deug println( a / y)
```

Exercice 3.

Julien veut préparer son sac pour partir en randonnée. Selon les conditions, il doit emporter plus ou moins de matériel.

- Si la température est inférieure à 15 degrés, il doit prendre des vêtements supplémentaires, qui pèsent au total 1,5 kg.
- Si la pluie est annoncée, il emporte une cape de pluie de 0,5 kg, sauf si la température est supérieure à 25 degrés.
- Il prend une gourde (pleine) de 1 kg, et si la température dépasse 30 degrés, il en prend une deuxième.
- Si son ami Max l'accompagne, c'est Max qui apporte le casse-croûte, sinon Julien doit porter 0,4 kg de plus.
- Si la pluie n'est pas annoncée, Julien prolongera un peu la randonnée, il prend donc 0,2 kg d'aliments supplémentaires (sauf si c'est Max qui apporte le casse-croûte).

Pour chaque paramètre (température, présence de Max, pluie), expliquer quel type de donnée Java est adapté à sa représentation (`int, float, double, boolean, char, String`). Écrire une méthode qui prend en arguments la température, le risque de pluie, la présence de Max et qui renvoie le poids du sac à dos.

⟶ suite ...

Exercice 4.

Cet exercice sera traité sans utiliser de tableau.

Écrire un programme qui demande à l'utilisateur un entier `n`, puis lit une suite de `n` entiers, et affiche combien de fois l'utilisateur a entré deux nombres *successifs* égaux. Les nombres seront lus et traités successivement. Par exemple :

- pour 5 puis la suite $5, 3, 3, 5, 7$ le programme doit répondre 1.
- pour 4 puis la suite $1, 5, 5, 5$ le programme doit répondre 2.
- pour 7 puis la suite $8, 6, 6, 1, 6, 6, 6$ le programme doit répondre 3.

Exercice 5.

5.1. Écrire une méthode `decale` qui reçoit en paramètre un tableau d'entiers `t` (à une dimension) et décale les éléments d'une position vers la gauche de façon circulaire (le premier élément prend la place du dernier).

Si le tableau est initialement `t` | 5 | 2 | 3 | 0 | 8 | 7 | 3 | 1 |,

la méthode le transforme en | 2 | 3 | 0 | 8 | 7 | 3 | 1 | 5 |.

Cette méthode doit modifier le tableau qu'elle reçoit en paramètre, et ne renvoie rien.

5.2. On veut écrire de deux manières différentes une méthode `decaleD` qui reçoit en paramètres un tableau d'entiers `t` et un entier `d` positif et décale les éléments de `d` positions vers la gauche de façon circulaire.

Si on appelle `decaleD(t,4)` et que `t` est le tableau donné initialement ci-dessus, il est transformé en | 8 | 7 | 3 | 1 | 5 | 2 | 3 | 0 |.

- *Première version* : écrire une méthode `decaleD` qui utilise la méthode `decale`.
- *Deuxième version* : écrire une méthode `decaleD2` qui utilise un tableau auxiliaire, où chaque élément sera écrit une seule fois, puis qui recopie ce tableau dans le tableau initial.

Exercice 6.

On veut ici modéliser le jeu de l'Awélé (ou Awalé). Il est composé d'une suite de cases qui contiennent des graines, et est représenté par un tableau d'entiers. Chaque élément du tableau représente le nombre de graines contenues dans la case correspondante. Ainsi une case vide sera associée dans la tableau à un élément de valeur zéro.

6.1. Écrire une méthode `vide` qui renvoie le nombre de cases vides dans un état du jeu donné.

6.2. Écrire une méthode `repartirFin` qui reçoit un tableau d'entiers `t` et un indice `i`, qui enlève les graines la case d'indice `i` et les répartit dans les cases suivantes à raison d'une graine par case. Si on arrive à la fin du tableau et qu'il reste des graines à répartir, on les jette.

Si le tableau est initialement `t` | 5 | 2 | 3 | 0 | 8 | 7 | 3 | 1 |, et que l'indice est 4, on obtient | 5 | 2 | 3 | 0 | 0 | 8 | 4 | 2 |.

On ne demande pas de vérifier que l'indice `i` est acceptable.

6.3. Écrire une méthode `jouer` qui reçoit un tableau d'entiers `t` et demande à l'utilisateur de lui donner le numéro d'une case. La méthode vérifie que le numéro donné est bien un indice du tableau, et que la case correspondante contient au moins une graine.

- Si c'est bien le cas, la méthode appelle `repartirFin`.
- Si ce n'est pas le cas, la méthode demande à l'utilisateur de modifier son choix, jusqu'à ce qu'il donne un choix correct.

6.4. Écrire une méthode `repartirTour` identique à `repartirFin`, sauf que si on arrive à la fin du tableau et qu'il reste des graines à répartir, on continue à partir du début du tableau, et ainsi de suite.

6.5. Écrire une méthode `repartirSauf` identique à `repartirTour`, sauf qu'elle ne repose jamais de graine dans la case qui a été vidée au début (le cas échéant, elle passe à la suivante).