université PARIS DIDEROT PARIS 7

PARTIEL — CONCEPTS INFORMATIQUES CI2 — 24/03/2014 12^{30}-15^{30}

Aucun document ou support autre que le sujet ou les copies d'examen n'est autorisé
(la copie ou les brouillons du voisin ne sont pas des supports autorisés).
Positionner impérativement vos mobiles en mode « avion ».

1 Variables, appels et passages d'arguments

Pour le programme suivant, on rappelle que les caractères en Java peuvent être manipulés comme des nombres. Ainsi, si l'on ajoute par exemple 3 au caractère `'a'` on obtient le troisième caractère qui le suit dans l'ordre alphabétique, soit `'d'`.

```
1  public class Rot13 {
2    public static int VALEUR=13;
3    public static char rot(char c,int v) {
4      if (c<'a'+v)
5        return (char)(c+v);
6      else
7        return (char)(c-v);
8    }
9    public static void rot13(char []m) {
10     int i;
11     for (i=0; i<m.length; i++)
12       m[i] = rot(m[i],VALEUR);
13   }
14   public static void affiche(char []t) {
15     for (int i=0; i<t.length; i++)
16       System.out.print(t[i]);
17     System.out.println();
18   }
19   public static void main(String args[]) {
20     char []t = { 'o','b','a','w','b','h','e' };
21     affiche(t);
22     rot13(t);
23     affiche(t);
24   }
25 }
```

Rot13.java

1. Combien de fonctions sont définies dans le programme `Rot13.java` ? Pour chaque fonction préciser, son nom, son nombre d'arguments (pour chaque argument son type) et le type de sa valeur de retour.
2. Combien d'appels de fonctions sont présents dans le programme ? En faire la liste exhaustive (numéro de ligne et expression d'appel).
3. Si dans le programme on exécutait l'expression `rot('a',13)`, quelle serait sa valeur ?
4. Si dans le programme on exécutait l'expression `rot('x',13)`, quelle serait sa valeur ?
5. Quelle est la durée de vie de la variable `t` de la ligne 14 ? Dans quelle partie de la mémoire cette variable sera t-elle placée ?
6. Quelle est la durée de vie de la variable `VALEUR` définie en ligne 2 ? Dans quelle partie de la mémoire sera t-elle placée ?
7. Que produit à l'écran le programme après qu'il a terminé d'exécuter les instructions de la ligne 23 ?
8. À l'aide d'un diagramme comme l'un de ceux réalisés en cours ou en TD, décrire l'état de la mémoire de la machine après l'exécution de la ligne 10 ? Préciser toutes les valeurs que vous pouvez.

```
import java.util.*;
public class Argh {
  public static void echange(int i,int j) {
    int tmp = i;
    i = j;
    j = tmp;
  }
  public static void f(int [][]t) {
    for (int i=0; i<t.length; i++) {
      for (int j=0; j<t[i].length-1-i; j++) {
        echange(t[i][j],t[t[i].length-j-1][t[i].length-i-1]);
      }
    }
  }
  public static void affiche(int [][]t) {
    System.out.println("-----");
    for (int i=0; i<t.length; i++) {
      for (int j=0; j<t[i].length; j++) {
        System.out.print(t[i][j]+"␣");
      }
      System.out.println();
    }
  }
  public static void main(String args[]) {
    Scanner in = new Scanner(System.in);
    int taille = in.nextInt();
    int [][]tableau = new int[taille][taille];
    for (int i=0; i<taille; i++) {
      for (int j=0; j<taille; j++) {
        tableau[i][j] = in.nextInt();
      }
    }
    affiche(tableau);
    f(tableau);
    affiche(tableau);
} }
```

Argh.java.bad

9. Qu'affiche le programme `Argh.java.bad` si, à l'exécution, l'utilisateur entre au clavier les deux nombres `1` puis `1` ?
10. Qu'affiche le programme si, à l'exécution, l'utilisateur entre au clavier les nombres `2` puis `1`, `2`, `3`, `4` ?
11. Quel problème y a-t-il avec la fonction `echange` ? Corriger le programme en conséquence.

2 Piles

On considère des piles contenant des entiers, on choisit le type `Stack<Integer>` de Java et on se restreint aux opérations `void push(int v)` (qui correspond à empiler), `int pop()` (qui correspond à dépiler) et `boolean empty()` (qui teste si la pile est vide).
Pour chacune des méthodes demandées, l'utilisation d'exactement une pile auxiliaire est possible. Attention à prendre à chaque fois en compte le cas de la pile vide.
Par convention, si l'on part d'une pile vide et si l'on y dépose les entiers `3`, puis `4`, puis `9`, puis `9`, on décrit son contenu par `3,4,9,9` (le fond de pile est à gauche, le sommet à droite). On prend comme exemple témoin une pile `p0` dont le contenu est `3,4,9,9,8,5,3,3,3,7,6,0,0,1`.

1. Écrire une méthode statique `dupliquerTous` qui reçoit en argument une pile et qui la modifie en dupliquant chacun de ses éléments. Ainsi le contenu de `p0` deviendrait `3,3,4,4,9,9,9,9,8,8,5,5,3,3,3,3,3,7,7,6,6,0,0,0,0,1,1`.
2. Écrire une méthode statique `dupliquerFond` qui reçoit en argument une pile et qui la modifie en dupliquant son élément de fond de pile s'il existe. Ainsi le contenu de `p0` deviendrait `3,3,4,9,9,8,5,3,3,3,7,6,0,0,1`.
3. Écrire une méthode statique `dupliquerUn` qui reçoit en argument une pile et un indice `k` et qui modifie cette pile en dupliquant son k-ème élément s'il existe. Le contenu de `p0` avec `k` valant `2` deviendrait `3,4,4,9,9,8,5,3,3,3,7,6,0,0,1`.
4. Écrire une méthode statique `filtrerElement` qui reçoit en argument une pile et un entier `v` et qui modifie cette pile en supprimant tous les éléments égaux à `v` et en conservant les autres dans l'ordre initial. Ainsi le contenu de `p0` avec `v` valant `3` deviendrait `4,9,9,8,5,7,6,0,0,1`.

5. Écrire une méthode statique `filtrerMultiple` qui reçoit en argument une pile et un entier `v` et qui modifie cette pile en supprimant tous les entiers multiples de `v` et en conservant les autres dans l'ordre initial. Ainsi le contenu de `p0` avec `v` valant `3` deviendrait `4,8,5,7,1`.
6. Écrire une méthode statique `supprimerDuplicata` qui reçoit en argument une pile et qui la modifie en remplaçant toutes les séquences d'un même entier par un unique exemplaire de cet entier. Ainsi le contenu de `p0` deviendrait `3,4,9,8,5,3,7,6,0,1`.

3 Traduction de programme

Dans cet exercice, il s'agit de traduire des programmes Java en d'autres programmes Java de forme particulière comme étudiée en cours : le programme principal ne devra pas contenir d'autre variable qu'un entier de nom `instructionCourante`, un booléen de nom `fin` et éventuellement d'un entier de nom `sommetDePile` (des symboles plus « courts » `ic` ou `sp` conviendront !), un tableau d'entiers de nom `memoire` et éventuellement une pile, auquel cas il faudra en détailler les caractéristiques (on rappelle qu'en cours nous avons codé la pile des appels directement dans la mémoire – ce n'est pas obligatoire mais c'est mieux !) ou une variable de type `Scanner` si nécessaire. Le code sera simplement constitué d'une boucle `while` contenant un unique `switch` avec autant de cas que nécessaire. Dans le programme traduit, il est formellement interdit d'employer toute autre construction de haut-niveau : pas de boucle `for`, ni d'autre boucle `while`, ni d'autre `switch`, etc.

```
1  import java.util.*;
2
3  public class Prog1 {
4    public static int v;
5    public static void main(String []a) {
6      Scanner s = new Scanner(System.in);
7      do {
8        System.out.print("Donnez␣un␣nombre␣:␣");
9        v = s.nextInt();
10       if (v%2==1)
11         System.out.println("raté!");
12     } while(v%2==1);
13     System.out.println("Gagné!");
14 } }
```

Prog1.java

1. Traduire le programme.

Dans le programme suivant on utilise un booléen. Pour la traduction, il est possible d'utiliser un entier comme booléen, en convenant que si l'entier vaut 0, il représente la valeur `false` et s'il vaut 1 il représente la valeur `true`.

```
1  import java.util.*;
2
3  public class Prog2 {
4    public static int v;
5    public static boolean b;
6    public static boolean perdant(int valeur) {
7        if (valeur%2==1)
8          System.out.println("raté!");
9        return valeur%2==1;
10   }
11   public static void main(String []a) {
12     Scanner s = new Scanner(System.in);
13     do {
14       System.out.print("Donnez␣un␣nombre␣:␣");
15       v = s.nextInt();
16       b = perdant(v);
17     } while(b);
18     System.out.println("Gagné!");
19 } }
```

Prog2.java

2. Traduire le programme.