le cnam

Programmation avec Java : notions de base
(NFA031)

le cnam

Corrigé pour la session du 11 avril 2013

Aucun document n'est autorisé. Le barème est donné à titre indicatif.

Exercice 1 : exécution de programme (3 points)

Retracez l'exécution du programme suivant en indiquant l'évolution des valeurs des variables, des conditions testées et des affichages. Vous pourrez faire cela au moyen d'un tableau comportant une colonne pour chaque variable, en précisant sur chaque ligne du tableau le numéro de ligne de code exécutée.

```
1  public class Ex13_4{
2      public static void main(String[] args){
3          int x = 0;
4          String st = "abc";
5          boolean bl = true;
6          char ch = 'b';
7          while (bl){
8              x = st.length();
9              bl = (x < 9);
10             if (x % 2 == 0){
11                 st = st + x + ch;
12             }else{
13                 st = st + ch + x;
14             }
15         }
16         Terminal.ecrireStringln("valeur st:" + st);
17     }
18 }
```

La méthode `Terminal.ecrireStringln` affiche la chaîne donnée en argument à l'écran. L'expression `x % 2 == 0` vaut true si x est pair et vaut false si x est impair. La concaténation de chaîne notée + permet d'ajouter un entier ou un caractère à une chaîne de caractères. Par exemple `"xx"+12+'a'` a pour résultat la chaîne de caractères `"xx12a"`.

Notation : 1 point si la réponse est adéquate dans sa forme et les valeurs initiales des variables sont correctes, 1 point si les instructions de la boucles sont exécutées 4 fois, 1 point si les valeurs finales des variables sont bonnes

3:	x=0	st=X	bl=X	ch=X	
4:	x=0	st=abc	bl=X	ch=X	
5:	x=0	st=abc	bl=**true**	ch=X	
6:	x=0	st=abc	bl=**true**	ch=b	
7:	x=0	st=abc	bl=**true**	ch=b	
8:	x=3	st=abc	bl=**true**	ch=b	
9:	x=3	st=abc	bl=**true**	ch=b	
10:	x=3	st=abc	bl=**true**	ch=b	(x%2==0)=**false**

```
13: x=3   st=abcb3         bl=true  ch=b
7:  x=3   st=abcb3         bl=true  ch=b
8:  x=5   st=abcb3         bl=true  ch=b
9:  x=5   st=abcb3         bl=true  ch=b
10: x=5   st=abcb3         bl=true  ch=b   (x%2==0)=false
13: x=5   st=abcb3b5       bl=true  ch=b
7:  x=5   st=abcb3b5       bl=true  ch=b
8:  x=7   st=abcb3b5       bl=true  ch=b
9:  x=7   st=abcb3b5       bl=true  ch=b
10: x=7   st=abcb3b5       bl=true  ch=b   (x%2==0)=false
13: x=7   st=abcb3b5b7     bl=true  ch=b
7:  x=7   st=abcb3b5b7     bl=true  ch=b
8:  x=9   st=abcb3b5b7     bl=true  ch=b
9:  x=9   st=abcb3b5b7     bl=false ch=b
10: x=9   st=abcb3b5b7     bl=false ch=b   (x%2==0)=false
13: x=9   st=abcb3b5b7b9   bl=false ch=b
16: x=9   st=abcb3b5b7b9   bl=false ch=b             ecran= valeur st: abcb3b5b7b9
```

Exercice 2 : tableau de booléens (4 points)

Ecrivez un programme (méthode `main`) qui réalise successivement les opérations suivantes :

1. **(1 point)** Déclaration et initialisation d'un tableau de 10 booléens en lisant les 10 valeurs au clavier.
2. **(1 point)** Tester si toutes les cases du tableau ont la même valeur, que ce soit true ou false. Selon le cas, le message *même valeur* ou *valeurs différentes* est affichée.
3. **(1 point)** Calculer l'indice de la première inversion de valeur, c'est à dire le numéro de la première case du tableau qui contient une valeur différente de la case précédente. Cet indice est affiché, sauf si toutes les valeurs sont identiques. Dans ce cas, afficher le message *pas d'inversion*.
4. **(1 point)** Calculer et afficher le "et logique" de toutes les valeurs du tableau. Vous utiliserez **impérativement** une boucle pour le faire.

```
public class Exo2_3{
    public static void main(String[] args){
      boolean[] tab = new boolean[10];
      for (int i=0; i<tab.length; i++){
        Terminal.ecrireStringln("Entrez un boolean");
        tab[i] = Terminal.lireBoolean();
      }
      boolean tousEgaux = true;
      boolean prem = tab[0];
      int indInv = 0;
      for (int i=0; i<tab.length && tousEgaux; i++){
        if (tab[i]!=prem){
          tousEgaux = false;
          indInv = i;
        }
      }
      if (tousEgaux){
```

```
        Terminal.ecrireStringln("Memes valeurs");
        Terminal.ecrireStringln("Pas d'inversion");
      }else{
        Terminal.ecrireStringln("Valeurs differentes");
        Terminal.ecrireStringln("Premiere inversion a l'indice: "+ indInv);
      }

      boolean etLogique = true;
      for (int i=0; i<tab.length; i++){
        etLogique = etLogique && tab[i];
      }
      Terminal.ecrireStringln("Reusultat du et logique: " +etLogique);
    }
}
```

Exercice 3 : procédures et fonctions (3 points)

Donnez sans justification les affichages produits à l'exécution par le programme suivant :

```
public class ExoMethodes2{
   static boolean isD(char c) {
      return (c>='0' && c <='9');
   }
   static void ch(int i, char d, char [] t) {
      if (0<=i && i < t.length && isD(d))
        t[i] = d;
   }
  /* Convertit un caractère chiffre vers sa valeur numerique */
  static int toDigit(char c) {
     return ((int) c - (int) '0');
  }
  static int eval(char [] tc) {
      int s = 0;
      int p = 1;
      for (int i=tc.length-1; i>=0; i--)  {
        s = s + toDigit(tc[i])*p;
        p= p*10;
      }
      return s;
   }
   static void af (char [] t) {
      for (int i=0; i<t.length; i++)  {
         Terminal.ecrireString(" " + t[i]);
      }
      Terminal.sautDeLigne();
   }

   public static void main (String args[]) {
      char [] tab = {'1', '3', '5'};
      ch(2,'k', tab);
```

```
        af(tab);
        ch(0,'9', tab);
        af(tab);
        Terminal.ecrireStringln("resultat = "+ eval(tab));
    }
}
```

Nota bene : La méthode `toDigit` permet d'obtenir la valeur numérique d'un caractère correspondant à un chiffre. Par exemple, l'appel `toDigit('8')` renvoie en résultat l'entier 8.

Affichages (1 point par ligne correcte) :

```
false false true
 true false true
resultat = 5
```

Exercice 3 : séquences de caractères (4 points)

La méthode suivante permet de tirer au sort un caractère compris entre 'A' et 'E', c'est-à-dire un caractère représentant l'une des 5 premières lettres de l'alphabet en majuscule.

```
    public static char auSort(){
        return (char) (65+Math.random()*5);
    }
```

On veut écrire un programme (méthode main) qui tire au sort 50 lettres au moyen de cette méthode et affiche un résumé de cette séquence où chaque lettre est suivie du nombre d'occurrences successives de cette lettre dans la séquence. Une exception : s'il n'y a qu'une occurrence, seule la lettre est affichée. Par exemple, pour la séquence AABAAACC, c'est A2BA3C2 qui est affiché. Note : ce programme ne nécessite pas l'usage d'un tableau.

Solution :

```
public static void main(String[] args){
   char premier = auSort();
   int nbe = 1;
   String seq = "";
   for (int i=1; i<50; i++) {
      char c = auSort();
      if (c == premier) {nbe++;
      } else {
          seq = seq + premier;
          if (nbe > 1) {seq = seq + nbe;}
          premier = c;
          nbe = 1;
       }
    }
    Terminal.ecrireStringln(seq);
}
```

Exercice 4 : tic tac toe (6,5 points)

Le tic tac toe est un jeu qui se joue sur un damier de 9 cases. Un joueur marque des croix dans les cases, l'autre des ronds. Le premier joueur qui aligne trois de ses marques (croix ou rond) horizontalement, verticalement ou en diagonale a gagné. Initialement, aucune case n'est marquée. Chaque joueur pose une marque a son tour jusqu'à ce qu'un des deux ait gagné ou qu'il n'y ait plus de case vide.

Etat initial

Milieu de partie

O		X
		O
X		

Partie terminée

O		X
	X	
X		O

Partie nulle

O	O	X
X	X	O
O	X	O

Pour les correcteurs : le barême est donné à titre indicatif. Si certains tests ou appels demandés sont réalisés correctement par des méthodes auxiliaires plutôt que dans le main, donner les points quand même.

Ecrire un programme avec les caractéristiques suivantes.

1. (**1 pt**) Ecrire les déclarations et instructions permettant de représenter la situtation initiale (dans la méthode main).

```
public static void main(String args[]) {
    int [][] grille = new int [3][3]; // Le plateau de jeu vide
    int quiJoue = 1; // Le joueur qui doit jouer
    boolean termine = false;
    boolean bloque = false;
    int gagnant = 0;
    Terminal.ecrireStringln("Plateau Initial");
    affichePlateau(grille);
    Terminal.ecrireStringln("Le joueur x commence:");
   // Suite du main en question 4
```

2. (**2 pts**) Ecrire une méthode qui affiche un état du jeu.

Notation : dans cette solution, la méthode affiche le plateau, le gagnant ou si la partie est bloquée, et le prochain à joueur sinon. 1 pt pour l'affichage du plateau, 1 pt pour indication correcte de l'issue du jeu et/ou le blocage.

```
static void affichePlateau(int [][] p){
        final char [] symbole = {' ','x', 'o'};
        Terminal.ecrireString("\n   0  1  2");
        Terminal.ecrireString("\n  ------------\n");
        for(int i = 0;i<p.length;i++){
            Terminal.ecrireInt(i);
            Terminal.ecrireString("|");
            for(int j = 0;j<p[i].length;j++){
                Terminal.ecrireChar(symbole[p[i][j]]);
                Terminal.ecrireString(" | ");
            }
            Terminal.ecrireString("\n  ------------\n");
        }
        Terminal.sautDeLigne();
```

```
    }

  /* Teste si un plateau n'a plus de cases vides  */
  static boolean plateauBloque(int [][] p){
        for(int i = 0;i<p.length;i++){
            for(int j = 0;j<p[i].length;j++){
                if (p[i][j]==0) return false;
            }
        }
        return true;
  }
 static void afficheEtat(int [][] p, boolean ter, boolean blocus,int g, int next){
        final char [] symbole = {' ','x', 'o'};
        Terminal.ecrireStringln("La nouvelle grille:");
        affichePlateau(p);
        if (ter) {
          Terminal.ecrireStringln("Le joueur "+symbole[g]+" gagne!");
        } else if (blocus) {
          Terminal.ecrireStringln("Jeu bloque. Fin sans gagnant");
        } else {
          Terminal.ecrireStringln("Le tour au joueur "+symbole[next]);
        }
 }
```

3. (**0.5 pts**) Ecrire une méthode qui enregistre dans la mémoire un coup joué par un des joueurs. Toutes les données nécessaires, notamment le damier, la position du coup et le type de la marque, seront passés en argument à la méthode. Celle-ci ne doit faire ni affichage ni lecture au clavier.

```
   static void jouer(int joueur,int x, int y, int[][] p) {
        p[x][y]=joueur;
    }
```

4. (**1 pt**) Ecrire une méthode qui teste si le damier contient trois marques alignées.

```
  /* Teste si le joueur j a gagne  */
  static boolean victoireDe(int [][] p,int j){
      return (
              // lignes
              (p[0][0]==j && p[0][1]==j &&  p[0][2]==j) ||
              (p[1][0]==j && p[1][1]==j &&  p[1][2]==j) ||
              (p[2][0]==j && p[2][1]==j &&  p[2][2]==j) ||
              // colonnes
              (p[0][0]==j && p[1][0]==j &&  p[2][0]==j) ||
              (p[0][1]==j && p[1][1]==j &&  p[2][1]==j) ||
              (p[0][2]==j && p[1][2]==j &&  p[2][2]==j) ||
              // diagonales
              (p[0][0]==j && p[1][1]==j &&  p[2][2]==j) ||
              (p[0][2]==j && p[1][1]==j &&  p[2][0]==j) );
  }
```

5. (**2 pts**) Ecrivez dans la méthode main un appel à chacune des quatre méthodes écrites précédemment.

Solution sans la boucle de jeu (acceptable). La méthode lireCoup n'est pas demandée :

```
public static void main(String args[]) {
   int [][] grille = new int [3][3]; // Le plateau de jeu vide
   int quiJoue = 1; // Le joueur qui doit jouer
   boolean termine = false;
   boolean bloque = false;
   int gagnant = 0;
   Terminal.ecrireStringln("Plateau Initial");
   affichePlateau(grille);
   Terminal.ecrireStringln("Le joueur x commence:");
   int [] coord = lireCoup(grille);
   jouer(quiJoue,coord[0],coord[1], grille);
   if (victoireDe(grille,quiJoue)){ // teste si victoire
            gagnant = quiJoue;
            termine = true;
   } else if (plateauBloque(grille)) {// teste si on peut encore joueur
         bloque = true;
   } else {
           if (quiJoue == 1)
                   quiJoue = 2;
           else    quiJoue = 1;
   }
  afficheEtatJeu(grille,termine,bloque,gagnant,quiJoue);
}
```

Solution complète :

```
    public static void main(String args[]) {
        int [][] grille = new int [3][3]; // Le plateau de jeu vide
        int quiJoue = 1; // Le joueur qui doit jouer
        boolean termine = false;
        boolean bloque = false;
        int gagnant = 0;
        Terminal.ecrireStringln("Plateau Initial");
        affichePlateau(grille);
        Terminal.ecrireStringln("Le joueur x commence:");
        while(!termine && !bloque){
           int [] coord = lireCoup(grille);
           jouer(quiJoue,coord[0],coord[1], grille);
           if (victoireDe(grille,quiJoue)){ // teste si victoire
                gagnant = quiJoue;
                termine = true;
            } else if (plateauBloque(grille)) {// teste si on peut encore joueur
              bloque = true;
            } else {  // Le tour a l'autre joueur
                if (quiJoue == 1) quiJoue = 2;
                else    quiJoue = 1;
            }
           afficheEtatJeu(grille,termine,bloque,gagnant,quiJoue);
```

```
        }
        Terminal.ecrireStringln("\n Au revoir et a bientot ... \n");
    }

/* Deux méthodes auxiliares non denmandees
 */
  static boolean coordValides(int x, int y, int [][] p){
     return(x>= 0 && x<= p.length-1 && y>= 0 && y<= p.length-1
            && p[x][y] == 0);
  }

   /* Lecture d'un coup valide (non demandé) */
    static int [] lireCoup(int [][] plateau) {
       boolean coupValide = false;
       int [] coup =new int [2];
       while(!coupValide){
             Terminal.ecrireString("Coordonnee x= ");
             int x = Terminal.lireInt();
             Terminal.ecrireString("Coordonnes y= ");
             int y = Terminal.lireInt();
             if (!coordValides(x,y,plateau)){  // coordonnees invalides
                Terminal.ecrireStringln("Coordonnees invalides , recommencez");
             } else {
               coup[0] = x;
               coup[1] = y;
               coupValide = true;
             }
       }
       return coup;
    }
```