20/20

NOM : ..... PRENOM : ..... .

Test IP10 n°1 - Jeudi 02 mars 2006

Question 1 : 2 points 2/2

Supposons qu'il existe un fichier MonFichier.java dans mon répertoire courant. Corrigez les commandes suivantes dans le terminal et expliquez :

javac
java -c MonFichier.java : on compile des fichiers .java oui 1/1

java MonFichier.class : on exécute le programme dont le nom n'a plus d'extension
oui 1/1

Question 2 : Corrigez le programme suivant (5 types d'erreurs). Attention ! Si vous corrigez des choses justes, alors la notation est négative. Utilisez un stylo de couleur. **5 points** 5/5

NB: Le texte en gras ne contient pas d'erreur.

```
class
Class MaClasse {                                   oui 1

    public static void main (String argv [ ]) {

        double r ;                                 oui 1
        r = Lire . d();
        if (r <= 0) {                              1
System. out .println(calcul impossible) ;          2
        }else{
System. out. println(surface : r * r * 3,14);
        }

    }

}
```

Question 3 : Calcul de surface de figure géométrique - 13 points 13/13

L'application que vous devez programmer doit calculer la surface de trois types de figures géométriques :

- le carré. La surface d'un carré est c*c où c est la mesure du côté du carré ;
- le disque. La surface d'un disque est r*r*pi où r est la mesure du rayon du disque et pi vaut 3.14 ;
- le triangle équilatéral (trois côté égaux). La surface d'un triangle équilatéral est 0.433*c*c où c est la mesure d'un côté du triangle.

Le programme doit :

- déclarer les variables `cote` (de type réel) et `typeFigure` (de type caractère). Vous pouvez bien entendu déclarer les autres variables dont vous auriez besoin ;
- lire la valeur de la mesure de référence dans la variable `cote` (le rayon du disque peut aussi être lu dans cette variable);
- lire le type de figure dans la variable `typeFigure`. Les valeurs de typeFigure possibles sont 'c' pour un carré, 'd' pour un disque et 't' pour un triangle équilatéral ;
- En fonction du type de figure :
  1. afficher le nom de la figure en toutes lettres ("carré", "disque", "triangle équilatéral") ;
  2. calculer la valeur de la surface et afficher la valeur du résultat précédé du message "Surface de la figure :";

Hypothèse : nous supposons que l'utilisateur du programme ne saisit que des valeurs correctes.

Certaines lignes n'ont pas été écrites dans le programme suivant. Il s'agit de l'implémentation de chacun des commentaires.
Complétez ce programme.

```
class MaClasse2 {

    public static void main (String argv [ ]){

        // déclarer les variables
double cote, surface;
char typefigure;                                   | 1,5

        // lire les valeurs de cote et typeFigure
System.out.println ("Entrez le type de figure : c pour un carré, d pour un disque,
t pour un triangle équilatéral");
typefigure = Lire.c();
System.out.println ("Entrez la valeur de la cote");
cote = Lire.d();                                   | 3
        // tester le type de figure et afficher son nom en toutes lettres
        // calculer la surface  puis afficher le résultat
switch (typefigure){
   case ('c') : System.out.println ("c'est un carré.");
                surface = cote * cote;
                break;
   case ('d') : System.out.println ("c'est un disque");
                surface = cote * cote * pi;
                break;
   case ('t') : System.out.println ("c'est un triangle équilatéral");
                surface = 0,433 * cote * cote;
                break;
        }
System.out.println ("Surface de la figure = " + surface);
    }
}
```

oui 8.5

NOM : ..... PRENOM : ..: .....
GROUPE : ..U3.

Test IP10 n°2 - Lundi 31 mars 2003 - après-midi

Question 1 (5 points)
Donner le résultat de l'affichage du programme suivant :

```
class Mystere {
   public static void main (String[] arg){

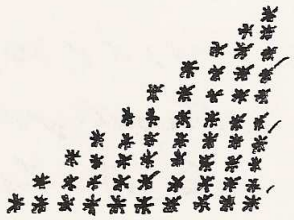
      int i, j ;
      for (i = 1 ; i <= 10 ; i++) {
         for (j =1; j <= 10 - i ; j++){

            System.out.print(" ");

         }
         for (j = 1; j <= i ; j++) {
            System.out.print("*");
         }
         System.out.println();
      }

   }
}
```

5/5



Question 2 (1 point) : Vrai ou faux ?

Un tableau peut etre agrandi ou retréci après sa définition. FAUX 1/1

Question 3 (1 point)
Sachant que la variable unTableau est un tableau d'entier, quelle(s) est(sont) le(s) instruction(s) correcte(s) ? (entourer les bonnes propositions)

- unTableau.length = 3 :
- unTableau[unTableau.length] = 3 ;
- unTableau[0] = 3 ;

1/1

Question 4 (13 points) : Notes de partiels

Le programme suivant doit :

- définir un tableau de 30 entiers nommé `notes` ;
- lire le tableau (l'utilisateur entre des notes d'élèves comprises entre 0 et 20, on supposera qu'elles sont entrées par ordre croissant) ;
- calculer la moyenne des notes comprises dans le tableau
- afficher toutes les notes strictement inférieures à la moyenne

```
class MaClasse2 {

    public static void main (String [] arg){

        // déclarer les variables -tableau compris - dont vous
        aurez besoin,
int [] notes = new int [30];
int i;
double moyenne = 0;

        // Lire le tableau de 30 notes, en calculant au passage la
        moyenne
for( i=0; i<30; i++) {
   System.out.println("Entrez la "+i+1+"ème note");
   notes[i] =Lire.i();
   moyenne = moyenne + notes[i] }

moyenne = moyenne/30;
System.out.print(" la moyenne est de "+moyenne);
        }
        // Afficher les notes inférieures à la moyenne
for( i=0; i<30; i++) {
   if ( notes[i]<moyenne){
      System.out.println("la note"+ notes[i]+"est inférieur à la
                                                    moyenne");

   }
}
```

3/3

5/5

4/5

```
SO.Pl("les notes < à la moyenne sont: ");
i = 0;
while (notes[i] < moyenne) {
    SOPrint(notes[i] + ", ");
    i = i + 1;
}
}
```

20/20

NOM : ..... PRENOM : .....

Test IP10 n°2 - Jeudi 16 mars 2006

Sans document

oui 5/5

Question 1 : 5 points

Donnez (dessinez) le résultat de l'affichage du programme suivant.

NB : L'opération *print* ne fait pas de retour à la ligne, à l'inverse de *println*.

```
class Mystere {

  public static void main (String arg []) {
        int i,j ;

        for (i = 1; i <= 5; i=i+1) {
            for (j=1; j<=5; j = j+1){
                if ((i%2 == 0) && (j%2==0)){
                    System.out.print(" ");
                } else {
                    System.out.print("*");
                }
            }
            System.out.println();
        }
    }
}
```

	j=1	j=2	j=3	j=4	j=5
i=1	*	*	*	*	*
i=2	*	␣	*	␣	*
i=3	*	*	*	*	*
i=4	*	␣	*	␣	*
i=5	*	*	*	*	*

(Ce qui s'affiche réellement est en bleu)

oui 1/1

Question 2 : 1 point

Quelle(s) instruction(s) permet(tent) d'accéder au dernier élément d'un tableau de 3 éléments nommé unTableau ? (entourer la(les) bonne(s) proposition(s))

1. unTableau[unTableau.length]
2. unTableau[unTableau.length-1]
3. unTableau[3]
4. unTableau[2]

14/14

Question 3 : Jeu de dé - 14 points

Un joueur de dé vous demande de programmer une application permettant de jouer avec 5 dés. Un dé possède 6 faces, les valeurs allant de 1 à 6. Le joueur lance les 5 dés et renseigne les valeurs du tirage de dé, **par ordre croissant**, dans un tableau d'entiers. Le programme doit alors calculer la somme des valeurs et indiquer s'il y a un Flush (5 valeurs identiques).

Le programme doit suivre les étapes suivantes :

1) créer un tableau d'entiers nommé `tableau5D` de taille 5;
2) Saisir le tableau case par case. Ne pas oublier un message pour l'utilisateur ;
3) Calculer la somme des valeurs des différents dés et afficher cette somme ;
4) Tester s'il s'agit d'un flush et afficher "Flush" si les 5 dés ont la même valeur ou "Pas de Flush" sinon ;

Hypothèse : nous supposons que l'utilisateur du programme ne saisit que des valeurs correctes.

Certaines lignes n'ont pas été écrites dans le programme suivant.
Complétez ce programme en suivant les indications données en commentaires.

```
class JeuDe {

    public static void main (String[ ] argv){

        // Créer un tableau d'entiers de taille 5
int [] tableauSD = new int [5];

        // Déclarer toutes les autres variables nécessaires, et les initialiser si besoin
int i;
int somme = 0;

        // Saisir les valeurs du tableau (supposées dans l'ordre croissant)
for (i=0; i<tableauSD.length-1; i=i+1) {
System.out.println("Entrez la valeur du "+i+"ème tirage");
tableauSD[i] = Lire.i();
}

        // Calculer la somme des valeurs et l'afficher
for (i=0; i<tableauSD.length-1; i=i+1){
somme = somme + tableauSD[i];
}
System.out.println("La somme des 5 dés est : "+somme);

        // Tester si les valeurs forment un Flush et afficher le résultat
if ((tableauSD[0]==tableauSD[1]) && (tableauSD[1]==tableauSD[2])
&& (tableauSD[2]==tableauSD[3]) && (tableauSD[3]==tableauSD[4]))
{
System.out.println("les 5 dés forment un flush");
}
else
{System.out.println("les 5 dés ne forment pas de flush");}

    }

}
```

déclaration : 1,5
initialisation somme : 1
oui 2
1,5
oui 1
oui 1
oui 1
oui bon 4

Contrôle des Connaissances

Introduction à la Programmation
Date : 08/04/2005
Coordonnatrices :

S. Chabridon et C. Lecocq

Module : IP10
Durée : 1h30
Documents autorisés : polycopiés distribués en cours et notes personnelles uniquement

---

Avertissements

1/ Lisez attentivement le sujet

2/ Soyez clairs et précis dans vos réponses

3/ Soignez la présentation

4/ Respectez les noms de variables et de paramètres demandés

5/ Utilisez la classe *Lire* si nécessaire

6/ Conseil : Complétez le programme principal (`main`) même si vous n'avez pas terminé l'écriture des méthodes

7/ Barème indicatif

Question 1 : 2 points.

Question 2 : 3 points.

Question 3 : 4 points.

Question 4 : 3 points.

Question 5 : 3 points.

Question 6 : 3 points.

Question 7 : 2 points.

Un laboratoire de recherche en génétique vous demande de programmer une application permettant de déterminer les caractéristiques génétiques d'un enfant. Dans le processus de reproduction, chaque parent donne un « génotype ». L'association du génotype de la mère avec celui du père détermine le génotype de l'enfant. Un génotype est modélisé à l'aide d'un tableau de taille 5 correspondant aux cinq gènes à considérer. Il n'existe que deux valeurs possibles pour chaque gène, codées à l'aide d'un caractère : 'D' pour dominant et 'R' pour récessif.

La liste des cinq gènes à considérer est la suivante :

0 : sexe	Dominant (Y) : garçon	Récessif (X) : fille
1 : couleur des yeux	Dominant : yeux noirs	Récessif : yeux bleus
2 : couleur des cheveux	Dominant : cheveux bruns	Récessif : cheveux blonds
3 : taille	Dominant : grande taille	Récessif : petite taille
4 : hémophilie	Dominant : non hémophile	Récessif : hémophile

Exemple :

Supposons qu'un père « donne » le génotype suivant :

R	D	R	D	D

Ce tableau se lit de la façon suivante :

- indice 0, caractère récessif c'est-à-dire le caractère « fille » (X) ;
- indice 1, caractère dominant c'est-à-dire le caractère « yeux noirs » ;
- indice 2, caractère récessif c'est-à-dire le caractère « cheveux blonds » ;
- indice 3, caractère dominant c'est-à-dire le caractère « grande taille » ;
- indice 4, caractère dominant c'est-à-dire le caractère « non hémophile ».

Supposons qu'une mère « donne » le génotype suivant :

R	D	D	R	D

Ce tableau se lit donc : « fille », « yeux noirs », « cheveux bruns », « petite taille », « non hémophile ».

Le caractère « apparent » de l'enfant est déterminé par association des deux génotypes. Les gènes dominants sont apparents lorsqu'au moins l'un des deux parents donne un gène dominant. Les gènes récessifs ne sont apparents que si les gènes des deux parents sont récessifs.

Dans notre exemple, le génotype apparent de l'enfant sera :

R	D	D	D	D

L'enfant sera donc une « fille », « yeux noirs », « cheveux bruns », « grande taille », « non hémophile ».

Questions

Question 1 : définition des structures de données

Le génotype de chacun des membres de la famille est mémorisé dans un tableau de caractères. Il y aura donc trois tableaux nommés `mere`, `pere`, `enfant` de cinq caractères chacun. Définir ces trois tableaux et les autres variables que vous jugerez nécessaires. Pour cela, vous devez compléter le programme principal (`main`) fourni en annexe au sujet.

Question 2 : saisir un gène

Écrire une méthode permettant de saisir (lire) un gène valide. Les deux seules valeurs possibles de gène sont 'D' pour dominant et 'R' pour récessif. Cette méthode doit demander à l'utilisateur de saisir une valeur de gène valide. Tant que l'utilisateur ne propose pas de valeur de gène valide, cette méthode doit redemander une nouvelle valeur. Lorsque l'utilisateur a proposé une valeur de gène valide, cette méthode doit la retourner. Le prototype (l'entête apparaissant sur la 1ère ligne) de la méthode est le suivant :

```
static char saisirGene()
```

Pour chacune des questions suivantes, écrire la méthode correspondante sur votre copie puis compléter le programme principal (`main`) pour appeler cette méthode.

Question 3 : saisir le génotype d'un individu

1. Écrire une méthode permettant de saisir (lire), gène par gène, la liste des gènes constituant le génotype d'un individu. Pour saisir la valeur d'un gène, vous devez appeler la méthode `saisirGene()` définie à la question 2. Le génotype (tableau) doit être rempli entièrement (il contiendra toujours 5 gènes renseignés). Le prototype (l'entête) de la méthode est le suivant :

```
static void saisirGenotype(char[] tabGenes)
```

2. Compléter le programme principal (`main`) afin d'appeler la méthode que vous venez de définir pour saisir les deux tableaux correspondant aux génotypes de la mère et du père définis à la question 1.

Question 4 : afficher le génotype d'un individu

1. Écrire une méthode permettant d'afficher (écrire) le génotype d'un individu. La liste des gènes constituant le génotype d'un individu doit être affichée sur une seule ligne précédée de « Génotype de l'individu : ». L'affichage doit se terminer par un retour à la ligne. Le prototype (l'entête) de la méthode est le suivant :

```
static void afficherGenotype(char[] tabGenes)
```

2. Compléter le programme principal (`main`) afin d'appeler la méthode que vous venez de définir pour afficher les deux tableaux correspondant aux génotypes de la mère et du père définis à la question 1.

Conseil : vous supposerez qu'il existe une méthode permettant d'obtenir la légende associée à un indice du tableau de gène et une valeur de gène. Le prototype de cette méthode est :

```
static String afficherLegende(int indiceCaractere, char genre)
```

Exemple d'appel de cette méthode :

```
String legende ;
Legende = afficherLegende(0, 'R') ;
System.out.println(legende) ;
```

Cette dernière instruction affiche « fille ». Dans notre cas, la valeur 'R' est contenue dans une case d'un tableau décrivant un génotype; 0 est l'indice de cette case.

Question 5 : Fréquence d'un gène

1. Écrire une méthode permettant de calculer la fréquence d'une valeur ('D' pour Dominant ou 'R' pour Récessif) de gène dans un génotype. Il s'agit, par exemple, de calculer le nombre de 'D' ou le nombre de 'R' présents dans le tableau listant les gènes d'un génotype. Cette méthode reçoit en paramètres un tableau contenant le génotype et un caractère contenant la valeur de gène dont il faut calculer la fréquence. Cette méthode renvoie la fréquence calculée après avoir parcouru tout le tableau de gènes.

Le prototype (l'entête) de la méthode est le suivant :

```
static int calculerFrequenceGene(char[] tabGenes, char geneCherche)
```

2. Compléter le programme principal (`main`) afin d'appeler la méthode que vous venez de définir pour calculer le nombre de gènes dominants ('D') dans le génotype du `pere` (tableau défini à la question 1).

Question 6 : Père ou mère dominant

1. Écrire une méthode permettant de déterminer quel est le parent dominant. Cette méthode doit faire appel à la méthode `calculerFrequenceGene()` une première fois pour le génotype du père en demandant la fréquence du gène 'D' et une deuxième fois pour le génotype de la mère en demandant la fréquence du gène 'D'. Cette méthode doit ensuite comparer les deux résultats et afficher :

6. « Père dominant » si la fréquence du gène 'D' chez le père est supérieure à la fréquence du gène 'D' chez la mère ;
7. « Père et mère équivalents » si la fréquence du gène 'D' chez le père est égale à la fréquence du gène 'D' chez la mère ;
8. « Mère dominante » si la fréquence du gène 'D' chez le père est inférieure à la fréquence du gène 'D' chez la mère.

Le prototype (l'entête) de la méthode est le suivant :

```
static void testerDominant (char[] tabGenesPere, char[] tabGenesMere)
```

2. Compléter le programme principal (`main`) afin de d'appeler la méthode que vous venez de définir pour tester quel est le parent dominant pour les tableaux `pere` et `mere` définis à la question 1.

Question 7 : génotype de l'enfant

1. Écrire une méthode permettant de calculer le génotype de l'enfant. Pour chaque caractère (indice du tableau décrivant le génotype), le gène de l'enfant vaut :

9. 'D' si l'un ou l'autre des deux parents a un gène 'D' pour ce caractère ;
10. 'R' si les deux parents ont un gène 'R' pour ce caractère.
11. Cette méthode reçoit en paramètres les trois tableaux de génotype à savoir dans l'ordre le père, la mère et l'enfant. Il détermine le génotype de l'enfant à partir des deux génotypes parentaux.

Le prototype (l'entête) de la méthode est le suivant :

```
static void genEnfant (char[] tabGenesPere, char[] tabGenesMere,
char[] tabGenesEnfant)
```

2. Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour déterminer le génotype de l'enfant à partir des génotypes parentaux saisis à la question 3, puis afficher le génotype de l'enfant.

Introduction à la Programmation
Date : 06/04/2006
Coordonnatrices :

S. Chabridon et C. Lecocq

Module : IP10
Durée : 1h30
Documents autorisés : ceux distribués en cours et notes personnelles

---

Avertissements

1/ Lisez attentivement le sujet.

2/ Soyez clairs et rigoureux dans vos réponses.

3/ Soignez la présentation.

3/ Respectez les noms de variables et de paramètres demandés.

4/ Barème indicatif

Question 1 : 1 point.
Question 2 : 3 points.
Question 3 : 3,5 points.
Question 4 : 3,5 points.
Question 5 : 3 points.
Question 6 : 3 points.
Question 7 : 3 points.

Des agents secrets vous demandent de programmer une application permettant de décrypter des codes secrets.

Questions

Question 1 : définition de la structure de données

Le code secret est mémorisé dans un tableau de 200 caractères, nommé `codeSecret`. Définir ce tableau. Pour cela vous devez compléter le programme principal (main) fourni en annexe au sujet.

Pour chacune des questions suivantes, écrire la méthode correspondante sur votre copie puis compléter le programme principal (main) permettant d'appeler cette méthode.

Question 2 : saisir le code secret

1. Écrire une méthode permettant de saisir (lire), caractère par caractère, l'ensemble des 200 caractères du code secret dans le tableau défini à la question 1. Vous devez afficher un message avant chaque saisie de caractère indiquant le numéro du caractère à saisir (« Veuillez saisir le caractère no » suivi du numéro par exemple). Vous n'avez pas à vérifier la validité du caractère saisi. De plus, si l'utilisateur ne tape aucun caractère au clavier mais appuie directement sur la touche entrée, la case restera vide (caractère '\n' ou caractère vide ''). Le tableau peut contenir des cases vides au début, suivies du code secret et éventuellement d'autres cases vides à la fin. Ci-dessous, un exemple de code secret sur 7 cases.

		c	o	d	e	

Le prototype (l'entête) de la méthode est le suivant :

static void saisirCode(char[] tableauCode)

2. Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour saisir le tableau défini à la question 1.

Question 3 : afficher le code secret

1. Écrire une méthode permettant d'afficher à l'écran le code secret. L'affichage du code secret est précédé du message « code secret : », directement (sur la même ligne) suivi des caractères du code secret. Les caractères vides ne doivent pas être affichés. Un saut de ligne est fait après l'affichage du code secret. Cette méthode reçoit en paramètre un tableau de caractères contenant le code secret.

Le prototype (l'entête) de la méthode est le suivant :

static void afficherCode(char[] tableauCode)

2. Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour afficher le tableau saisi à la question 2.

Question 4 : échanger deux caractères

1. Écrire une méthode permettant d'échanger deux caractères dans le tableau. Cette méthode reçoit en paramètre le tableau et les positions (indices) des deux caractères à échanger. Le prototype (l'entête) de la méthode est le suivant :

   static void echanger (char[] tableauCode, int position1, int position2)

2. Compléter le programme principal (main) afin de :

   1. demander à l'utilisateur d'indiquer les positions de deux caractères **non vides** à échanger. Tant que la position donnée correspond à un caractère vide, il faudra en demander une autre;

   2. appeler cette méthode pour échanger les caractères (dont les positions viennent d'être données) dans le tableau saisi à la question 2.

   3. afficher le résultat en appelant la méthode de la question 3.

Question 5 : échanger les 'a' et 'e' avec le caractère suivant

1. Écrire une méthode permettant d'échanger tous les 'a' et 'e' d'un code secret avec le caractère le suivant directement. Attention, lorsque vous avez échangé un 'a' ou un 'e' avec le contenu de la case suivante, il faut alors continuer le traitement après les deux cases échangées. Cette méthode doit faire appel à la méthode définie à la question 4.

Considérons par exemple un code secret simplifié sur 6 cases :

a	b	t	a	c	u

La première case (indice 0) contient un 'a'. Elle doit donc être échangée avec la deuxième (indice 1). Le traitement continue à partir de la troisième case (indice 2).

La quatrième case contient un 'a'. Elle doit donc être échangée avec la cinquième. Le traitement continue et se termine à la sixième case.

Le résultat des échanges est donc :

b	a	t	e	a	u

Le prototype (l'entête) de la méthode est le suivant :

static void echangerAEtE (char[] tableauCode)

2. Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour décrypter, à l'aide de cette méthode, le tableau saisi à la question 2, et d'afficher le résultat.

Question 6 : 'a' ou 'e' le plus fréquent

1. Écrire une méthode permettant de déterminer quelle est la voyelle 'a' ou 'e' la plus fréquente dans un tableau de caractères. Cette méthode doit retourner le caractère le plus fréquent comme résultat. Si les fréquences sont équivalentes, la valeur 'a' est retournée.

Le prototype (l'entête) de la méthode est le suivant :

static char plusFrequentAouE (char[] tableauCode)

Question 7 : décryptage avec échange des 'a' et 'e' avec le caractère suivant et élimination de la voyelle la plus fréquente

1. Écrire une méthode, recevant en paramètre un tableau de caractères, permettant de :
   - appeler la méthode qui échange tous les 'a' et 'e' avec le caractère qui le suit (question 5) dans le tableau passé en paramètre ;
   - appeler la méthode permettant de déterminer le caractère le plus fréquent entre le 'a' et le 'e' (question 6) dans le tableau passé en paramètre ;
   - afficher le code, après échange, en ignorant le caractère le plus fréquent dans le tableau passé en paramètre.

   Considérons un code secret simplifié sur 7 cases :

a	j	e	a	v	e	e

   Après application de la méthode d'échange, le résultat temporaire est :

j	a	e	v	a	e	e

   Le 'e' est plus fréquent que le 'a'. Le 'e' n'apparaît donc pas dans l'affichage final, qui donne :

   java

   Le prototype (l'entête) de la méthode est le suivant :

   static void decrypter (char[] tableauCode)

2. Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour le code secret saisi à la question 2.

Corrigé contrôle final 1 IP 10 2006

```
class CodeSecret {
   public static void main (String arg[]){

                // Question 1
                char [] codeSecret = new char[10] ;

                // Question 2
                saisirCode(codeSecret);

                // Question 3
                afficherCode(codeSecret);



                //Question 4
                int pos1, pos2;
                do {
                        System.out.println("Position du 1er caractere a
echanger (non vide) ?");
                        pos1 = Lire.i();
                }
                while (codeSecret[pos1] == '\n');

                do {
                        System.out.println("Position du 2eme caractere a
echanger (non vide)?");
                        pos2 = Lire.i();
                }
                while (codeSecret[pos2] == '\n');

                echanger(codeSecret, pos1, pos2);

                System.out.println("resultat apres echange 2 positions");
                afficherCode(codeSecret);



                //Question 5
                echangerAEtE(codeSecret);
                System.out.println("resultat apres echange a,e - car
suivant");
                afficherCode(codeSecret);

                // Question7
                decrypter(codeSecret);
   }
   // Question 2
   static void saisirCode(char[] tableauCode) {
        int i ;
        for (i=0; i < tableauCode.length ; i=i+1) {
                System.out.println("caractere : "+ i + " ?");
                tableauCode[i]=Lire.c();
        }
   }
   // Question 3
   static void afficherCode(char[] tableauCode){
        int i;

        System.out.print("Code secret : ");
```

```
     for (i=0; i < tableauCode.length; i=i+1) {
             if (tableauCode[i] != '\n') {
                     System.out.print(tableauCode[i]);
             }
     }
     System.out.println();

}

// Question 4
static void echanger (char[] tableauCode, int position1, int position2){

     char temp ;

     temp = tableauCode[position1];
     tableauCode[position1]=tableauCode[position2];
     tableauCode[position2]=temp;
}
// Question 5
static void echangerAEtE (char[] tableauCode) {
     int i = 0 ;
             while (i < tableauCode.length -1) {
                     if (tableauCode[i] == 'a' || tableauCode[i] == 'e'){
                             echanger(tableauCode, i, i+1);
                             i = i+1;
                     }
                     i=i+1;
             }
}

// Question 6
static char plusFrequentAouE (char[] tableauCode){
        int freqA = 0, freqE=0;
        int i;
        for (i=0; i < tableauCode.length; i = i+1){
                if (tableauCode[i] == 'a') {
                        freqA = freqA + 1 ;
                }
                if (tableauCode[i] == 'e') {
                   freqE = freqE + 1 ;
                }
        }
        if (freqA >= freqE) {
            return 'a';
        } else {
            return 'e';
        }
}

//Question7
static void decrypter (char[] tableauCode){

     int i;
     char carPlusFrequent;

     echangerAEtE(tableauCode);

     carPlusFrequent = plusFrequentAouE(tableauCode);

     System.out.println("code apres decryptage complet");
```

```
        for (i=0; i < tableauCode.length; i=i+1){
                if (tableauCode[i] != carPlusFrequent){
                        System.out.print(tableauCode[i]);
                }
        }
        System.out.println();
        }
}
```

Exemple CC

NOM : ..... PRENOM : ..... GROUPE : .....

Test IP10 n°1 - Mercredi 10 mars 2004 - Matin

Question 1 : 2 points
J'essaie de compiler un programme java, écrit dans le fichier monFichier.java. La compilation génère des erreurs. Ces erreurs sont elles sauvegardées dans un fichier ? Si oui, lequel ? Si non, où sont elles ?

Non. Dans le terminal où a été lancée la compilation

Question 2 : Corrigez le programme suivant (5 types d'erreurs). Attention ! Si vous corrigez des choses justes, alors la notation est négative. Utilisez un stylo de couleur. **5 points**

```
class MaClasse {

    public static void main (String [ ] arg) {
        real unReel = 3,0 ;

        system.in.println("valeur de unReel ?") ;

    }

}
```

Question 3 : calcul de la moyenne d'un module - 14 points
Le programme suivant doit calculer la moyenne d'un module. Pour cela il doit:

- lire la categorie de l'étudiant (un caractère) dans une variable nommée categorie. Cette variable vaut :
  - 'E' s'il s'agit d'un étudiant EM "classique" ;
  - 'R' pour un redoublant du module ;
  - 'A' pour un auditeur libre ;
- calculer la note d'UV selon les règles suivantes :
  - un étudiant EM a une note de controle continu (un réel qui devra être lu dans une variable noteCC) et une note de controle final (un réel qui devra être lu dans un e variable noteCF). La moyenne est la somme des deux notes divisée par deux;
  - un redoublant ne repasse que le contrôle final. Cette note constitue donc seule la moyenne ;
  - un auditeur libre a le choix d'être évalué ou non dans le module. Il faut donc lire le choix de l'étudiant. S'il choisit d'être évalué, seule la note de controle final compte, ce qui est équivalent à la situation d'un étudiant redoublant. S'il choisit de ne pas être évalué, la moyenne sera arbitrairement à 0.
- afficher la valeur de la categorie d'étudiant et sa moyenne.

Hypothèse : nous supposons que l'utilisateur du programme ne saisit que des valeurs correctes (il ne saisira pas de categorie d'etudiant différente de 'E', 'R', 'A' par exemple).

Certaines lignes n'ont pas été écrites dans le programme suivant. Il s'agit de l'implémentation de chacun des commentaires. Complétez ce programme.

```
class MaClasse2 {

    public static void main (String [] arg){

        // déclarer une variable pour la categorie de l'étudiant
        char categorie ;

        // declarer deux réels pour la  note de controle continu et de controle final
        double noteCC, noteCF ;

        // declarer une variable de type réel pour le calcul de la moyenne
        double moyenne ;

        // declarer une variable pour mémoriser le choix d'etre evalué (ou non) d'un
        auditeur libre

        char choix ;
        // Lire la valeur de la categorie de l'étudiant

        System.out.println("Categorie de l'etudiant ");
        categorie = Lire.c() ;

        //En fonction de la categorie de l'étudiant, lire les informations necessaires et
        calculer la moyenne.

        if (categorie == 'E') {

        // "if" imbriqués ou  switch

            System.out.println("Note de controle continu ?");
            noteCC = Lire.d();
            System.out.println("Note de controle final ?");
            noteCF = Lire.d();

            // calcul de la moyenne
            moyenne = (noteCC+ noteCF )/2 ;

        } else {
            if (categorie == 'A') {

                System.out.println( "Voulez-vous etre evalue (O/N) ?");
                choix= Lire.c();

                if (choix == 'O') {
                    System.out.println("Note de controle final ?");
                    moyenne = Lire.d() ;
                }else{
                    moyenne = 0 ;
                }
            } else { // élève redoublant
                System.out.println("Note de controle final ?");
                moyenne = Lire.d() ;
            }
        }
```

```
        // Afficher la valeur de la categorie d'etudiant et la moyenne calculee  sous la
        forme
        // eleve A, moyenne : 12,65
        // ou A est la valeur de la categorie de l'élève - ici A pour auditeur libre - et 12,65
        est la moyenne calculée

        System.out.println("eleve "+categorie+", moyenne : "+ moyenne);

    }

}
```

CF1
le 220307

Institut National des Télécommunications
Contrôle des Connaissances

Introduction à la Programmation
Date : 22/03/2007
Coordonnatrices :
S. Chabridon et C. Lecocq

Module : IP10
Durée : 1h30
Documents autorisés : ceux distribués en cours et notes personnelles

---

Avertissements

1/ Lisez attentivement le sujet.

2/ Soyez clairs et rigoureux dans vos réponses.

3/ Soignez la présentation.

3/ Respectez les noms de variables et de paramètres demandés.

4/ Barème indicatif

Question 1 : 1 point.
Question 2 : 2 points.
Question 3 : 3,5 points.
Question 4 : 3,5 points.
Question 5 : 4 points.
Question 6 : 6 points.

OpenINT vous demande de programmer une application simplifiée permettant de gérer les matchs de tennis. Un match de tennis oppose un participant à un adversaire. Pour simplifier le décompte des points, il est convenu que les points ne sont pas comptés avec les conventions classiques du tennis (15, 30, 40, jeu), mais avec les règles suivantes :

- lorsque le participant marque, il faut compter 1 point. Lorsque l'adversaire marque, il faut compter -1 point ; les points des deux joueurs sont additionnés au fur et à mesure de la partie ;
- il y a jeu dès que l'un des deux joueurs a marqué 4 fois (4 points pour le participants, -4 points pour l'adversaire).
- il y a match si l'un des deux joueurs a obtenu 6 jeux.

Les points des deux joueurs sont mémorisés au fur et à mesure de la partie dans le même tableau. Un exemple d'exécution du programme est proposé ci-après. Les séquences de points correspondant à un jeu de l'adversaire sont sur fond gris. Les séquences de points correspondant à un jeu du participant sont en gras.

Première exécution

Entrez la valeur du point : 1 si point du participant, -1 si point de l'adversaire, 0 pour fin de match

-1

(...) liste des interactions non montrée

Entrez la valeur du point : 1 si point du participant, -1 si point de l'adversaire, 0 pour fin de match

0

Points saisis pendant la partie [illegible] **/1/1/1/1 /** [illegible] **/1/1/1/1 /** [illegible]

Votre participant a gagné en points absolus : 26 à 24 points.

Votre participant a perdu 2 jeux à 6

Résultat différents en points absolus et en jeux

Deuxième exécution

Entrez la valeur du point : 1 si point du participant, -1 si point de l'adversaire, 0 pour fin de match

1

(...) liste des interactions non montrée

Entrez la valeur du point : 1 si point du participant, -1 si point de l'adversaire, 0 pour fin de match

0

Points saisis pendant la partie **:1/1/1/1 /1/1/1/1 /** [illegible] **/1/1/1/1 /1/1/1/1 /** [illegible] **/1/1/1/1 /1/1/1/1**

Votre participant a gagné en points absolus : 28 à 8 points.

Votre participant a gagné 6 jeux à 2

Match du participant

Questions

Question 1 : définition de la structure de données

Les points du match sont mémorisés dans un tableau de 100 entiers, nommé score. Définir ce tableau. Pour cela vous devez compléter le programme principal (main) de votre application.

Conseil : Prévoyez 2 parties sur votre copie. L'une pour le programme principal (main) et l'autre pour les méthodes des questions 2 à 6.

Pour chacune des questions suivantes :

1. **écrire la méthode (son entête et son corps, encore appelé contenu) correspondante sur votre copie puis**
2. **compléter le programme principal (main) permettant d'appeler cette méthode.**

Question 2 : remise à 0

1. Écrire une méthode permettant d'initialiser à 0 la totalité du tableau défini à la question 1. Le prototype (l'entête) de la méthode est le suivant :

   static void remiseAO (int[] tab)

2. Compléter le programme principal (main, fourni en annexe du sujet) afin d'appeler la méthode que vous venez de définir pour remettre à 0 (initialiser à 0) le tableau défini à la question 1.

Question 3 : saisir les points du match

1. Écrire une méthode permettant de saisir (lire), point par point, l'ensemble des points du match dans le tableau défini à la question 1. Vous devez afficher un message avant chaque saisie de point (« Entrez la valeur du point : 1 si point du participant, -1 si point de l'adversaire, 0 pour fin de match »). Vous n'avez pas à vérifier la validité du point saisi. Le tableau ne doit pas impérativement être rempli (certaines parties de tennis peuvent être courtes !); la saisie s'arrête dès que l'utilisateur entre la valeur 0. Il ne faut alors pas demander les valeurs du reste du tableau. Le prototype (l'entête) de la méthode est le suivant :

   static void lireTab (int[] tab)

2. Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour saisir le tableau défini à la question 1.

Question 4 : afficher les scores

1. Écrire une méthode permettant d'afficher à l'écran les points du match. L'affichage des points du match est précédé du message « Points saisis pendant la partie : », directement (sur la même ligne) suivi des points du match. Entre deux points, il faut afficher un « / »; il n'y a pas de saut de ligne avant la fin de l'affichage complet des points. Le tableau n'est pas affiché dans sa totalité : seule la partie contenant des points (-1 ou 1) doit être affichée. La partie restée à 0 n'est pas affichée. Cette méthode reçoit en paramètre un tableau d'entiers contenant le score (les points) du match et a pour prototype (entête) :

   static void afficherTab (int[] tab)

2. Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour afficher le tableau saisi à la question 3.

Question 5 : gain absolu

1. Écrire une méthode permettant de calculer qui est le vainqueur de la partie en points absolus. Pour cela il suffit de compter les points du participant (nombre de 1) et les points de l'adversaire (nombre de - 1) dans le tableau des scores. Le gagnant en points absolus est évidemment celui ayant totalisé le plus grand nombre de points. La méthode de nom « gain absolu » doit :

   - compter les points des deux joueurs ;
   - afficher un message du type :

     - « Votre participant a gagné en points absolus : 26 à 24 points. » si le participant a gagné. « 26 » et « 24 » sont bien entendu des exemples de valeurs ;

     - « Votre participant a perdu en points absolus : 10 à 24 points. » si le participant a perdu. « 10 » et « 24 » sont bien entendu des exemples de valeurs ;

   - retourner la valeur 1 à la méthode appelante si le participant a gagné, - 1 s'il a perdu.

   Le prototype (l'entête) de la méthode est le suivant :

   int gainAbsolu(int[] tab)

2. Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour déterminer quel est le vainqueur pour le tableau saisi à la question 3. Mémoriser le résultat retourné par la méthode que vous venez d'appeler dans la variable de votre choix.

Question 6 : gain jeux et match

1. Écrire une méthode permettant de déterminer quel joueur a gagné le match en nombre de jeux. Un joueur marque un jeu dès qu'il obtient 4 points cumulés (-4 pour l'adversaire). Il faut alors augmenter son nombre de jeux et remettre à 0 les points de chacun des joueurs. La méthode doit :

   - compter le nombre de jeux de chacun des participants ;
   - afficher un message du type :

     - « Votre participant a gagné 6 jeux à 5. » si le participant a gagné. « 6 » et « 5 » sont bien entendu des exemples de valeurs ;

     - « Votre participant a perdu 2 jeux à 6. » si le participant a perdu. « 2 » et « 6 » sont bien entendu des exemples de valeurs ;

   - retourner le nombre de jeux du gagnant; ce nombre est négatif si l'adversaire est le gagnant (la valeur retournée est alors -nombreDeJeuxAdversaire).

   Le prototype (l'entête) de la méthode est le suivant :

   int gainJeux(int[] tab)

2. Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour déterminer quel est le vainqueur en nombre de jeux pour le tableau saisi à la question 3. Mémoriser le résultat retourné par la méthode que vous venez d'appeler dans la variable de votre choix. A partir de cette valeur, vous devez :

   - indiquer s'il y a eu match. Il y a match si un des joueurs a gagné 6 jeux (afficher « Match du participant » si la valeur retournée est 6, ou bien« Match de l'adversaire » si la valeur retournée est - 6) ;
   - afficher le message « Résultats différents en points absolus et en jeux » si les résultats de match sont différents en comptant les points absolus ou en comptant les jeux (ce n'est pas le même joueur qui gagne en points absolus et en jeux). Ne rien afficher si les résultats sont cohérents.