

Titre :
(LgP6Linf.EPS)
Auteur :
Adobe Illustrator(TM) 3.2
Aperçu :
Cette image EPS n'a pas été enregistrée
avec un aperçu intégré.
Commentaires :

Contrôle continu écrit de novembre Corrigé sommaire

A QCM (ordre d'idée : 5 points)

Pour chacune des questions qui suivent, plusieurs affirmations sont proposées en guise de réponses. Parmi celles-ci, *zéro, une ou plusieurs* peuvent être vraies. Vous devez cocher la case figurant devant chaque affirmation vraie. Toute rature susceptible de donner un caractère ambigu à votre réponse sera considérée comme une erreur. Chaque question est notée indépendamment des autres. Le barème appliqué sera le suivant : 1 point attribué *si tout est juste*, 0 point sinon.

a1 On considère la classe suivante :

```
class A1 {  
    public static void main(String[] a) {  
        int i= 15, j= 20;  
        System.out.println("Les deux valeurs : "+i.toString()+" "+ j);  
    }  
}
```

Cocher les affirmations justes : *(le caractère carré noir indique les cases à cocher)*

- Le programme affiche **Les deux valeurs : 15 20**
- L'opérateur + utilisé effectue l'addition des valeurs de i et de j
- La définition de la classe A1 comporte une erreur détectée à la compilation
- out est un champ de donnée de classe de la classe System

La variable i, déclarée comme étant d'un type de base (mot-clé int), ne peut accéder à une méthode d'objet de nom toString car les seules actions disponibles pour les types simples sont les opérateurs prédéfinis du langage; en conséquence, il y a erreur de compilation et le programme ne fournit aucun résultat. L'expression de calcul comporte une constante chaîne et les opérateurs + sont donc des concaténations (cette remarque sert également en a3). Le champ de donnée out (nom commençant par une minuscule, non suivi d'une parenthèse) est préfixé par le nom de classe System (nom commençant par une majuscule) : out est un champ de donnée de classe.

a2 On considère la déclaration des deux classes suivantes :

```
class A2a {  
    private int j;  
    double k;  
  
    A2a() { k= 20; j= 10; }  
}
```

```

class A2b {
    private A2a ref= new A2a();

    public String toString() {
        return ""+ref.j+ref.k;
    }
}

```

Cocher les affirmations justes : *(le caractère carré noir indique les cases à cocher)*

- La définition de la classe A2a compile sans erreur
- La définition de la classe A2b compile sans erreur
- Le champ ref de A2b est uniquement accessible via un accesseur
- La définition de la classe A2a comporte un constructeur par défaut
- La définition de la classe A2b comporte un constructeur par défaut

ou bien

- La définition de la classe A2a compile sans erreur
- La définition de la classe A2b compile sans erreur
- Le champ ref de A2b est uniquement accessible via un accesseur
- La définition de la classe A2a comporte un constructeur par défaut
- La définition de la classe A2b comporte un constructeur par défaut

La méthode toString de A2b tente d'accéder directement à un champ de donnée private de la classe A2a : il y a erreur de compilation. Tout champ private de A2b est directement accessible depuis tout endroit de la définition de la classe A2b : on n'a pas besoin d'un accesseur pour accéder aux champs de données de la définition de la classe, ce qui veut dire que l'accesseur est une notion valable pour les classes qui utilisent d'autres classes. Le constructeur sans paramètre de A2a correspond au constructeur par défaut; sous cet angle de vue, A2b ne possède pas de constructeur par défaut, mais on peut dire que l'héritage permet de considérer que A2b possède un constructeur par défaut à l'exécution.

a 3 On considère la déclaration des deux classes suivantes :

```

class A3a {
    public int l=40, m=50;
}

class A3b {
    public static void main(String[] a) {
        A3a ref= new A3a();
        System.out.print(ref.m+ref.l);
        System.out.println(" "+ref.l+ref.m);
    }
}

```

Cocher l'affirmation juste : *(le caractère carré noir indique la case à cocher)*

- Le programme affiche 5040 4050
- Le programme affiche 90 4050
- Le programme affiche 90 90

On dit que l'opérateur + est surchargé : si les deux opérandes sont entiers, alors il réalise l'addition des entiers, si l'un de ses deux opérandes est un objet de la classe String, alors il réalise la concaténation des chaînes. Le premier + est l'addition des entiers, tandis que le deuxième est la concaténation des chaînes.

a 4 On considère la déclaration des deux classes suivantes :

```
class A4a {
    private int nbrA4a= 1;
    private int rang;

    A4a() {
        rang= nbrA4a;
        nbrA4a++;
    }

    public String toString() { return ""+rang; }
}

class A4b {
    public static void main(String[] a) {
        A4a q0,q1,q2;
        q1= new A4a(); q0= new A4a(); q2= new A4a();
        System.out.print(q0+" "+q1+" "+q2);
    }
}
```

Cocher l'affirmation juste : (*le caractère carré noir indique la case à cocher*)

- Le programme affiche 1 2 3
- Le programme affiche 2 1 3
- Le programme affiche 3 2 1
- Le programme affiche 1 1 1
- Le programme affiche 3 3 3

Le champ de donnée nbr4a de la classe A4a est un champ de donnée d'objet (absence du mot-clé static) : il est initialisé à 1 (comme indiqué dans la déclaration de la variable) à chaque création d'objet de la classe A4a.

a 5 On modifie la deuxième ligne de la déclaration de A4a :

```
class A4a {
    private static int nbrA4a= 1;
    ...
}
```

Cocher l'affirmation juste : (*le caractère carré noir indique la case à cocher*)

- Le programme affiche 1 2 3
- Le programme affiche 2 1 3
- Le programme affiche 3 2 1
- Le programme affiche 1 1 1
- Le programme affiche 3 3 3

Le champ de donnée nbr4a de la classe A4a est maintenant un champ de donnée de classe : il est initialisé à 1 avant toute création d'objet de la classe A4a et est incrémenté de 1 à chaque création d'objet de la classe A4a. Attention : l'ordre de création des objets de A4a est objet repéré par q1, puis objet repéré par q0, et enfin objet repéré par q2, mais l'ordre d'affichage est bien q0, puis q1, puis q2, ce qui veut dire que l'on obtient bien 2 1 3 et non pas 1 2 3.

B Problème (ordre d'idée : 15 points)

Présentation du problème

Il y a seulement 30 ans, on trouvait encore des régions de France où chaque village parlait en plus du français son propre parler local : chacun des villages de la région parle sa propre langue, commune à tous ses habitants. Il s'avère particulièrement intéressant d'étudier ces parlers locaux sur une aire géographique déterminée en fonction des caractéristiques langagières étudiées. Pour cela, on se rend dans un certain nombre de lieux d'enquête (des villages) de la région concernée, et l'on y enregistre le parler local de quatre à sept individus dits « informateurs ». On ne s'intéresse nullement ici aux caractéristiques des manières de parler, mais l'on se restreint à la *description sommaire des villages et des informateurs*.

Pour des raisons de respect de la vie privée des individus, chaque informateur est identifié par un code numérique, unique, automatiquement attribué. Il en est de même pour les villages. Chaque village comporte une description de tous les individus-informateurs qui y ont été « interrogés ».

Chaque informateur sera représenté en mémoire comme un objet de la classe **Individu**. Chaque lieu d'enquête est représenté par un objet de la classe **Village**. La classe **Enquete** permet de représenter l'ensemble des villages étudiés, et, en conséquence, l'ensemble des informateurs pour tous les villages enquêtés. On va étudier l'ensemble de ces trois classes, auxquelles on adjoindra une quatrième classe **Test** qui permettra d'exécuter le programme. Le nombre d'informateurs par villages et leur âge seront déterminés aléatoirement.

L'exécution du programme que forment ces quatre classes fournit l'affichage à l'écran suivant :

```
Village 0 : 1(79) 2(83) 3(71) 4(62)
Village 1 : 5(81) 6(72) 7(68) 8(79) 9(85) 10(70)
Village 2 : 11(69) 12(62) 13(84) 14(71)
Village 3 : 15(83) 16(64) 17(85) 18(64) 19(76) 20(61) 21(60)
Village 4 : 22(76) 23(63) 24(85) 25(60) 26(75)

Ages des 26 informateurs (en vrac)
79 83 71 62 81 72 68 79 85 70 69 62 84 71 83 64 85 64 76 61 60 76 63 85 60 75
```

Pour cette exécution particulière, cela veut dire :

Il y a cinq villages.

Il y a eu 4 informateurs pour le village 0, 6 pour le village 1, ... et 5 pour le village 4.

Il y a au total 26 informateurs, de code numérique 1 à 26 dans l'ordre de leur prise en compte.

L'âge de chacun des informateurs est indiqué entre parenthèses, après son code.

Enfin, on liste en vrac l'âge de tous les informateurs.

b1 classe *Individu*

Les objets de cette classe ne comportent que deux champs de donnée, pour l'âge et pour le numéro de l'individu informateur. Le premier de tous les individus de l'enquête a comme numéro 1, le second 2, le troisième 3 ... Il y a deux constructeurs, l'un à un seul paramètre pour l'âge de l'informateur, et l'autre sans paramètre. Un accesseur permet de connaître l'âge de l'individu; il est également nécessaire de définir un modifieur de l'âge car il existe la possibilité d'initialiser l'âge à zéro à la création de l'objet. La méthode toString décrit l'objet conformément à certaines des caractéristiques de l'affichage donné dans la présentation du problème.

Donner la déclaration complète de la classe.

b2 classe *Village*

Les objets de cette classe ne comportent que deux champs de donnée, pour la liste des informateurs du village (un tableau) et pour le numéro du village. Le premier village a comme numéro 0, le second 1, ... L'âge d'un individu informateur est déterminé aléatoirement, entre 60 et 85 ans, valeurs qui sont des constantes **MINAGE** et **MAXAGE** déclarées comme variables de classe de Village. Il y a un seul constructeur, à un seul paramètre pour le nombre d'informateurs du village. La méthode toString décrit l'objet conformément à certaines des caractéristiques de l'affichage donné dans la présentation du problème.

Donner la déclaration complète d'une telle classe.

b3 classe *Village* : méthode `infoSuivant`

Cette méthode permet de parcourir successivement un à un et à la « demande » tous les informateurs d'un village : elle renvoie donc un individu, repéré par une variable (champ de donnée) entière `enCours` servant à repérer la case du tableau des informateurs contenant l'informateur « suivant ». Si la valeur de cette variable est au delà des cases du tableau, la méthode renvoie null et la valeur de `enCours` est mise pour la première case du tableau des informateurs.

Donner la déclaration complète de la méthode d'objet `infoSuivant` de la classe `Village`.

b4 classe *Enquete*

L'enquête, c'est l'ensemble des villages enquêtés, représentés par un tableau. Le constructeur sans paramètre utilise trois constantes de classe de `Enquete` : `NBRVILL` pour le nombre de villages enquêtés; et `MININFO` et `MAXINFO` comme bornes de détermination aléatoire du nombre d'informateurs du village. La méthode `toString` décrit l'objet conformément à l'affichage donné dans la présentation du problème, à l'exception de l'affichage des ages de tous les informateurs.

Donner la déclaration complète d'une telle classe.

b5 classe *Test*

Elle permet d'exécuter l'ensemble de ces trois classes, pour fournir un résultat conforme à l'affichage donné dans la présentation du problème, à l'exception de l'affichage des ages de tous les informateurs, qui fait l'objet de la question suivante.

Donner la déclaration complète d'une telle classe.

b6 classe *Enquete* : méthodes `ages` et `agesToString`

La méthode sans paramètre `ages` renvoie un tableau d'entiers, dont les contenus des cases sont les ages de tous les informateurs de l'enquête. Ce tableau est rendu affichable grâce à la méthode `agesToString()`.

On dispose dans `Enquete` de la liste des villages; un premier parcours de cette liste, et pour chaque village un parcours des informateurs du village, permet de déterminer le nombre total des informateurs de l'enquête, sans rien avoir à modifier à la classe `Village`. On peut alors créer l'objet tableau, qui sera fourni en résultat de la méthode. Un nouveau parcours, dans les mêmes conditions, de la liste des villages permet alors d'initialiser les valeurs des cases du tableau fourni en résultat.

Donner la déclaration complète de la méthode d'objet `ages` de la classe `Enquete`.

Donner la déclaration complète de la méthode d'objet `agesToString` de la classe `Enquete`.

Quelle instruction faut-il rajouter dans la classe `Test` pour faire afficher ce tableau..

```

class Individu {
    private static int rang= 1;

    private int numero,age;

    Individu(int a) {
        age= a;
        numero= rang++;
    }
    Individu() { this(0); }

    public void setAge(int a) { age=a; }
    public int getAge() { return age; }

    public String toString() { return numero+"("+age+")"; }
}

class Village {
    public static final int MINAGE= 60,MAXAGE= 85;
    private static int numVill=0;
    private int numero;
    private Individu[] informateurs;

    Village(int nbInd) {
        numero= numVill++;
        informateurs= new Individu[nbInd];
        for (int i= 0; i<nbInd; i++)
            informateurs[i]= new Individu((int) (Math.random()*(MAXAGE-MINAGE+0.4))+MINAGE);
    }

    private int enCours= 0;
    public Individu infoSuivant() {
        if (enCours==informateurs.length) { enCours= 0; return null; }
        else return informateurs[enCours++];
    }

    public String toString() {
        String res= "Village "+numero+" : ";
        for (int i= 0;i<informateurs.length;i++)
            res+= informateurs[i]+" ";
        return res;
    }
}

class Enquete {
    public final static int MININFO= 4,MAXINFO= 7, NBRVILL= 5;

    private Village[] lesVillages;

    Enquete() {
        lesVillages= new Village[NBRVILL];
        for (int v= 0; v<NBRVILL; v++)
            lesVillages[v]= new Village( (int) (Math.random()*(MAXINFO-MININFO+0.4))+MININFO );
    }

    public String toString() {
        String res="";
        for (int v=0; v<lesVillages.length; v++) res+= lesVillages[v]+"\\n";
        return res;
    }
}

```

```

private int[] ages() {
    int nbInfo= 0;
    for (int vl= 0; vl<lesVillages.length; vl++) {
        Individu indiv= lesVillages[vl].infoSuivant();
        while (indiv!=null) {
            nbInfo++;
            indiv= lesVillages[vl].infoSuivant();
        }
    }
    int[] res= new int[nbInfo];
    int a= 0;
    for (int vl= 0; vl<lesVillages.length; vl++) {
        Individu indiv= lesVillages[vl].infoSuivant();
        while (indiv!=null) {
            res[a++]= indiv.getAge();
            indiv= lesVillages[vl].infoSuivant();
        }
    }
    return res;
}

public String agesToString() {
    int[] tbAges= ages();
    String res="Ages des "+tbAges.length+" informateurs (en vrac)\n";
    for (int a=0; a<tbAges.length; a++) res+= tbAges[a]+" ";
    return res;
}

}

class Test {
    public static void main(String[] a) {
        Enquete e= new Enquete();
        System.out.println(e);
        System.out.println(e.agesToString());
    }
}

/* exemple d'affichage a l'ecran

Village 0 : 1(79) 2(83) 3(71) 4(62)
Village 1 : 5(81) 6(72) 7(68) 8(79) 9(85) 10(70)
Village 2 : 11(69) 12(62) 13(84) 14(71)
Village 3 : 15(83) 16(64) 17(85) 18(64) 19(76) 20(61) 21(60)
Village 4 : 22(76) 23(63) 24(85) 25(60) 26(75)

Ages des 26 informateurs (en vrac)
79 83 71 62 81 72 68 79 85 70 69 62 84 71 83 64 85 64 76 61 60 76 63 85 60 75

*/

```