le cnam
Nouvelle-Aquitaine

Sujet d'examen

Semestre 1 / session 2
Année 2018/2019

CODE UE : M-NFA031-1

INTITULE : Programmation avec Java : notions de base

DATE : vendredi 12 avril 2019

HORAIRES : 18h00 à 20h00

DUREE : 2h00

CONSIGNES : Documents autorisés, matériel électronique interdit

NOMBRE DE PAGES : 5 celle-ci comprise

1. Cours (4 points)

1.1. Quels usages peut-on faire des exceptions ? Donner un exemple.

1.2. Dans le bout de code qui suit

```
return tab[i-1]*2;
```

a) De quoi s'agit-il ?
b) Qu'est-ce que tab ?
c) Quels sont les opérateurs ?
d) Quels sont les opérandes de * ?
e) Sachant que la méthode dans laquelle ce code apparaît est déclarée comme ci-dessous, quelles paires de types sont admissibles en lieu et place de XXX et YYY ?

```
public static XXX fct(YYY [] tab) {…
```

2. Erreurs (2 points)

2.1. Dans cet extrait de correction d'un exercice, des erreurs empêchant la compilation ont été introduites. Donnez-en 7 et proposez une correction pour chacune d'elles. A noter que les instructions import sont bien présentes un peu plus haut dans le fichier et n'engendrent aucune erreur.

```
01.         public static void ajouter(ArrayList<Double> notes,
02.                                    double lanote) {
03.               return notes.add(lanote);
04.         }
05.
06.         public static double moyenne(ArrayList<double> notes) {
07.               cumul = 0.0;
08.               for (int i = 0; i < notes.size(); i++)
09.                     cumul = cumul + notes.get(i)
10.               }
11.               return cumul / notes.length;
12.         }
13.
14.         public static afficher(ArrayList<Double> notes) {
15.               Terminal.ecrireString(Notes: );
16.               for (int i = 0; i < notes.size(); i++) {
17.                     Terminal.ecrireString(notes.get(i));
18.                     Terminal.ecrireString(" ");
19.               }
20.               Terminal.sautDeLigne();
21.         }
```

3. Algorithmique (4 points)

3.1. Donner les entrées, les sorties et l'algorithme d'un programme cherchant la plus petite valeur dans un tableau et la ramène par échange de valeurs dans la première case du même tableau.

3.2. L'algorithme suivant permet de trouver la lettre qui apparait le plus souvent dans un texte donné. Le traduire sous forme d'une méthode Java retournant cette lettre.

Rappels : L'expression 'a' convertie en int a pour valeur l'unicode du caractère a qui est 97, 'b', celui du caractère b qui est 98, et ainsi de suite. De manière générale, un caractère converti en int donne comme résultat l'unicode du caractère.

Le calcul d'une opération arithmétique (seulement + et – sont autorisées) entre deux caractères provoque la conversion implicite des deux caractères en int, puis effectue l'opération entière correspondante sur les int obtenus.

Entrées

texte : un chaîne de caractères.

Sortie

lettre : la lettre la plus fréquemment utilisée dans texte.

Variables locales

txtConv : le texte converti en minuscules.

tabCPT : un tableau de 26 compteurs tous initialisés à 0.

posMax : l'indice du compteur le plus élevé initialisé à 0.

indice : un entier pour calculer l'indice du compteur à incrémenter.

Algorithme

```
txtConv = texte converti en minuscules
pour i allant de 0 à la longueur de txtConv - 1
   si le caractère d'indice i de txtConv est une lettre
         alors
      indice = l'unicode du caractère d'indice i de
               txtConv – l'unicode du caractère a.
      tabCPT[indice]++
      si tabCPT[posMax] < tabCPT[indice] alors
         posMax = indice
      fin si
   fin si
fin pour

retourner (l'unicode du caractère a + posMax) converti
       en caractère.
```

fin de l'algorithme

4. Simulation (2 points)

4.1. Donner un tableau récapitulant la simulation de l'exécution ligne à ligne du programme suivant.

```
01.       public static void main(String[] args) {
02.            int n = 10;
03.            int p = 1;
04.            int r = 0;
05.
06.            while (n>0) {
07.                 if ((n+r)%2 == 0) {
08.                      if (n%2 == 0)
09.                           r = r / 2;
10.                      else
11.                           r = (r+p)/2;
12.                 } else {
13.                      if (n%2 == 0)
14.                           r = (3*r+1)/2;
15.                      else
16.                           r = (3*(p+r)+1)/2;
17.                      p = p*3;
18.                 }
19.                 n = n/2;
20.            }
21.            Terminal.ecrireStringln("Le résultat est " + r);
22.       }
```

5. Exercices (8 points)

Dans cette partie, si vous n'arrivez pas à répondre à une question, vous pouvez, dans les questions qui suivent, utiliser la méthode demandée comme si vous y aviez répondu.

5.1. Ecrire une fonction *estPair* prenant en paramètre un nombre entier *n* et retournant true si *n* est pair, false sinon.

5.2. Ecrire une fonction nommée *fairePas* ne retournant pas de résultat et prenant en paramètre : une *ArrayList* d'*Integer* nommée *list* et un nombre entier *n*. Cette fonction ajoutera à la fin de *list* : 0 si *n* est pair, 1, sinon.

5.3. Ecrire une fonction prenant en paramètre un entier *n* et retournant une *ArrayList* d'*Integer* contenant le code binaire de *n* inversé.

Pour ce faire, il faut créer l'*ArrayList* initialement vide, puis, tant que *n* est différent de 0, appeler *fairePas* avec la liste créée et *n*, et diviser *n* par 2. Quand *n* atteint 0, il suffit de retourner la liste.

Par exemple, si *n* vaut initialement 10 :

- on crée une liste vide
- n est différent de 0 et est pair
  - on ajoute 0 à la fin de la liste,
  - on divise n par 2 ; n vaut alors 5

- n est alors différent de 0 et est impair
  - on ajoute 1 à la fin de la liste
  - on divise n par 2 ; sa nouvelle valeur est alors 2
- n est toujours différent de 0 et est pair
  - on ajoute 0 à la fin de la liste,
  - on divise n par 2 ; n vaut alors 1
- n est encore différent de 0 et est impair
  - on ajoute 1 à la fin de la liste
  - on divise n par 2 ; n vaut 0
- n est égal à 0 -> on sort de la boucle
- on retourne la liste obtenue qui contient alors {0, 1, 0, 1}

Le code binaire de 10 est 1010…

Attention ! La méthode doit fonctionner quelle que soit la valeur de *n* passée en paramètre lors de l'appel et pas uniquement pour l'exemple donné ci-dessus

5.4. Ecrire une méthode prenant en paramètre une *ArrayList* d'*Integer* et retournant une nouvelle liste contenant les mêmes valeurs que celle prise en paramètre mais dans l'ordre inverse.

Par exemple, si la liste prise en paramètre contient {0, 1, 0, 1}, le résultat devra être une nouvelle liste contenant {1, 0, 1, 0}.

Attention ! La méthode doit fonctionner quelle que soit la liste passée en paramètre lors de l'appel et pas uniquement pour l'exemple donné ci-dessus.

5.5. Ecrire un *main* lisant au clavier un nombre entier positif *n* et, à l'aide des méthodes décrites ci-dessus, affichant à l'écran le code binaire de *n* dans le bon ordre.