IPST-CNAM
Programmation JAVA
NFA 031
Dans la semaine du 14/04/2014

Avec document
Durée : **2 h30**
Enseignant : LAFORGUE Jacques

2ème Session NFA 031

CORRECTION

*L'examen se déroule **en deux parties**. **Une première partie de 1h15mn**, **sans document**, consacrée à des questions de cours, et **une deuxième partie de 1h15mn**, avec document, consacrée en la réalisation de programmes Java.*

Au bout de 1h15mn, les copies de la première partie seront ramassées avant de commencer la deuxième partie.

*Pour la première partie, vous devez rendre le QCM rempli avec NOM et PRENOM renseigné, et les réponses aux questions libres **écrites sur des copies vierges**.*

Pour la deuxième partie, vous écrivez vos programmes sur des copies vierges.

1ère PARTIE : COURS (sans document)

1. QCM (35 points)

Mode d'emploi :

Ce sujet est un QCM dont les questions sont de 3 natures :

- **les questions à 2 propositions**: dans ce cas une seule des 2 propositions est bonne.
  - +1 pour la réponse bonne
  - –1 pour la réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est bonne
  - + 1 pour la réponse bonne
  - –½ pour chaque réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est fausse
  - + ½ pour chaque réponse bonne
  - –1 pour la réponse fausse

Il s'agit de faire une croix dans les cases de droite en face des propositions.

On peut remarquer que cocher toutes les propositions d'une question revient à ne rien cocher du tout (égal à 0).

Si vous devez raturer une croix, faites-le correctement afin qu'il n'y ait aucune ambiguïté.

N'oubliez pas d'inscrire en en-tête du QCM, votre nom et prénom.

Vous avez droit à **4 points** négatifs sans pénalité.

NOM:	PRENOM:

Dans un langage orienté objet, comme Java, les traitements informatiques sont portés par les classes et les objets du langage		Q 1.
1	OUI	**X**
2	NON	

Le langage JAVA est **portable** sur la plupart des plateformes (windows,unix,linux,....)		Q 2.
1	OUI	**X**
2	NON	

Dans la programmation orientée objet, une classe est un concept informatique qui contient la déclaration des informations suivantes :		Q 3.
1	des attributs privés ou publics, et des méthodes privés ou publics	**X**
2	des classes internes publics	
3	des méthodes statics et des attributs non statics	**X**

Le "garbage collector" ou ramasse miettes est un traitement de la JVM qui permet automatiquement de détruire les objets dont la référence n'est contenue dans aucun autre objet de la JVM		Q 4.
1	OUI	**X**
2	NON	

Soit le fichier suivant C:\CodeJava\exercices\cnam\util\Terminal.java.
Le fichier Terminal.java contient en 1ère ligne : **package exercices.cnam.util;**
Dans C:\CodeJava\programme se trouve le fichier Prog.java suivant:

```
import exercices.cnam.util.*;
public class Prog {
  public static void main(String… args) {
    Terminal.ecrireStringln("Bonjour");}
}
```

On est dans le répertoire C:\CodeJava\programme, et on veut compiler le programme.
Quelle(s) commande(s) est(sont) valide(s) :

Q 5.

1	javac –classpath "." Prog.java	
2	javac Prog.java	
3	javac –classpath ".." Prog.java	**X**

Le compilateur Java (javac) permet de créer un ensemble de fichier .class qui sont ensuite interprétés par une JVM		Q 6.
1	OUI	**X**
2	NON	

La signature d'une méthode **main** de la class **C1** permettant l'exécution d'un programme JAVA est de la forme **public static void main(String… args).** Cette méthode est static parce que la commande **java C1 a1 a2** consiste à demander à la JVM Java d'exécuter l'instruction JAVA suivante : **C1.main(a1,a2)**		Q 7.
1	OUI	**X**
2	NON	

La classe Exemple.java appartient au package "fr.cnam.prog". Soit l'arborescence de répertoires suivante : Exemple00/ bin/ fr/ cnam/ prog/ Exemple.java La commande de compilation est exécutée dans le répertoire Exemple00. Les fichiers compilés sont créés dans le répertoire bin. Cette commande peut être :		Q 8.
1	javac –d bin Exemple.java	
2	javac –d bin fr/cnam/prog/Exemple.java	**X**
3	javac fr/cnam/prog/Exemple.java	

La commande **java**		Q 9.
1	prend en entrée un fichier .java afin de l'interpréter	
2	prend en entrée un fichier .class afin de l'interpréter	**X**
3	exécute la méthode main de la classe java contenue dans le fichier .class qui est en entrée de la commande	**X**

La commande suivante : **java MaClasse.main2 AAA 999** exécute un programme Java dont la classe MaClasse est dans le fichier MaClasse.java et dont la méthode **main2** est déclarée de la manière suivante : `public static void main2(String arg1, int arg2)`		Q 10.
1	OUI	
2	NON	**X**

Soit une classe contenant les méthodes mstat1 et m2. mstat1 est une méthode statique et m2 n'est pas une méthode statique :		Q 11.
1	la méthode mstat1 peut utiliser les attributs statiques de la classe	**X**
2	la méthode m2 peut utiliser les attributs statiques de la classe	**X**
3	la méthode mstat1 peut utiliser les attributs non statiques de la classe	

En JAVA, les méthodes déclarées en dehors d'une classe sont appelées des méthodes statics		Q 12.
1	OUI	
2	NON	**X**

La caractéristique "private" d'un attribut rend inaccessible l'attribut par toutes les autres classes		Q 13.
1	OUI	**X**
2	NON	

Dans la programmation objet, en JAVA, le rôle du constructeur d'une classe est de :		Q 14.
1	affecter les valeurs des attributs de la classe	**X**
2	allouer en mémoire du programme les attributs de l'objet	**X**
3	construire la classe (ou .class) qui permet à un autre programme de créer les objets de la classe	

Quelque soit le cas de figure, il est possible d'utiliser le constructeur par défaut d'une classe (Le constructeur par défaut est exécuté par l'instruction ***Classe* obj = new *Classe* ( );**)		Q 15.
1	OUI	
2	NON	**X**

Le constructeur d'une classe A :		Q 16.
1	peut initialiser les attributs privés de la classe A	**X**
2	ne peut pas initialiser les attributs privés de la classe A	
3	peut initialiser les attributs publics d'une autre classe B public	**X**

Soit le code JAVA suivant : Q 17.

```
Livre l1 = new Livre();
l1.nom = "UN";
Livre l2 = l1 ;
l2.nom = "DEUX";
System.out.println(l1.nom);
```

Ce code affiche :

1	UN	
2	DEUX	**X**

En JAVA, le passage des paramètres se fait :		Q 18.
1	par référence	
2	par valeur	**X**

Soit le code suivant : Q 19.

```
int v=13;
boolean premier=true;
for(int k=2;k<v;k=k+1)
   if (v%k == 0) premier = false;
if (premier) System.out.println("PREMIER");
else System.out.println("NON PREMIER");
```

Ce code :

1	contient une erreur	
2	affiche "NON PREMIER"	
3	affiche "PREMIER"	**X**

Soit le code suivant : Q 20.

```
int tab_int[] = new int[10];
[...]
for(int i=0; i< A ;i++)
      Terminal.ecrireIntln( B );
```

Ce code affiche tous les éléments du tableau tab_int.
A et B peuvent être remplacés par :

1	**A** ➔ `tab_int.size()` **B** ➔ `tab_int.get(i)`	
2	**A** ➔ `10` **B** ➔ `i`	
3	**A** ➔ `tab_int.length` **B** ➔ `tab_int[i]`	**X**

Soit le code JAVA suivant : Q 21.

```
String slue = "un::deux trois,:quatre cinq:six sept huit";
StringTokenizer str = new StringTokenizer(slue," ,:");
while (str.hasMoreTokens())
{
   String s = str.nextToken();
   s=s+"_";
}
Terminal.ecrireString (s);
```

Ce code :affiche :

1	un_deux_trois_quatre_cinq_six_sept_huit	**X**
2	un_ _deux_trois_ _quatre_cinq_six_sept_huit	
3	un::deux_trois,:quatre_cinq:six sept_huit	

Soit le code JAVA suivant :
`String[] tab = new String[4];`
`tab[0]="UN"; tab[1]="DEUX";`
`for(String s : tab){System.out.println(s);}`
		Q 22.

1	Ce code n'est pas valide	
2	Ce code affiche : UN DEUX	
3	Ce code affiche : UN DEUX null null	**X**

Le constructeur de la classe ArrayList suivant : public ArrayList<String>(int taille) crée une collection de chaine de caractère dont le tableau en interne est dimensionné à *taille* éléments et il n'est donc pas possible de mettre plus de *taille* éléments dans la collection		Q 23.
1	OUI	
2	NON	**X**

On utilise les tableaux java [] et les ArrayList dans un même programme car un tableau peut contenir que des données de type primitif (int, double, ...) alors qu'un ArrayList peut contenir que des objets		Q 24.
1	OUI	
2	NON	**X**

En JAVA, il est possible de créer un tableau qui est le résultat de la concaténation de deux autres tableaux		Q 25.
1	OUI	**X**
2	NON	

Soit la création suivante d'un tableau dont les éléments sont des livres : **Livre[] mes_livres;** Ensuite il est possible de faire le code valide suivant :		Q 26.
1	mes_livres[0].titre = "Les misérables"; mes_livres[0].auteur = "Victor Hugo";	
2	mes_livres[0] = new Livre("Les misérables","Victor Hugo")	
3	mes_livres = new Livre[100]; mes_livres[0] = new Livre("Les misérables","Victor Hugo");	**X**

En JAVA, pour qu'un objet puisse être passé en paramètre d'une méthode, il faut que la classe d'appartenance de l'objet soit une classe public		Q 27.
1	OUI	
2	NON	**X**

Le fichier C1.java contient 2 classes : une classe publique C1 et une classe privée C2. La commande **javac** C1.java		Q 28.
1	crée qu'un fichier : C1.class	
2	compile la classe C1 et exécute la méthode main de la classe C1	
3	crée deux fichiers : C1.class et C2.class	**X**

Soit le code suivant :
`StringTokenizer str = new StringTokenizer("AA  BB  CC");`
`System.out.println(str.nextToken());`
`System.out.println(str.nextToken());`
`System.out.println(str.nextToken());`
`System.out.println(str.nextToken());`
		Q 29.

1	Ce code affiche : AA BB CC (chaine vide)	
2	Ce code affiche : AA BB CC null	
3	Ce code est en erreur	**X**

Soit la classe suivante :
`public class Individu`
`{`
`  private String nom;`
`  private String prenom;`
`  private int age;`
`  public Individu(String nom, String prenom, int age)`
`  {`
`  this.nom=nom;  this.prenom=prenom;  this.age = age;`
`  }`
`  public Individu(String nom, String prenom)`
`  {`
`  this.nom=nom;  this.prenom=prenom;  this.age = 0;`
`  }`
`}`
On peut écrire les codes suivants :
		Q 30.

1	Individu ind = new Individu("LAFONT","Pierre",45);	**X**
2	Individu ind = new Individu();	
3	Individu ind = new Individu("LAFONT","Pierre");	**X**

La méthode equals de l'API Java permet de tester l'égalité de deux objets. Cette méthode teste l'égalité des attributs de chacun des objets. Elle retourne VRAI si tous les attributs sont égaux deux à deux.		Q 31.
1	OUI	
2	NON	**X**

Soit le code suivant :
`StringTokenizer str1 = new StringTokenizer("AA;BB;CC)";");`
`StringTokenizer str2 = new StringTokenizer("AA/BB/CC","/");`
`if (str1.equals(str2)) System.out.println("EGAL");`
`else System.out.println("NON EGAL");`
Ce code affiche :
		Q 32.

1	EGAL	
2	NON EGAL	**X**

Soit la classe C1 qui contient un attribut privé. Une méthode de la classe C2 prend en paramètre une instance de la classe C1 et veut changer la valeur de cet attribut. Cela n'est possible que s'il existe un setteur sur cet attribut dans la classe C1.		Q 33.
1	OUI	**X**
2	NON	

Soit le code suivant est :

```
int[][] matrice = new Matrice[10][20];
matrice[0] = new int[3];
```

On peut écrire ensuite les codes suivants :

		Q 34.
1	`System.out.println(matrice[0][0]);`	**X**
2	`System.out.println(matrice[0][2]);`	**X**
3	`System.out.println(matrice[1][1];`	

Dans un fichier source JAVA (.java), on peut mettre plusieurs classes publiques		Q 35.
1	OUI	
2	NON	**X**

(Tourner la page)

2. Questions libres (15 points)

Chaque question est notée sur 5 points.
*Vous répondez à ces questions sur une **copie vierge** en mettant bien le numéro de la question, sans oublier votre nom et prénom.*

Q 1

Quels sont les rôles d'un constructeur ?
Ecrire un exemple de constructeurs suffisamment représentatif des rôles cités.

Les rôles d'un constructeur sont :
- allouer en mémoire l'objet c'est-à-dire réservez une place mémoire pour chacun des attributs qu'ils soient privés ou publiques
- initialiser les attributs avec des valeurs calculées ou passées en paramètre
- réaliser des traitements qui doivent être fait à chaque fois qu'un objet est créé

```
public Individu(String insee, String nom, String prenom, int age)
{
  this.insee = insee;
  this.nom = nom;
  this.prenom = prenom;
  this.age = age;
  this.adresse = "adresse inconnue";
  this.numeroCarteIdentite = accederFichierNational(insee);
  if (this.numeroCarteIdentite==null)
    System.out.println("Le numero de caret d'identite de "+insee+" est inconnue";
}
```

Q 2

Expliquer à quoi sert la classe StringTokenizer.

La classe StringTokenizer est utilisée pour extraire, depuis une chaine de caractère, des "tokens (ou mots)" qui sont séparés par des caractères spécifiques (souvent des caractères spéciaux). Par exemple, extraire les mots d'une phrase espacés par des blancs, extraire des noms de répertoire d'un path espacés par le caractère "/".

Q 3

Qu'est ce qu'un *attribut static* dans une classe ?
Donner un exemple de l'utilisation d'un attribut statique.

En programmation objet, un attribut static d'une classe est un attribut qui est commun à toutes les instances de la classe. Il est souvent initialisé dans le corps de la classe. Il peut être utilisé dans toutes les méthodes (non statiques et statiques) de la classe.

Par exemple un attribut statique est une constante : public static double PI = 3.141592654;

2ème PARTIE : PROGRAMMATION (avec document)

Exercice 1 [25 points]

Soit un programme qui gère un agenda dans lequel il est possible d'ajouter des rendez-vous qui contiennent les informations suivantes :

- **date** de début (String au format :
  "JJ//MM/AA:HH/MM" **ou**
  "Jour Numéro_semaine HH/MM" où Jour est égal à "Lundi" ou "Mardi", ...)
- **durée** du rendez-vous, en minutes
- **objet** du rendez-vous, texte libre
- **lieu** du rendez-vous, texte libre (exemple, une salle de réunion)
- **invités**, liste des noms des invités au rendez-vous (par exemple, pour une réunion).

1/ Ecrire la classe **RendezVous** qui est la définition d'un rendez-en fonction des consignes suivantes :

La classe contient trois constructeurs :
- un constructeur sans paramètre qui crée un rendez-vous dont les informations sont renseignées avec des valeurs par défaut.
- un constructeur qui a en paramètre que la date et l'objet du rendez-vous. Les autres champs sont optionnels : la durée (par défaut 60mn), le lieu (par défaut vide), les invités (par défaut vide)
- un constructeur qui a en paramètre tous les champs d'un rendez-vous.

La classe contient une méthode qui permet de saisir tous les champs d'un rendez-vous (utilisez la classe Terminal vu en cours). Si le champ n'est pas optionnel alors si la saisie est vide le signaler avec un texte d'erreur. Si le champ est optionnel alors un simple retour chariot valide la saisie, et le champ prend la valeur par défaut.

```
public class RendezVous
{
   private String date;
   private int duree; // en minutes
   private String objet;
   private String lieu;
   private ArrayList<String> invites;

   public RendezVous()
   {
      date=dateAujourdhui(); //date système d'aujourd'hui
      duree=60;
      objet="sans objet";
      lieu="";
      invites=new ArrayList<String>();
   }

   public RendezVous(String date,String objet)
   {
      this.date=date;
      this.duree=60;
      this.objet=objet;
      this.lieu="";
      this.invites=new ArrayList<String>();
   }

   public RendezVous(String date,int duree,String objet,String lieu,
                     ArrayList<String> invites)
   {
      this.date=date;
      this.duree=duree;
      this.objet=objet;
      this.lieu=lieu;
```

```
      this.invites=invites;
   }

   public void saisir()
   {
      Terminal.ecrireString("Date:");
      String sdate = Terminal.lireString();
      Terminal.ecrireString("Duree:");
      String sduree = Terminal.lireString();
      Terminal.ecrireString("Objet:");
      String sobjet = Terminal.lireString();
      Terminal.ecrireString("Lieu:");
      String slieu = Terminal.lireString();

      if (sdate.equals(""))
         { Terminal.ecrireStringln("Erreur:La date est vide"); return; }
      if (sobjet.equals(""))
         { Terminal.ecrireStringln("Erreur:La date est vide"); return; }

      this.date=sdate;
      if (sduree.equals("")) this.duree=0;
      else this.duree=Integer.parseInt(sduree);
      this.objet=sobjet;
      this.lieu=slieu;

      ArrayList<String> liste = new ArrayList<String>();
      String invite;
      do{
         Terminel.ecrireString("invite:");
         invite=Terminal.lireString();
         liste.add(invite);
      }while(! invite.equals(""));
      this.invites=liste;
   }
```

2/ On veut gérer ces rendez-vous dans un agenda (classe **Agenda**). Cette classe contient la méthode ajouterRdv qui saisie un agenda et l'ajoute dans une collection de rdv gérée par la classe.

```
publi class Agenda
{
   private ArrayList<RendezVous> rdvs;

   public Agenda
   {
      rdvs = new ArrayList<RendezVous>();
   }

   public void ajouterRdv()
   {
      RendezVous rdv = new RendezVous();
      rdv.saisir();
      rdvs.add(rdv);
   }
}
```

Exercice 2 [15 points]

Soit une collection ArrayList<String> dont chaque élément est une phrase en français. Les mots de chaque phrase sont espacés par un ou plusieurs blancs.

Ecrire la méthode qui retourne tous les mots de ces phrases sans qu'ils se répètent et qui ne prend pas les articles ("l'", "le", "la", "les", "un, "une", "des"). Les articles sont stockés dans un tableau String[] articles.

```
public ArrayList<String> exercice2(ArrayList<String> phrases)
{
   ArrayList<String> resultats = new ArrayList<String>');
   String[) exceptions = {"l'", "le", "la", "les", "un, "une", "des"};

   for(String phrase : phrases)
   {
      StringTokenizer strtok = new StringTokenizer(phrase);
      while(strtok.hasMoreTockens())
      {
         String mot = strtok.nextToken();

         boolean trouve=false;
         for(String s:exceptions)
            if (mot.equals(s)) trouve=true;
         if (!trouve)
         {
            if (! resultats.contains(mot))
               resultats.add(mot);
         }
      }
   }

}
```

(Fin du sujet)