jIPST-CNAM
Programmation JAVA
NFA 001
Avril 2016

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

2ème Session NFA 031

L'examen se déroule en deux parties. *Une première partie de 1h15mn, sans document, consacrée à des questions de cours, et une deuxième partie, avec document, consacrée à 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 et les réponses aux questions libres écrites sur des copies vierges.

Pour la deuxième partie, vous écrivez vos programmes sur des copies vierges. Vous devez écrire le code en Java.

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, un objet est :		Q 1.
1	une structure de données contenant des valeurs qui répond à un ensemble de « messages » (ou « méthodes »)	
2	une zone mémoire du langage, allouée lors de la création de l'objet contenant les caractéristiques de l'objet	
3	une vision virtuelle d'un paquet d'information qui est extérieur au programme	

Dans la programmation orientée objet, un objet est :		Q 2.
1	une instance d'une classe créée par exemple dans le programme principal	
2	un processus informatique qui s'exécute indépendamment du programme principal	

Le langage JAVA est **portable** sur la plupart des plateformes (windows,unix,linux,....)		Q 3.
1	OUI	
2	NON	

Dans un langage orienté objet, un principe fort est que les attributs (ou données) non statiques sont alloués dans une instance d'une classe appelée un objet. Ainsi,		Q 4.
1	les données d'une instance sont toujours directement (sans passer par une méthode) modifiées par toutes les autres instances du programme	
2	Une instance peut "protéger" ses données en ne les rendant pas directement accessibles par les autres instances du programme	
3	les données du programme sont réparties dans toutes les instances du programme	

Soit le fichier suivant C:\CodeJava\exercices\fr\cnam\util\Terminal.java. Le fichier Terminal.java contient en 1ère ligne : **package exercices.fr.cnam.util;** Le répertoire C:\bin est vide. Dans C: on réalise la commande suivante : **javac -d C:\bin C:\CodeJava\exercices\fr\cnam\util\Terminal.java** On obtient la création du fichier suivant : C:\bin\exercices\fr\cnam\util\Terminal.class		Q 5.
1	OUI	
2	NON	

Les fichiers .class générés par la compilation javac sont des fichiers intermédiaires qui contiennent un programme exprimé dans un autre langage (P-CODE) et qui est ensuite :		Q 6.
1	traduit en fichier binaire directement exécuté par le microprocesseur	
2	interprété par un interpréteur de code P-CODE	

En Java, l'exécution d'un programme peut se faire d'autant de façons différentes qu'il existe de méthode main dans les classes du programme		Q 7.
1	OUI	
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 8.
1	OUI	
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 9.
1	javac –d bin Exemple.java	
2	javac –d bin fr/cnam/prog/Exemple.java	
3	javac fr/cnam/prog/Exemple.java	

Le paramètre **-classpath** ou la variable d'environnement **CLASSPATH** est utilisée pour désigner une liste de plusieurs path d'accès à des répertoires. Chacun de ces répertoires contient les fichiers .class ou les packages utilisés dans la compilation ou dans l'exécution d'un programme JAVA.		Q 10.
1	OUI	
2	NON	

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	
2	la méthode m2 peut utiliser les attributs statiques de la classe	
3	la méthode mstat1 peut utiliser les attributs non statiques de la classe	

Soit le code JAVA suivant : Q 12.

```
public class C {
   private int attribut;
   public C(){attribut=10;}
   public static void main(String[] args)
   {
      C  objet = new C();
      objet.attribut = Integer.parseInt(args[0]);
   }
}
```

Ce code est correct		
1	OUI	
2	NON	

Les méthodes public de la classe A peuvent utiliser directement les attributs privés de la classe A		Q 13.
1	OUI	
2	NON	

Un attribut déclaré en protected est un attribut qui est protégé en écriture. Il est une constante du programme.		Q 14.
1	OUI	
2	NON	

Soit la classe C1 dont tous les attributs sont privés et statiques. Soit la classe C2 appartenant à un autre package que C1. Dans ce cas, les méthodes de C2 peuvent accéder aux attributs de C1		Q 15.
1	OUI	
2	NON	

La caractéristique "private" d'un attribut rend inaccessible l'attribut par toutes les autres classes		Q 16.
1	OUI	
2	NON	

En JAVA, il est possible de définir deux méthodes de même nom dans la même classe		Q 17.
1	OUI	
2	NON	

Le constructeur dans une classe d'objet permet :		Q 18.
1	de construire la classe de définition de l'objet (attributs et méthodes)	
2	d'initialiser les attributs de l'objet avec des valeurs bien particulières	
3	d'initialiser les attributs en fonction des paramètres du constructeur	

Soit la classe suivante : Q 19.

```
public class C1{
  private int x;
  public C1(int x){ this.x = x; }
}
```

L'instruction suivante : C1 c1 = new C1(); est valide et la valeur de x est la valeur par défaut de Java 0

1	OUI	
2	NON	

Le code suivant est un exemple correct d'un constructeur : Q 20.

```
public class Individu
{
  private String nom;
  private String prenom;
  private int age;

  public Individu Individu(String nom, String prenom, int age)
  {
    this.nom=nom;  this.prenom=prenom;  this.age = age;
    return this;
  }
}
```

1	OUI	
2	NON	

Quand une classe définit un constructeur avec des paramètres alors le constructeur par défaut (sans paramètre) n'est plus accessible		Q 21.
1	OUI	
2	NON	

En JAVA, une classe peut contenir plusieurs constructeurs		Q 22.
1	OUI	
2	NON	

Soit le code JAVA suivant — Q 23.

```
public class Constructeur {
    public static void main(String args[])
    {
        Exemple1 ex = new Exemple1();
        ex.tab[0] = 22;
    }
}

class Exemple1
{
    public int[] tab;

    public void Exemple1()
    {
        tab = new int[5];
    }
}
```

Ce code s'exécute correctement :

1	OUI	
2	NON	

Soit la classe suivante : — Q 24.

```
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 :

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

Soit le code JAVA suivant : Q 25.

```
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	

En Java, il est possible de modifier le contenu d'un objet passé en paramètre d'une méthode Q 26.

1	OUI	
2	NON	

Soit le code suivant : Q 27.

```
public class Test2 {
    public static void main(String a_args[])
    {
        int v = 10;
        proc(v);
        Terminal.ecrireIntln(v);
    }

    static void proc(int p)
    {
        p = 100;
    }
}
```

Le résultat est :

1	10	
2	100	

Soit le code suivant : Q 28.

```
int v=4;
boolean premier=true;
for(int k=2;k<v;k=k+1)
   if (v%k == 0) premier = false;
   else premier = true;
if (premier) System.out.println("PREMIER");
else System.out.println("NON PREMIER");
```

Ce code :

2	affiche "NON PREMIER"	
3	affiche "PREMIER"	

La boucle **for** dite "énumérative" permet : Q 29.

1	d'incrémenter une valeur scalaire de type enum et de réaliser un traitement à chaque valeur	
2	de parcourir tous les éléments d'une instance de ArrayList	
3	de parcourir tous les éléments d'un tableau java (tab[])	

Soit le code suivant : Q 30.

```
int i=0;
int[] tab = new int[3];
for(int v : tab)
{
  tab[i] = v*10;
  i = i+1;
}
for(int j=0;j<tab.length;j++)
  System.out.print(tab[j]+" ");
```

Ce code affiche :

1	0 10 100	
2	0 0 0	

En JAVA, le type de retour d'une méthode est soit void, soit un type primitif, soit la référence d'un objet	Q 31.
1 OUI	
2 NON	

En JAVA, un tableau (tab[])	Q 32.
1 peut contenir des éléments de type primitif	
2 peut contenir des références d'objet JAVA	
3 ne peut pas contenir de références d'objet JAVA	

Soit le code suivant : Q 33.

```
String str1[] = new StringTokenizer("AA;BB;CC",";");
for(String s: str1)
  System.out.println(s);
```

Ce code ::

1	affiche : AA BB CC
2	n'affiche rien
3	ne se compile pas correctement

Dans la classe String la méthode d'objet ***void set(int i, char c)*** change le i-ième caractère d'une chaine de caractère avec la valeur c.	Q 34.
1 OUI	
2 NON	

En JAVA, la déclaration d'un tableau comme suit : Q 35.

```
Individu[] tab_ind = new Individu[10]
```

contient :

1	10 éléments dont les valeurs sont égales au résultat de l'exécution du constructeur *Individu()*
2	10 éléments dont les valeurs sont toutes à *null*

(Tournez 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

Expliquez le rôle de la classe Java prédéfinie **ArrayList<E>**.
Dans votre explication précisez ce que signifie la syntaxe <E>.

Q 2

Expliquez de quoi est composée une classe en JAVA (Détaillez, ne restez pas trop général).

Q 3

Expliquez ce qu'est un constructeur en JAVA.

Fin de la 1ère partie

2ème PARTIE : PROGRAMMATION (avec document)

Problème 1 [15 points]

Soit les classes **Compte** et **Operation** vues dans le cadre du projet de cette année (voir l'annexe ci-après pour rappel).

Ecrire le code de la méthode de la classe **Compte** suivante :

```
public ArrayList<Operation> getOperationsType(String type)
```

Cette méthode retourne une collection contenant toutes les opérations qui sont du **type** passé en paramètres : "CB","CHEQ", "VIRT", "PRVT", "DEPT" ou "".

Problème 2 [35 points]

Soit les classes **Compte** et **Operation** vues dans le cadre du projet de cette année (voir l'annexe ci-après pour rappel).

On se propose de créer une classe **ComptesEnAnomalie** qui permet de gérer une liste des comptes bancaires des clients de la banque qui sont en anomalie.

Cette classe gère une liste de nom de compte (String).
Tous les comptes sont des fichiers qui se trouvent dans le répertoire data.
Le nom du compte correspond au fichier data/Compte<*nom du compte*>.txt contenant les opérations du compte.

Un compte en anomalie est un compte dont la somme des montants de toutes les opérations non budgétées est inférieure à la valeur seuil (exemple : -500 euros). Cette valeur seuil est un attribut de la classe **ComptesEnAnomalie**.

1/ Ecrire les attributs et le constructeur de la classe **ComptesEnAnomalie.**

2/ Ecrire la méthode de la classe **Compte** suivante :

```
public double solde()
```

Cette méthode calcule la somme des montants de toutes les opérations non budgétées du compte.

3/ Ecrire la méthode de la classe **ComptesEnAnomalie** suivante :

```
public void ajouter(String nomDuCompte)
```

Cette méthode ajoute le **nomDuCompte** dans la liste si le solde de ce compte est inférieur au seuil.

Annexe

Pour rappel (le code de ma correction) :

```
public class Compte
{
    private ArrayList<Operation> elements;  // La liste des operations

    public Compte(String nomFichier)
    {
       initialisation du compte avec la lecture du fichier
       […]
    }
}

public class Operation
{
    private String  date;           // date de l'operation
    private double  montant;        // le montant
    private String  type;           // "CB","CHEQ", "VIRT",
                                       "PRVT", "DEPT" ou ""
    private boolean budgete;        // vrai si operation budgete
                                          sinon faux
    private String  num_cb;         // numero de la CB de
                                       l'operation
    private int     num_cheque;     // numero du cheque de
                                       l'operation
    private String  libelle;        // libelle de l'operation
    private int     numero;         // numero de l'operation

    public Operation(int     numero,
                     String  date,
                     double  montant,
                     String  type,
                     boolean budgete,
                     String  num_cb,
                     int     num_cheque,
                     String  libelle)
    {
        //Initialisation des caracteristiques de l'operation
        this.numero      = numero;
        this.date        = date;
        this.montant     = montant;
        this.type        = type;
        this.budgete     = budgete;
        this.num_cb      = num_cb;
        this.num_cheque  = num_cheque;
        this.libelle     = libelle;
    }

    // Tous les getteurs
    […]
}
```

NB: Vous n'avez pas besoin d'écrire les getteurs, setteurs sur les attributs des classes et ni les méthodes toString() des classes. Vous considérez que ces méthodes vous sont données.

(Fin du sujet)