IPST-CNAM
Programmation JAVA
NFA 001
Mercredi 19 Février 2014

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

1ère 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 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 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 les codes commentés 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 la programmation orientée objet, un objet est :		Q 1
1	une instance d'une classe créée par exemple dans le programme principal	
2	une donnée structurée en mémoire de l'ordinateur	
3	un processus informatique qui s'exécute indépendamment du programme principal	

La difficulté dans la programmation objet est la maitrise de la destruction des objets		Q 2
1	Le langage JAVA ne fait pas exception à cette difficulté.	
2	Dans le langage JAVA cette difficulté est facilitée par l'implémentation systématique de la méthode ***dispose*** dans chacune des classes créées.	
3	Dans le langage JAVA cette difficulté n'existe plus grâce à un mécanisme interne appelé le garbage-collector (ou ramasse-miètes) qui détruit automatiquement les objets plus utilisés.	

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 3
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 intermédiaire (P-CODE) et qui est ensuite :		Q 4
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 5
1	OUI	
2	NON	

En Java, l'exécution d'un programme, via la commande "java", est possible par création de la méthode **main**. Cette méthode peut avoir la signature suivante :		Q 6
1	public void main()	
2	public static void main()	
3	public static void main(String… args)	

Soit le code suivant : Q 7

```
public class Exemple {
  public static void main(String args[]) {
    String str = args[1];
    System.out.println(str);
}}
```

Commande d'exécution : java Exemple toto

1	Ce programme ne se compile pas car il y a une erreur de syntaxe	
2	L'exécution échoue car il y a une erreur d'exécution	
3	L'exécution de ce programme affiche à l'écran la chaine de caractère passée en argument"	

Soit le code JAVA suivant : Q 8

```
public class Exemple
{
   ????? int var_x;
   public static void main(String a_args[])
      {
         var_x = Integer.parseInt(a_args[0]);
         System.out.println("X=" + var_x);
      }
}
```

On exécute ce programme avec la commande : java Exemple 123
Pour que ce code soit correct, il faut remplacer ????? par :

1	private	
2	public	
3	static	

Une méthode static est une méthode dont le contenu (le code) reste inchangé durant tout le temps d'exécution du programme Java Q 9

1	OUI	
2	NON	

Le constructeur dans une classe d'objet permet : Q 10

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 11

```
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	

Quand une classe définit un constructeur avec des paramètres alors le constructeur par défaut (sans paramètre) n'est plus accessible Q 12

1	OUI	
2	NON	

Soit le code suivant : Q 13

```
Livre l = new Livre();
l.nom = "Les cavernes d'acier";
ArrayList<Livre> livres = new ArrayList<Livre>();
livres.add(l);
l.nom="Face aux feux du soleil";
System.out.println(livres.get(0).nom);
```

Ce code affiche :

1	Les cavernes d'acier	
2	Face aux feux du soleil	

En Java, il est possible de modifier le contenu d'un objet passé en paramètre d'une méthode Q 14

1	OUI	
2	NON	

La boucle **for** dite "énumérative" permet :		Q 15
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 16

```
public class Exemple
{
   public ArrayList<String> arr;
   public Exemple(){}
   public add(String e)
   {
      arr.addElement(e);
   }
}

Dans un programme Java:
   Exemple ex = new Exemple();
   ex.add("EXEMPLE");
   Terminal.ecrireStringln(ex.arr.get(0));
```

Ce code :

1	affiche "EXEMPLE"	
2	affiche ""	
3	ne fonctionne pas correctement	

En JAVA, un tableau (tab[])		Q 17
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	

De ces 3 classes quelles sont les classes qui permettent de modifier une chaîne de caractère		Q 18
1	String	
2	StringBuffer	
3	StringTokenizer	

La classe StringTokenizer :		Q 19
1	a un constructeur dont la signature est : StringTokenizer(String str, String tokens)	
2	permet de parcourir une chaîne de caractère afin d'y extraire les sous-chaines qui sont séparées par un ou plusieurs caractères spécifiques	
3	permet de rechercher la position (indice) des tokens contenus dans la chaîne de caractère str	

En Java, il est possible de modifier le contenu d'un tableau passé en paramètre d'une méthode		Q 20
1	OUI	
2	NON	

Le code suivant permet d'augmenter la taille du tableau t1 : Q 21

```
int[] t1 = new int[10];
t1 = {1,2,3,4,5,6,7,8,9,10};

augmenterTailleTab(t1,100);
```

Avec :

```
public static void augmenterTaille(int[] t,int newTaille) {
  int[] tmp = new int[newTaille];
  for(int i=0;i<t.length;i++) tmp[i]=t[i];
  t = tmp;
}
```

1	OUI	
2	NON	

Une différence entre un tableau java [] et un ArrayList est : Q 22

1	un tableau ne peut pas contenir d'objet alors qu'un ArrayList peut contenir des objets	
2	un tableau a une capacité fixe alors qu'un ArrayList a une capacité dynamique	

En JAVA, la déclaration d'un tableau comme suit : Q 23

```
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 ont toutes à *null*	

Le compilateur Java (javac) permet de créer un exécutable qui ne s'exécute que sur le type de machine sur laquelle le programme a été compilé Q 24

1	OUI	
2	NON	

La commande **javac** C1.java Q 25

1	crée le fichier C1.class	
2	compile la classe C1 et exécute la méthode main de la classe C1	

Soit le code JAVA suivant : Q 26

```
public class C {
   public 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	

Soit la classe suivante :	Q 27

```
public class C1{
  private int nb;
  private ArrayList<String> liste;
  public C1(int nb){ this.nb = nb; }
  public void add(String s){liste.add(s);}
}
```

Le code suivant s'exécute sans erreur

```
C1 c1 = new C1(0);
c1.add("TOTO");
```

1	OUI	
2	NON	

En JAVA, le type de retour d'une méthode est soit void, soit un type primitif, soit la référence d'un objet	Q 28
1	OUI
2	NON

En JAVA, le passage des paramètres se fait :	Q 29
1	par référence
2	par valeur

Soit le code suivant : Q 30

```
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	
3	erreur d'exécution	

En JAVA, il est possible de définir deux méthodes de même nom dans la même classe	Q 31
1	OUI
2	NON

Soit le code JAVA suivant — Q 32

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

class Exemple1
{
    private int[] tab;

    public void Exemple1()
    {
        tab = new int[10];
    }
}
```

Ce code s'exécute correctement :

1	OUI
2	NON

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 33

1	OUI
2	NON

En Java augmenter la taille physique d'un attribut de type tableau est possible dans la mesure où il est toujours possible changer la valeur de l'attribut avec un nouveau tableau d'une taille supérieure — Q 34

1	OUI
2	NON

La classe StringBuffer de Java, est une variante de la classe String. Elle permet de créer des chaines de caractère dans lesquelles il est possible d'ajouter, supprimer et insérer des caractères. — Q 35

1	OUI
2	NON

(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

Expliquez le rôle et le contenu d'une classe dans la programmation objet.

Q 2

Expliquer à quoi sert la classe ArrayList<E>

Q 3

Qu'est ce qu'un *attribut d'objet* (non static) dans une classe.
Citez où dans la classe sont utilisés ces attributs.

2ème PARTIE : PROGRAMMATION (avec document)

Exercice 1 [25 points]

A terme, nous voulons réaliser un programme informatique qui gère les comptes des clients d'un supermarché (carte de fidélité).

L'objectif est de pouvoir connaitre à tout moment le montant total courant des achats d'un client (appelé solde) et le nombre de point de fidélité accumulé.
Le client met à jour son compte lorsqu'il donne sa carte de fidélité à chaque passage en caisse, c'est-à-dire, pour nous, son numéro d'identification.

Nous identifions 2 classes :
- la classe ComptesClient qui contient les comptes de chacun des clients
- la classe CarteClient qui contient l'identification du client et le montant de ses achats cumulés.

1/ Ecrire uniquement les attributs et le constructeur de chacune des classes. Nous supposons qu'il n'y a aucun client au départ.

2/ Nous voulons coder les 2 méthodes suivantes :
- la méthode qui permet de créer et ajouter un nouveau client
- la méthode qui permet de mettre à jour son compte lors de son passage en caisse. La méthode utilise la méthode suivante (cette méthode n'est pas à coder) :

> double[] traiterCaisseClient(String identificateur_client, String fichierCaisse)
> Cette méthode a en entrée un fichier caisse et l'identificateur d'un client. Elle retourne pour le client en entrée un tableau de 2 valeurs : le montant des achats et le nombre de points de fidélité accumulés (le client pourrait passer 2 fois à la même caisse dans la journée).

Ecrire l'algorithme de chacune des méthodes.
Ecrire le code JAVA de chacun des méthodes.

3/ Ecrire la méthode traiterCaisseClient définit comme suit.

Soit un fichier "caisse.txt" contenant la liste de tous les achats, article par article, réalisés par une caisse d'un supermarché.

Le fichier est structuré de la manière suivante.
Chaque ligne est de la forme :
"*numero_client numero_caisse date_achat nombre montant points_fidélité article*"

Exemple :

00012345	2	201402131731	4	12.50	2	boite de thon Anchor
00012345	2	201402131731	2	7.45	0	jambon blanc
00012345	2	201402131731	1	253.90	10	tv écran plat
00067890	2	201402131745	2	2.50	0	sopalin
00067890	2	201402131745	3	3.45	1	beurre
00067890	2	201402131745	1	53.90	3	cocotte minute

La méthode suivante de la classe Terminal vous est donnée et lit le contenu du fichier et retourne le contenu du texte sous la forme d'un tableau dont chaque élément est une ligne du fichier texte :

```
public static String[] lireFichierTexte(String nomFichier) ;
```

Nb: Vous n'avez pas à écrire le code des getteurs et des setteurs des attributs des classes

Exercice 2 [15 points]

Soit une collection ArrayList<String> dont chaque chaine de caractère est de la forme :
date;**heure**;**texte_rendez-vous**
où

date est de la forme JJ/MM/AA quand il s'agit d'un rendez-vous à la date donnée, ou
date est un jour ("lundi","mardi", ... "dimanche") quand il s'agit d'un rendez-vous hebdomadaire
date est une chaine quelconque (ex: "a faire", "urgent", ...) dans ce cas le rendezvous est valable quelque soit la date d'entrée et l'heure n'est pas renseignée.

Soit la date d'entrée au traitement **dateCourante** qui est une String de la forme "jour JJ/MM/AA" (ex: "Mercredi 19/02/2014").

Faire le traitement en JAVA qui affiche les rendez-vous de la date **dateCourante** :

```
public afficherRdv(String dateCourante,  ArrayList<String> rdvs)
```

(Fin du sujet)