IPST-CNAM
Programmation JAVA
NFA 032
Mercredi 26 Juin 2013

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

1ère Session NFA 032

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 1h 15mn, 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:

En Java, la classe Hashtable est une classe de définition d'une Collection qui permet l'accès à ses éléments via une chaine de caractères		Q 1
1	OUI	
2	NON	

En Java, la classe Collections permet de trier les éléments de n'importe quelle collection (classe qui implémente l'interface List) grâce à la méthode ***sort***. Cette méthode fonctionne si la classe d'appartenance des éléments de la collection implémente l'interface :		Q 2
1	Comparator	
2	Comparable	
3	Comparer	

En java, la notion d'interface est un moyen de conception des interfaces homme machine		Q 3
1	OUI	
2	NON	

Une classe abstraite est une classe dans laquelle toutes les méthodes sont abstraites (une méthode abstraite est une méthode sans code) et n'a pas de constructeur.		Q 4
1	OUI	
2	NON	

Soit trois classes A, B et C qui héritent de la classe abstraite H et H implémente l'interface I. Pour créer une collection polymorphe, on peut faire :		Q 5
1	ArrayList<I> elements;	
2	ArrayList<H> elements;	
3	ArrayList<A,B,C> elements	

Soit les classes A et B qui héritent de C. Les classes A, B et C ne sont pas abstraites.
Soit la classe Stock contenant l'attribut : ArrayList<C> elements;
On peut écrire :
`elements.add( new A() )`
`elements.add( new B() )`
		Q 6

1	OUI	
2	NON	

Une collection polymorphe est une collection qui peut être utilisée sous différentes formes : List, Set, Array, LinkedList, …		Q 7
1	OUI	
2	NON	

On a le code suivant :
`File fichier = new File("ListeDouble.bin");`
`FileOutputStream fos = new FileOutputStream(fichier);`
`DataOutputStream dos = new DataOutputStream(fos);`
`dos.writeInt(tab.length);`
`for(int i=0;i< tab.length;i++) dos.writeDouble( tab[i]);`
`dos.close();`
		Q 8

1	Ce code crée un fichier de nom 'ListeDouble.bin' contenant la liste de doubles	
2	Ce code crée un fichier dont les informations sont dans un format texte	
3	Ce code crée un fichier dont les informations sont dans un format binaire	

La sérialisation est un principe natif du langage JAVA permettant de plier et déplier les attributs des objets afin de pouvoir transporter les objets via un fichier ou un socket		Q 9
1	OUI	
2	NON	

Soit le code JAVA suivant : Q 10

```
ArrayList<String> liste = new ArrayList<String>();
Collections.addAll(liste,"1","3","3","1","4","5","4");
liste = new ArrayList<String>( new HashSet<String>(liste) );
```

1	La dernière ligne de ce code n'a pas de sens	
2	La dernière ligne de ce code convertit un ensemble en une liste	
3	La dernière ligne de ce code enlève de "liste" tous les éléments redondants	

En JAVA, une interface est un moyen de créer des traitements génériques		Q 11
1	OUI	
2	NON	

La déclaration de la méthode suivante : Q 12

```
public void traitement(String s) throws MyException
```

précise que la méthode doit capturer l'exception MyException dans le corps de sa méthode.

1	OUI	
2	NON	

Soit le code JAVA suivant : Q 13

```
public void traitement(String nom) throws Exception
{
   Individu ind=null;
   try {
       ind = rechercher(nom);
       System.out.println(ind.toString());
   }
   catch(NonTrouveException ex) {
      throws new Exception("non trouve"); }
}
avec :
    la     méthode     'rechercher'     retourne     l'exception
'NonTrouveException' si le nom de l'individu n'est pas trouvé.
```

Alors :

1	si l'individu que l'on veut ajouter n'est pas trouvé alors la méthode retourne l'exception NonTrouveException	
2	si l'individu que l'on veut ajouter n'est pas trouvé alors la méthode retourne l'exception Exception	
3	Si l'individu que l'on veut ajouter n'est pas trouvé alors la méthode ne retourne pas d'exception	

L'exception JAVA RuntimeException est une exception qui est retournée quand le moteur de la JVM (Runtime) plante		Q 14
1	OUI	
2	NON	

Soit le code JAVA suivant : Q 15

```
public void traitement()
{
   try{
      faire();
   }catch(Exception ex){ throws new RuntimeException("erreur"); }
```

avec la méthode "faire" qui déclenche l'exception "MyException".
Alors :

1	ce code ne se compile pas correctement	
2	la méthode "traitement" retourne une exception	
3	il manque dans l'en tête de la méthode la mention : "throws RuntimeException"	

En JAVA, l'instruction "accept" de ServerSocket tel que : Q 16

```
ServerSocket ssoc = new ServerSocket(9999);
Socket soc = ssoc.accept();
```

1	prévient le client qu'il peut envoyer ses informations au serveur	
2	se met en attente jusqu'à ce qu'un client crée un socket sur le port 9999	

Le socket est un moyen d'échanger des informations entre deux programmes qui ne sont pas écrits dans un même langage Q 17

1	OUI	
2	NON	

Le code JAVA suivant calcule la somme des N premiers nombres entiers positifs récursivement : Q 18

```
static int somme(int n)
    {
        return n + somme(n-1);
    }
```

C e code est correct :

1	OUI	
2	NON	

Le code JAVA, suivant liste les fichiers et les répertoires qui se trouvent sous le répertoire "/home/jl" Q 19

```
File varfile;
varfile = new File("/home/jl");
for(String nom : varfile.list())
   System.out.println(nom);
```

1	OUI	
2	NON	

Les classes ArrayOutputSream et ArrayInputStream permettent d'écrire et lire dans les fichiers des tableaux contenant n'importe quel objet Q 20

1	OUI	
2	NON	

En JAVA, pour faire la sauvegarde d'un ensemble de données, on utilise, notamment, la classe DataOutputStream qui permet d'écrire les attributs des objets de type élémentaire (chaine, entier, double, …) Q 21

1	OUI	
2	NON	

La class File ne gère que les fichiers. Pour gérer des répertoires on utilise la classe FileDirectory Q 22

1	OUI	
2	NON	

En JAVA, un moyen pour sérialiser les objets de nos propres classes, est que chaque classe utilisée dans la composition des objets, implémente l'interface prédéfinie 'Serializable'		Q 23
1	OUI	
2	NON	

Le code suivant permet de lire une chaine au clavier :
`BufferedReader in =`
`  new BufferedReader(new InputStreamReader(System.in));`
`String str = in.readLine();`
		Q 24

1	OUI	
2	NON	

En JAVA, créer un thread consiste à implémenter la méthode void run() dans une classe qui hérite de Thread		Q 25
1	OUI	
2	NON	

Deux threads peuvent exécuter en même temps la même méthode d'un même objet et donc, par cela, modifier en même temps les mêmes attributs de l'objet		Q 26
1	OUI	
2	NON	

Soit deux thread t1 et t2 qui utilisent la méthode définie ainsi `synchronized public void miseajour(){…` Quand t1 est en train d'exécuter cette méthode alors t2 (qui veut aussi l'exécuter) est en attente que t1 ait fini de l'exécuter		Q 27
1	OUI	
2	NON	

Dans un appel récursif seul les paramètres d'appel de la méthode récursive sont empilés sur la pile d'exécution		Q 28
1	OUI	
2	NON	

Le code JAVA suivant définit la structure d'une liste chainée : Q 29

```
class Liste
{
    private int nb;        // Nombre d'élément de la liste
    private Cellule prem;  // Premier élément de la liste
    }
class Cellule
{
    Object  value;   // Valeur de l'élément
    Cellule suiv;    // Element suivant
}
```

1	OUI	
2	NON	

La classe ArrayList est une structure de données récursive		Q 30
1	OUI	
2	NON	

Soit le code suivant : Q 31

```
try{
   System.out.println("AAA");
   call();
   System.out.println("BBB");
}
catch(MyException ex) {
   System.out.println("DDD");
}
catch(Exception ex) {
   System.out.println("CCC");
}
```

avec la méthode call qui déclenche l'exception **MyException**.

Ce code affiche :

1	AAA CCC	
2	AAA DDD CCC	
3	AAA DDD	

La méthode "start" est la méthode qui permet de démarrer un thread Q 32

1	Cette méthode doit être implémentée dans la classe qui hérite de Thread	
2	Cette méthode appartient à la classe Thread. On y accède par héritage de la classe Thread	
3	Cette méthode appelle la méthode run() que nous avons écrit dans notre thread	

Soit une collection "liste" définie par la classe ArrayList<Individu>. Nous voulons trier les éléments de cette liste suivant 3 critères de tri différents. Pour cela : Q 33

1	la classe Individu implémente l'interface Comparable et on code dans la méthode **compareTo** les 3 critères de tri. Le choix du tri se fait par la valeur d'un attribut statique	
2	la classe Individu implémente 3 fois l'interface Comparable	
3	pour chacun des tris faire les appels : Collections.sort(liste, comparator1) Collections.sort(liste, comparator2) ou Collections.sort(liste, comparator3) où comparator1, comparator2, comparator3 sont des instances des classe Comparator1, Comparator2, Comparator3 qui implémentent la méthode **compare** de l'interface Comparator<Individu>	

Soit une classe MyThread qui hérite de Thread et qui contient un attribut défini comme suit : Q 34

```
private static int tempo;
```

Tous les threads, instance de MyThread, ont tous la même valeur tempo (exemple: temps de temporisation des threads)

1	OUI	
2	NON	

Soit deux classes B et C qui héritent d'une classe abstraite A. Q 35
Les classes B et C peuvent surcharger les méthodes publiques non abstraites de la classe A.

1	OUI	
2	NON	

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

Précisez le rôle des 3 instructions JAVA suivantes :

```
... throws MyException
try{...}catch(MyException ex){...} catch(Exception ex){...}
throw new MyException()
```

Q 2

Donnez 2 exemples (principes différents) de l'utilisation d'une interface JAVA. Commentez

Q 3

A quoi servent les classes ObjectInputStream et ObjectOutputStream ?
Expliquez.

(Tourner la page)

2ème PARTIE : PROGRAMMATION (avec document)

Problème 1 [15 points]

On se propose de faire, en parallèle, deux tirages aléatoires et de compter le nombre de fois que les deux tirages sont identiques.

Faire le programme principal JAVA qui réalise le traitement suivant :
- pris en compte du paramètre du programme
- création d'un tableau de 2 entiers (tab)
- création d'un compteur
- création et exécution de deux threads (t1 et t2) qui calculent, chacun, toutes les 10 millisecondes un nombre entier aléatoire compris entre 1 et N (inclus). t1 dépose la valeur calculée en tab[0] et t2 en tab[1].
- ensuite le programme boucle et toutes les 100 millisecondes teste si tab[0] est égal à tab[1] alors il incrémente le compteur et affiche tab[0] et le compteur.

N est passé en paramètre du programme. Si N n'est pas un entier ou est supérieur à 100 alors le programme affiche un texte d'erreur et prend la valeur par défaut 10.

Les classes à créer sont :
- la classe **Exercice1** qui contient la méthode main
- la classe **AleatThread** qui permet la création des 2 threads.

Rappels :
- la méthode **Random int nextInt(int n)** retourne un nombre aléatoire entre 0 et n-1.
- la méthode **Thread.sleep(int n)** se bloque (attente) pendant n millisecondes.

Problème 2 [15 points]

Soit la classe Individu caractérisée par :
- le nom
- le prénom
- la date de naissance (Calendar)

Faire le programme JAVA complet qui réalise le traitement suivant :
- création d'une liste d'individu
- initialisation en dur de la liste avec 3 individus
- tri la liste par ordre croissant sur le nom et sur le prénom (si nom identique alors ordre sur le prénom)
- affiche la liste
- tri la liste par ordre croissant sur la date de naissance
- affiche la liste.

La date est affichée au format JJ/MM/AAAA.

Utilisez les principes de JAVA au maximum.

(Fin du sujet)