

IPST-CNAM
Programmation JAVA
NFA 032
Septembre

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

2ème 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. **N'oubliez de mettre votre nom et prénom en en tête du QCM.**

Pour la deuxième partie, vous écrivez vos programmes sur des copies vierges. Vous devez écrire les codes commentés en Java.

1^{ère} PARTIE : COURS (sans document)
Durée: 1h15

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:
-------------	----------------

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 1.
1	OUI	
2	NON	X

Une classe abstraite est une classe dont le type des attributs ne sont pas connus. C'est lors de l'instanciation de l'objet que les types des attributs sont résolus		Q 2.
1	OUI	
2	NON	X

Une classe qui contient au moins une méthode abstraite doit être déclarée abstraite.		Q 3.
1	OUI	X
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 4.
1	ArrayList<I> elements;	X
2	ArrayList<H> elements;	X
3	ArrayList<A,B,C> elements	

Soit les classes quelconques A et B qui héritent de C. Soit la classe Stock contenant l'attribut : ArrayList<C> elements; On peut écrire :		Q 5.
<pre> elements.add(new A()); elements.add(new B()); </pre>		
1	OUI	X
2	NON	

Soit le code suivant :		Q 6.
<pre> public class A extends B implements C { private int attr_A; public A() { attr_A = 10; attr_B = "TOTO"; C.attr_C = 100; } } </pre>		
1	attr_B est un attribut protected de B	X
2	attr_C est un attribut static public de l'interface C	X
3	attr_B est un attribut privé de B	

En JAVA, une classe peut implémenter plusieurs interfaces		Q 7.
1	OUI	X
2	NON	

En JAVA, une interface est un moyen de créer des traitements génériques		Q 8.
1	OUI	X
2	NON	

En Java, l' Interface permet de :		Q 9.
1	gérer des collections polymorphes (les éléments sont de type d'une interface)	X
2	gérer les fichiers (interface avec le système d'exploitation)	
3	de passer en paramètre d'une méthode "un traitement" (traitements génériques)	X

La déclaration de la méthode suivante : <pre>public void traitement(String s) throws MyException</pre> précise que la méthode doit capturer l'exception MyException dans le corps de sa méthode.		Q 10.
1	OUI	
2	NON	X

Soit le code JAVA suivant : <pre>public void traitement(String nom) throws Exception { Individu ind=null; try { ind = rechercher(nom); System.out.println(ind.toString()); } catch(NonTrouveException ex) { throw new Exception("non trouve"); } } avec : la méthode 'rechercher' qui retourne l'exception 'NonTrouveException' si le nom de l'individu n'est pas trouvé.</pre> Alors :		Q 11.
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	X
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 12.
1	OUI	
2	NON	X

En JAVA, la classe RuntimeException est une exception pour laquelle la méthode qui la déclenche n'a pas besoin de préciser qu'elle propage cette exception (l'usage de throws est inutile)		Q 13.
1	OUI	X
2	NON	

Soit le code suivant :		Q 14.
<pre> try{ System.out.println("AAA"); call(); System.out.println("BBB"); } catch(MyException ex) { System.out.println("CCC"); } catch(Exception ex) { System.out.println("DDD"); } </pre>		
avec la méthode call qui déclenche l'exception MyException .		
Ce code affiche :		
1	AAA CCC	X
2	AAA CCC DDD	
3	AAA DDD	

En JAVA, une exception est une classe qui hérite toujours directement ou indirectement de la classe Exception		Q 15.
1	OUI	X
2	NON	

En Java, la classe Hashtable est une classe de définition d'une Collection qui permet l'accès à ses éléments via une chaîne de caractères		Q 16.
1	OUI	X
2	NON	

Une collection polymorphe est une collection qui peut être utilisée sous différentes formes : List, Set, Array, LinkedList, ...		Q 17.
1	OUI	
2	NON	X

Le code JAVA suivant définit la structure d'une liste chaînée :		Q 18.
<pre> 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 } </pre>		
1	OUI	X
2	NON	

En Java, une liste chaînée (ex: classe LinkedList) est une liste dont les éléments ne sont pas continus dans la mémoire de la JVM.		Q 19.
1	OUI	X
2	NON	

En Java, une liste non chaînée (ex: ArrayList) est une liste dont les éléments sont continus dans la mémoire de la JVM.		Q 20.
1	OUI	X
2	NON	

En Java, les différences entre une liste chaînée et une liste non chaînée sont :		Q 21.
1	Il est plus rapide dans une liste chaînée d'ajouter un élément en début de la collection que dans une liste non chaînée	X
2	Il est plus rapide de parcourir tous les éléments dans une liste chaînée que dans une liste non chaînée	
3	Pour un même nombre d'élément, la liste chaînée est plus gourmande en mémoire que la liste non chaînée.	X

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 22.
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	X
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>	X

En Java, on déclare un tableau qui contient des éléments dont la classe d'appartenance implémente une interface I		Q 23.
1	Cela n'est pas possible	
2	Cela est possible et on ajoute dans le tableau des objets de type I (t[i]=new I();)	
3	Cela est possible et on ajoute dans le tableau des objets de type B (t[i]=new B();) avec la classe B qui implémente l'interface I	X

On a le code suivant :		Q 24.
<pre>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();</pre>		
1	Ce code crée un fichier de nom 'ListeDouble.bin' contenant la liste de doubles	X
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	X

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 25.
1	OUI	X
2	NON	

Le code JAVA, suivant liste les fichiers et les répertoires qui se trouvent sous le répertoire "/home/jl"		Q 26.
<pre>File varfile; varfile = new File("/home/jl"); for(String nom : varfile.list()) System.out.println(nom);</pre>		
1	OUI	X
2	NON	

Les classes <code>ArrayOutputStream</code> et <code>ArrayInputStream</code> permettent d'écrire et lire dans les fichiers des tableaux contenant n'importe quel objet		Q 27.
1	OUI	
2	NON	X

En JAVA, pour faire la sauvegarde d'un ensemble de données, on utilise, notamment, la classe <code>DataOutputStream</code> qui permet d'écrire les attributs des objets de type élémentaire (chaîne, entier, double, ...)		Q 28.
1	OUI	X
2	NON	

En JAVA, pour sérialiser un objet, il est nécessaire que la classe d'appartenance de l'objet, implémente l'interface prédéfinie 'Serializable'		Q 29.
1	OUI	X
2	NON	

Le code suivant crée un fichier de nom "exemple.txt" dans le répertoire courant d'exécution. Le fichier est créé, vide de toute information		Q 30.
<pre>File fichier; fichier = new File("exemple.txt");</pre>		
1	OUI	
2	NON	X

Le code suivant est correct :		Q 31.
<pre>import java.io.*; public class Terminal{ static BufferedReader in = new BufferedReader(new InputStreamReader(System.in)); public static String lireString() throws IOException { return in.readLine(); } }</pre>		
1	OUI	X
2	NON	

Le code JAVA suivant calcule la somme des N premiers nombres entiers positifs récursivement :		Q 32.
<pre>static int somme(int n) { return n + somme(n-1); }</pre>		
Ce code est correct :		
1	OUI	
2	NON	X

Le code suivant permet de lire une chaîne au clavier :		Q 33.
<pre>BufferedReader in = new BufferedReader(new InputStreamReader(System.in)); String str = in.readLine();</pre>		
1	OUI	X
2	NON	

Le code suivant permet de lire une chaîne au clavier :		Q 34.
<pre>String str = System.out.readLine();</pre>		
1	OUI	
2	NON	X

Soit la déclaration suivante :		Q 35.
ArrayList mes_elements;		
1	Ce code est correct : <pre>Individu ind = new Individu("LAFONT", "Pierre"); mes_elements.add(ind);</pre> car Individu hérite de Object	X
2	Ce code est correct : <pre>Individu ind = new Individu("LAFONT", "Pierre"); mes_elements.add((Object) ind);</pre> car Individu hérite de Object	X
3	Ce code est correct : <pre>Object ind = new Object("LAFONT", "Pierre"); mes_elements.add(ind);</pre> car Individu hérite de Object	

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

Un applicatif java contient une collection polymorphe qu'il est nécessaire de sauvegarder dans un fichier. Expliquez avec précision comment vous pensez faire cette sauvegarde.

Soit la classe polymorphe dont les éléments sont de type d'une interface Element contenant la méthode écrire(DataOutputStream dos).

La sauvegarde peut être faite de la manière suivante :

- on crée dans chacune des classes qui implémentent l'interface Element, la méthode écrire(dos) qui écrit chacun des attributs de l'objet dans le flot de fichier dos.
- on crée le traitement qui réalise :
 - l'ouverture du fichier en un flot DataOutputStream
 - une boucle sur tous les éléments de la collection dans laquelle on appelle la méthode écrire.

Q 2

Donner une définition précise de ce qu'est une " **exception** ".
Quels sont les mots clefs importants d'utilisation des exceptions en Java ? Donnez, en une phrase, le rôle de chacun de ces mots clefs.

En java, un exception est un objet dont la classe d'appartenance hérite directement ou indirectement de la classe Exception. En général, une exception est un cas d'erreur d'une méthode qui est remonté à l'appelant.

Les mots clefs importants sont : try catch throws et throw.

try permet d'ouvrir un bloc d'instruction dans lequel des exceptions peuvent être déclenchées ou remontées.

Le bloc try est suivi par le bloc catch qui contient la liste des exceptions capturées.

throw permet de déclencher une exception

throws permet d'indiquer pour une méthode quelles exceptions elle remonte.

Q 3

Donner une définition précise de ce qu'est une " **interface** ".
Donnez deux exemples d'utilisation d'une interface (Vous pouvez illustrer ces deux exemples avec du code Java).

En java, une interface est une déclaration abstraite d'un comportement décrit par une liste de méthodes abstraites (sans code).

Exemple 1 : l'interface peut être utilisée pour créer une collection contenant des objets de différents types. Pour cela, il faut que la collection soit du type de l'interface et que toutes les classes des objets qui sont mis dans la collection, implémentent l'interface.

Exemple 2 : l'interface peut être utilisée pour passer en paramètre d'une méthode un autre traitement. Il suffit pour cela de déclarer le paramètre sous la forme d'une interface et de passer en paramètre un objet dont la classe d'appartenance implémente l'interface.

(Tourner la page)

2^{ème} PARTIE : PROGRAMMATION (avec document)
Durée: 1h15

Problème [50 points]

S

```
public class SocieteLocation
{
    private ArrayList<Loueur> loueurs; // les loueurs de véhicule
    private ArrayList<Vehicule> vehicules; // les véhicules loués ou à louer

    public ArrayList<String> rechercherVehicule(String rech)
    {
        ArrayList<String> resultat = new ArrayList<String>();
        for(Vehicule v : vehicules)
        {
            String str = v.toStringRech();
            if (str.indexOf(rech)!=-1)
                resultat.add(v.getReference());
        }
        return resultat;
    }

    // La méthode toStringRech est une méthode qui est déclarée abstraite
    // dans Vehicule et implémentée dans chacune des classes VTourisme et
    // VTransport qui retourne chacune tous les attributs de l'objet dans
    // lesquels on veut faire la recherche, sous la forme d'une chaîne (espacés
    // par "\n")

    public String listeVehicule()
    {
        String resultat = "";
        for(Vehicule v : vehicules)
        {
            if ( (v.siLoue() &&
                (v.getDateFinLocation().compareTo(
                    Calendar.getInstance())<0) )
                resultat = resultat + v.toString();
        }
        return resultat;
    }
    // la méthode getDateFinLocation retourne la date de fin de location du
    // vehicule
    // la méthode siLoue retourne vrai si le véhicule a été loué sinon faux

    // Nous faisons le choix de sauvegarder dans un fichier binaire
    // en utilisant le principe de la sérialisation
    // Il faut que les classes Vehicule, VTourisme et VTransport
    // implémentent l'interface Serializable
    //
    public void sauvegarder()
    {
        File fic = new File("SocieteLocation.bin");
        FileOutputStream fos = new FileOutputStream(fic);
        ObjectOutputStream oos = new ObjectOutputStream(fos);

        oos.writeObject(loueurs);
        oos.writeObject(vehicules);

        oos.close();
    }
}
```

```
public void charger()
{
    File fic = new File("SocieteLocation.bin");
    FileInputStream fis = new FileInputStream(fic);
    ObjectInputStream ois = new ObjectInputStream(fis);

    loueurs = (ArrayList<Loueur>)(ois.readObject());
    vehicules = (ArrayList<Vehicule>)(ois.readObject());

    ois.close();
}
```

(Fin du sujet)