IPST-CNAM
Programmation JAVA
NFA 001
Mercredi 20 Juin 2012

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

CORRECTION

2ème Session NFA 001

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 (30 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 **3 points** négatifs sans pénalité.

NOM:	PRENOM:

Le langage JAVA n'est pas un langage orienté objet mais un langage fonctionnel de dernière génération		Q 1
1	OUI	
2	NON	X

Soit le code suivant :

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

:

		Q 2
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	X
3	L'exécution de ce programme affiche à l'écran la chaine de caractère passée en argument"	

Soit le code suivant :

```
Individu tab[] = new Individu[100];
Individu ind = new Individu();
ind.nom = "ROUSSEAU"
tab[0] = ind;
ind.nom = "VOLTAIRE";
Individu ind2 = tab[0];
Terminal.ecrireStringln(ind2.nom);
```

Ce code :

		Q 3
1	affiche VOLTAIRE	X
2	affiche ROUSSEAU	
3	est incorrect	

Soit le code suivant ::

```
public class A extends C
{
   public void faire(C c) {
      B b;
      b.x = 100;
      c.y = 200;
   }
}
```

Les classes A, B et C sont toutes des classes public.

		Q 4
1	si y est un attribut protected de C et x est un attribut public de B alors ce code est correct	X
2	si x est un attribut privé de B et y est un attribut public de C alors ce code est correct	
3	si x est un attribut public de B et y est un attribut public de C alors ce code est correct	X

Soit le code suivant : Q 5

```
public class Exemple
{
   public  String[] arr;
   private int n;
   public Exemple(){
      n=0;
      arr = new String[10];
   }
   public add(String e){
      arr[n++] = e;
   }
}

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

Ce code :

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

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	X

La commande suivante : **java MaClasse.main2 AAA 999** Q 7
exécute un programme Java dont la classe main est dans la classe "MaClasse" et dont la méthode main2 est déclarée de la manière suivante :

```
public static void main2(String arg1, int arg2)
```

1	OUI	
2	NON	X

En JAVA, un tableau Q 8

1	peut contenir des éléments de type primitif	X
3	ne peut pas contenir des références d'objet JAVA	

Soit le code suivant : Q 9

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

Pour que ce code soit correct, il faut remplacer ????? par :

1	private	
2	public	
3	static	X

La compilation d'un fichier **.java** (usage de la commande javac) Q 10

1	crée un et un seul fichier .class	
2	crée autant de fichier .class que de classes définies dans le fichier .java	X

Dans la programmation orientée objet, une classe est un concept informatique qui contient la déclaration des informations suivantes :		Q 11
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

En JAVA, un attribut static est une donnée dont la valeur est toujours constante durant l'exécution du programme JAVA		Q 12
1	OUI	
2	NON	X

Le constructeur d'une classe A :		Q 13
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

Quand la classe A hérite de B, cela signifie que :		Q 14
1	A a accès à tous les constructeurs publics de B	X
2	A a accès à tous les attributs privés de B	
3	A a accès à toutes les méthodes publics de B	X

En programmation objet, dire que la classe A "est composée de" la classe B, signifie que A contient un attribut de classe B dont la valeur est allouée (new B()) dans le constructeur de A		Q 15
1	OUI	X
2	NON	

En programmation objet, une collection polymorphe est une classe qui hérite de plusieurs classes		Q 16
1	OUI	
2	NON	X

De ces 3 classes quelles sont les classes qui permettent de modifier une chaîne de caractère		Q 17
1	String	
2	StringBuffer	X
3	StringTokenizer	

La classe StringToKenizer est une classe qui permet de convertir une chaine de caractère (String) en une Tokenizer (collection polymorphe dont chaque élément appartient à une classe qui hérite de la classe "Token")		Q 18
1	OUI	
2	NON	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 19
1	OUI	X
2	NON	

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

On a une classe A qui contient qu'un seul constructeur :
`public A(int x){ }`
 	
On a la classe B qui hérite de A, qui contient le constructeur suivant :
`public B(){`
`  attr = 100;`
`}`
		Q 21

1	Ce code est correct	
2	Ce code n'est pas correct	X

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 22
1	OUI	X
2	NON	

En JAVA, la déclaration d'un tableau comme suit : `Individu[] tab_ind = new Individu[10]` contient :		Q 23
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*	X

Dans la conception objet d'une classe, l'agrégation et la composition d'objet sont deux notions bien distinctes. Soit, la classe A contenant par agrégation un objet de classe B et contenant par composition un objet de classe C		Q 24
1	Quand on détruit un objet de classe A alors les objets de classe B et C sont également détruits	
2	Quand on détruit un objet de classe A alors seul l'objet de classe B est détruit	
3	Quand on détruit un objet de classe A alors seul l'objet de classe C est détruit	X

Dans la classe ***Collections*** la méthode ***sort*** permet de rechercher un élément dans un collection dont les éléments sont de type T. Pour cela, il est faut que la classe T implémente l'interface **Comparable** :		Q 25
1	OUI	X
2	NON	

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 26
1	OUI	
2	NON	X

Si on ne crée pas de constructeur à une classe alors il n'est pas possible de créer une instance de cette classe		Q 27
1	OUI	
2	NON	X

Le code suivant se compile-t-il correctement :
`public class Individu {`
`  private String nom;`
`  private String prenom;`
`  public void Individu(){`
`  nom=""; prenom="";`
`  }`
`}`
		Q 28

1	OUI	X
2	NON	

Soit la classe A définie ainsi :
`public class A {`
`  ArrayList<String> liste;`
`  public void ajouter(String s) {`
`  liste.add(s);}`
`}`
Le code suivant s'exécute-t-il correctement :
`a = new A();`
`a.ajouter("TOTO");`
		Q 29

1	OUI	
2	NON	X

La classe String contient la méthode : `public void replace(String str, String str_replace)` qui remplace dans l'objet de type String toutes les sous-chaine str par str_replace		Q 30
1	OUI	
2	NON	X

La classe StringBuffer est une classe prédéfinie JAVA qui gère une chaîne de caractère dans laquelle il est possible d'ajouter de nouveaux caractères		Q 31
1	OUI	X
2	NON	

En JAVA, il n'existe pas d'ordre dans la déclaration des attributs et des méthodes		Q 32
1	OUI	X
2	NON	

Quand une classe définit un constructeur avec des paramètres alors le constructeur par défaut (sans paramètre) n'est plus accessible		Q 33
1	OUI	X
2	NON	

En JAVA, il est possible de déclarer des attributs en dehors de la classe. Ils sont alors déclarés en static.		Q 34
1	OUI	
2	NON	X

Soit le code suivant :
`int tt[] = 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 doivent être remplacés par :
		Q 35

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

2. Questions libres (15 points)

Q 1

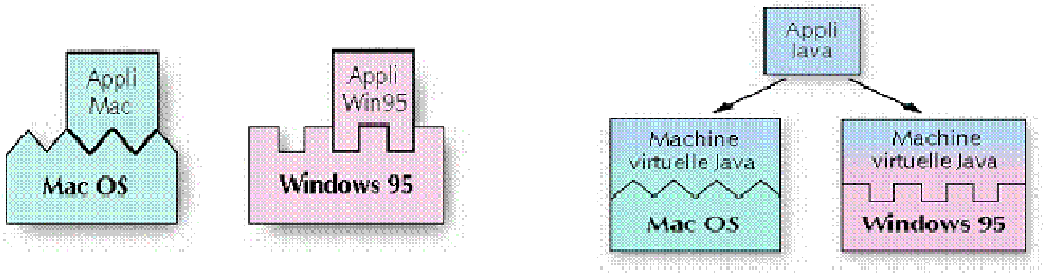
Soit une classe A qui contient une méthode m quelconque avec des paramètres.
Enumérez et précisez tous les éléments du langage Java que la méthode peut utiliser dans son corps de méthode. Expliquez.

La méthode d'une classe peut utiliser les paramètres de la méthode. Ils sont vus comme des variables et surcharge les attributs de même identifiants.
Elle utilise aussi les variables locales à la méthode qui sont déclarés dans la méthode.
Elle utilise tous les attributs de la classe d'appartenance (sauf les attributs non static si la méthode est static).
Elle utilise tous les attributs et méthodes publics des autres classes.

Q 2

Veuillez commenter le schéma suivant :

Ce schéma montre la portabilité de l'exécution d'un programme JAVA sur tous les systèmes d'exploitation car au lieu d'être directement exécuté par le système d'exploitation, l'application JAVA est interprétée par une JVM.
La JVM est par contre spécifique à chaque système d'exploitation.



Q 3

Donner une définition assez complète pour chacun des concepts suivants :
- héritage
- polymorphisme
- généricité

L'héritage est un concept fort des LOO. Une classe A hérite d'une autre classe B signifie que la classe A a accès aux méthodes et aux attributs de la classe B (suivant certaines règles de visibilité). On dit que la classe A étend les propriétés de la classe B.

Le polymorphisme est la possibilité d'une collection d'élément de contenir des éléments de type différents. Ces types correspondent à des classes qui héritent toutes d'une classe abstraite ou qui implémentent toutes la même interface.

La généricité est la possibilité d'écrire des traitements qui utilisent des méthodes dont on ne cannait pas encore le code mais dont on ne connait que la signature. On dit que le traitement est paramétré par du code (et non des données comme habituellement).

(Tourner la page)

2ème PARTIE : PROGRAMMATION (avec document)

Exercice [20 points]

```
// Correction de l'exercice
//  Cette correction utilise un tableau dont l'indicie est une
longueur de mot
//  et chaque valeur est une chaine contenant les mots espacés par 1
blanc
//
import java.util.*;
public class ExerciceCorrection2
{
    public static void main(String args[])
    {
        // Saisie de la chaine a traiter
        //
        Terminal.ecrireStringln("Entrez la chaine:");
        String chaine = Terminal.lireString();

        // Les longueurs de mot ne peuvent depasser 100
        // Tableau de liste de mots
        //
        String[] tableau = new String[100];

        // On ajoute chacun des mots de la phrase dans la liste
        //  correspondante à la longueur du mot
        //
        StringTokenizer strtok = new StringTokenizer(chaine," ");
        while(strtok.hasMoreTokens())
            {
                String str = strtok.nextToken();
                if (tableau[str.length()]==null)
                    tableau[str.length()]="";
                tableau[str.length()]=tableau[str.length()]+str+" ";
            }

        // On affiche toutes les listes de mots non vides en triant
        //  chaque liste
        //
        for(int i=0;i<100;i++)
            {
                if (tableau[i]!=null)
                    {
                        String[] motsTab = tableau[i].split(" ");
                        ArrayList<String>      mots      =      new
ArrayList<String>();
                        for(String s:motsTab) mots.add(s);
                        Collections.sort(mots);
                        for(String s:mots)
                            Terminal.ecrireString(s+" ");
                        Terminal.ecrireStringln("");
                    }
            }

    }
}
```

Probleme [30 points]

```
// Classe de définition d'un répertoire téléphonique
//
class Repertoire
{
```

```
    // Le répertoire est une litse de contacts
    ArrayList<Contact> contacts;

    // Constructeur pour créé la liste vide
    public Repertoire()
    {
        contacts = new ArrayList<Contact>();
    }

    // Ajouter un nouveau contact
    public void ajouterContact(String nom, String intitule, String
numero)
    {
        Contact c = new Contact(nom);
        c.addNumero(intitule,numero);
        contacts.add(c);
    }

    // Ajouter un nouveau numero a un contact existant
    public void ajouterNumero(String nom, String intitule, String numero)
    {
        Contact c = rechercherContact(nom);
        if (c==null)
            System.out.println("Le contat " + nom + " n'existe pas");
        else
            c.addNumero(intitule,numero);
    }

    // Modifier un de snumeros d'un contact
    public void modifierNumero(String nom, String intitule, String
numero)
    {
        Contact c = rechercherContact(nom);
        if (c==null)
            System.out.println("Le contat " + nom + " n'existe pas");
        else
            c.modifierNumero(intitule,numero);
    }

    // Recherhce un contact en fonction de son nom
    public Contact rechercherContact(String nom)
    {
        for(Contact c:contacts)
            if (c.nom.equals(nom)) return(c);
        return(null);
    }

    // Retourne tous les numeros d'un contact donné
    public String[] getNumeros(String nom)
    {
        Contact c = rechercherContact(nom);
        if (c==null)
        {
            System.out.println("Le contat " + nom + " n'existe pas");
            return null;
        }
        else
        {
            String[] tab = new String[c.numeros.size()];
            return c.numeros.toArray(tab);
        }
    }

    // Affiche tous les contacts par ordre alphabetique
    public void afficher()
    {
        Collections.sort(contacts);
        for(Contact c:contacts)
            c.afficher();
    }
```

```
}

// Classe de définition d'un contact
//
class Contact implements Comparable<Contact>
{
    // Le nom du contact
    String nom;

    // Les numeros du contact : liste de chaine
    ArrayList<String> numeros = new ArrayList<String>();

    // Constructeur qui crée un contact avec une liste vide de numéro
    public Contact(String nom)
    {
        this.nom=nom;
        numeros = new ArrayList<String>();
    }

    // Ajoute un nouveau numero
    public void addNumero(String intitule,String numero)
    {
        numeros.add(intitule+"/"+numero);
    }

    // Modifie un des numeros en fonction de son intitulé
    public void modifierNumero(String intitule, String numero)
    {
        for(int i=0;i<numeros.size();i++)
            {
                String num = numeros.get(i);
                String[] t = num.split("/");
                if (t[0].equals(intitule))
                    {
                        numeros.set(i,intitule+"/"+numero);
                        break;
                    }
            }
    }

    // Methode d'implémentation de Comparable
    // Utilisé pour trier.
    // L'ordre de tri est le nom du contact
    public int compareTo(Contact c)
    {
        return this.nom.compareTo(c.nom);
    }

    // Affiche un contact
    public void afficher()
    {
        System.out.println(nom);
        for(String num:numeros)
            System.out.println("   " + num);
        System.out.println("----------------------");
    }

}
```

(Fin du sujet)