IPST-CNAM
Programmation JAVA
NFA 001
Mercredi 29 Février 2012

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

1ère Session NFA 001

CORRECTION

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 est-il un langage orienté objet ?		Q 1
1	OUI	**X**
2	NON	

L'en-tête de déclaration d'une méthode **main** d'une classe JAVA est:		Q 2
1	static public void main(String args[])	**X**
2	static public void main(int nb_args, String args[])	
3	static private void main(String args[])	

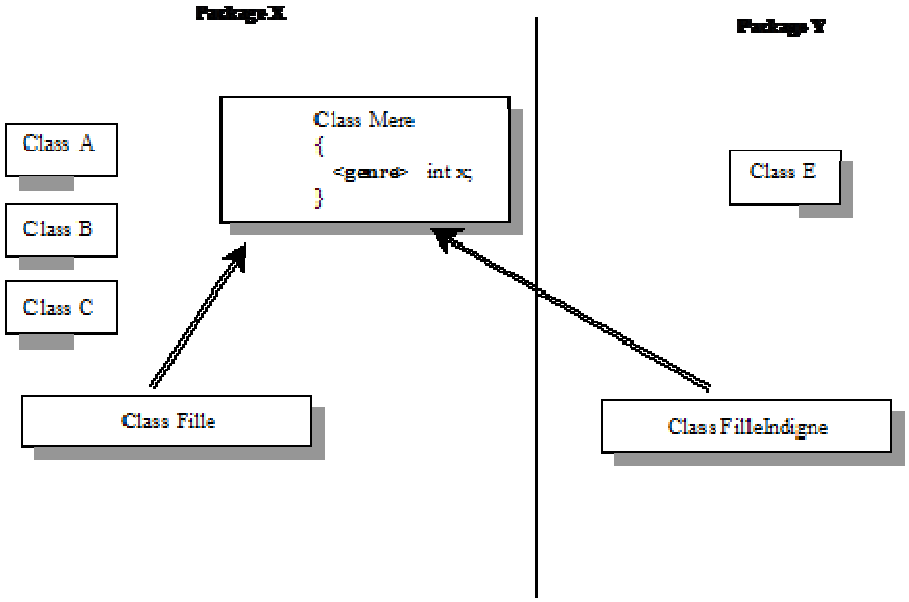
Soit le code suivant : Q 3

```
Individu tab[] = new Individu[100];
Individu ind = new Individu();
ind.nom = "DUPONT"
tab[0] = ind;
tab[0].nom = "LAFONT"
Terminal.ecrireStringln(ind.nom);
```

Ce code affiche:

1	DUPONT	
2	LAFONT	**X**

Q 4



Compléter le tableau de visibilité suivant (0,25 point par réponse juste

si <genre> a la valeur suivante :	**alors x est visible dans les classes :**
private	Mere
public	Toutes
protected	Mere, Fille, FilleIndigne, A, B, C
indéfini (vide)	Mere, Fille, A, B, C

Soit le code suivant : Q 5

```
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	**X**

La commande **javac**		Q 6
1	crée un fichier .exe exécutable par le système d'exploitation via la JVM Java	
2	crée un fichier .class qui sera ensuite interprété par la machine virtuelle JAVA	**X**

La commande **java**		Q 7
1	prend en entrée un fichier .java afin de l'interpréter	
2	prend en entrée un fichier .class afin de l'interpréter	**X**
3	exécute la méthode main de la classe java contenue dans le fichier .class qui est en entrée de la commande	**X**

En JAVA, un tableau		Q 8
1	peut contenir des éléments de type primitif	**X**
2	peut contenir des références d'objet JAVA	**X**
3	ne peut pas contenir des références d'objet JAVA	

Le langage JAVA est **portable** sur la plupart des plateformes (windows,unix,linux,....)		Q 9
1	OUI	**X**
2	NON	

La compilation d'un fichier **.java** (usage de la commande javac)		Q 10
1	crée un et un seul fichier .class	
2	crée un fichier .jar	
3	crée autant de fichier .class que de classes définies dans le fichier .java	**X**

Dans la programmation orientée objet, un objet est :		Q 11
1	une instance d'une classe créée par exemple dans le programme principal	**X**
2	une donnée structurée en mémoire de l'ordinateur	**X**
3	un processus informatique qui s'exécute indépendamment du programme principal	

Une méthode statique est une méthode dont le contenu (le code) reste inchangé durant tout le temps d'exécution du programme Java		Q 12
1	OUI	
2	NON	**X**

Le constructeur dans une classe d'objet permet :		Q 13
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	**X**
3	d'initialiser les attributs en fonction des paramètres du constructeur	**X**

Dans la classe String la méthode d'objet ***void replace(String replaced, String new)*** permet de remplacer dans l'objet String toutes les occurrences de la chaîne *replaced* par la chaîne *new*.		Q 14
1	OUI	
2	NON	**X**

En JAVA, les méthodes déclarées en dehors d'une classe sont appelées des méthodes statiques		Q 15
1	OUI	
2	NON	**X**

Une méthode publique d'une classe peut utiliser les attributs privés de la classe		Q 16
1	OUI	**X**
2	NON	

La classe StringBuffer de Java permet de modifier une chaîne de caractère directement (exemple : méthode *insert* pour insérer une sous-chaîne)		Q 17
1	OUI	**X**
2	NON	

La classe StringTokenizer :		Q 18
1	a un constructeur dont la signature est : void StringTokenizer(String str)	**X**
2	permet de parcourir une chaîne de caractère afin d'y extraire les sous-chaines qui sont séparées par un ou plusieurs caractères spécifiques	**X**
3	permet de rechercher la position des tokens (ou caractères séparateurs) contenus dans une chaîne de caractère	

Le code suivant :

```
String slue = "un::deux trois,:quatre    cinq///six:sept huit neuf";
StringTokenizer str = new StringTokenizer(slue," ,:/");
while (str.hasMoreTokens())
{
   String s = str.nextToken();
   Terminal.ecrireString (s+"_");
}
```

affiche :

Q 19

1	un_deux_trois_quatre_cinq_six_sept_huit_neuf_	**X**
2	un_ _deux_trois_ _ quatre_ _ _ cinq_ _ _ _ six_sept_huit_neuf_	
3	un:deux_trois,quatre_cinq/six:sept_huit_neuf_	

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

Dans les langages orientés objet, le polymorphisme est la capacité pour un objet de se transformer en n'importe quel autre type d'objet		Q 21
1	OUI	
2	NON	**X**

En JAVA, une classe peut hériter de plusieurs autres classes		Q 22
1	OUI	
2	NON	**X**

En JAVA, l'interface est :		Q 23
1	une méthode permettant de transformer et échanger des données entre deux classes JAVA	
2	un mécanisme JAVA permettant d'écrire des traitements génériques	**X**
3	un moyen de décrire l'interface graphique d'un programme JAVA	

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 ***ArrayList<Individu>*** la méthode ***indexOf*** permet de rechercher un individu dans la collection. Pour cela, il est indispensable que le traitement de comparaison entre l'élément recherché et les éléments de la collection soit implémenté dans :		Q 25
1	la méthode ***equals*** de la classe Individu : boolean equals(Object obj)	**X**
2	la méthode ***equals*** de la classe Individu : boolean equals(Individu ind)	

En JAVA, un objet est un pointeur		Q 26
1	OUI	**X**
2	NON	

Quelque soit le cas de figure, il est possible d'utiliser le constructeur par défaut d'une classe (Le constructeur par défaut est exécuté par l'instruction ***Classe* obj = new *Classe* ( );**)		Q 27
1	OUI	
2	NON	**X**

La difficulté dans la programmation objet est la maitrise de la destruction des objets qui doit être faite dans l'ordre inverse de leurs constructions.		Q 28
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.	**X**

Il faut initialiser les attributs d'un objet dans le(s) constructeur(s) car le langage JAVA n'initialise pas par défaut les attributs d'un objet		Q 29
1	OUI	
2	NON	**X**

Si une classe B hérite d'une classe A alors :		Q 30
1	lors de la création d'un objet de type B, les attributs privés et les attributs publiques de A sont alloués en mémoire et initialisés par le(s) constructeur(s) hérité(s) de A	**X**
2	B hérite des méthodes privées de A	

La boucle **for** dite "énumérative" permet :		Q 31
1	d'incrémenter une valeur scalaire de type enum et de réaliser un traitement à chaque valeur	
2	créer une boucle permettant de faire évoluer un indice qui sert d'accès à une collection	
3	de parcourir tous les éléments de n'importe quelle collection	**X**

Un constructeur d'une classe C est une méthode de la classe dont la forme de déclaration est : public C C(*paramètres*). Exemple d'utilisation : C x = new C(12,"xx")		Q 32
1	OUI	
2	NON	**X**

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	

Soit le code suivant :
`int[] tab_int;`
`tab_int[0]=12;`
`tab_int[1]=3;`
 	
Ce code s'exécute correctement.
		Q 34

1	OUI	
2	NON	**X**

Soit le code suivant :
`int tab_int[] = 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** → `tab_int.size()` **B** → `tab_int.get(i)`	
2	**A** → `10` **B** → `i`	
3	**A** → `tab_int.length` **B** → `tab_int[i]`	**X**

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

Donner une définition de ce qu'est un **objet** dans un langage orienté objet. Vous pouvez prendre en exemple le langage JAVA. (Soyez précis dans vos explications et vos affirmations).

Un objet est un élément du langage dont la valeur est une instance d'une classe.
Un objet contient des données (attributs) et des traitements (méthodes). Les attributs de l'objet sont initialisés par le(s) constructeur(s) de la classe. Les méthodes sont créées lors du chargement de la classe.

En Java, un objet est un pointeur vers une zone mémoire contenant les attributs et la référence de la classe d'appartenance de l'objet contenant les méthodes.

Exemple de création : MaClasse obj = new MaClasse();
L'instruction *new* crée l'objet *obj*.

Q 2

Expliquer pourquoi, en JAVA, alors que le passage de paramètre se fait par valeur, un objet passé en paramètre peut être modifié (modification de ses attributs).

En JAVA, tous les objets sont des pointeurs.
Lors du passage en paramètre d'un objet dans une méthode, c'est le pointeur qui est passé par valeur. On ne peut pas changer la valeur de ce pointeur mais on peut changer le contenu de ce que le pointeur pointe. Il est donc possible dans la méthode de changer les attributs de l'objet.

Q 3

Décrivez toutes les étapes indispensables qu'il est nécessaire de faire pour écrire un programme Java puis de l'exécuter (moyens, commandes, code).

1/ Créer un fichier texte avec l'extension .java contenant la classe Java. On peut écrire autant de fichier texte que de classes différentes;
2/ Créer dans une des classes une méthode main dans laquelle on écrit le programme principal.
3/ On compile le fichier contenant cette classe via la commande javac. Exemple: javac MonProgramme.java
4/ S'il n'y a aucune erreur de compilation alors on exécute le programme via la commande javac qui prend en entrée le nom de la classe contenant la méthode main. Exemple : java MonProgramme

(Tourner la page)

2ème PARTIE : PROGRAMMATION (avec document)

Exercice [15 points]

```
public class Exercice
{
    public static void main(String args[])
    {
        //
        while(true)
            {
                // Saisie de la valeur à tester
                Terminal.ecrireString("N= ");
                int n = Terminal.lireInt();
                // Si 0 alors on sort et le programme s'arrête
                if (n==0) break;

                // test si premier
                //
                if (Math.siPremier(n))
                    Terminal.ecrireStringln("Est premier");
                else
                    Terminal.ecrireStringln("N'est pas premier");
            }
    }
}

public class Math
{
    // Méthode statique qui teste si un entier est premier
    //
    static boolean siPremier(int n)
    {
        // On essaye tous les entiers antérieurs et on retourne false
        //  si on rencontre un diviseur
        //
        for(int i=2;i<n;i++)
            if (n%i == 0) return(false);

        return(true);
    }
}
```

Probleme [35 points]

1/

```
public class Probleme
{
    // Programme principal
    public static void main(String args[])
    {
        // Création de l'agende
        Agenda monAgenda = new Agenda();

        // Création de plusieurs rendez-vous
        //
        RendezVous rdv1 = new RendezVous("12/01/2012 08:30",
                                         "12/01/2012 10:30",
                                         "Salle Wilson",
```

```
                                        "Reunion avancement");

        RendezVous rdv2 = new RendezVous("2/01/2012 08:30",
                                        "2/01/2012 10:30",
                                        "Salle Michel",
                                        "Points AIV");

        RendezVous rdv3 = new RendezVous("23/02/2012 08:30",
                                        "29/02/2012 18:30",
                                        "",
                                        "Congés");

        // Ajout des rendez-vous dans l'agenda
        //
        monAgenda.add(rdv1);
        monAgenda.add(rdv2);
        monAgenda.add(rdv3);

        // Affichage des rendez-vous de l'agenda
        //
        for(RendezVous r:monAgenda.rdvs)
            Terminal.ecrireStringln(r+"\n========================");

    }
}

// Classe de définition d'un agenda
public class Agenda
{
    // La liste des rendez-vous
    public ArrayList<RendezVous> rdvs;

    // Le constructeur pour initialiser la liste
    public Agenda(){
        this.rdvs = new ArrayList<RendezVous>();
    }

    // Ajoute un rendez-vous dans l'agenda
    public void add(RendezVous rdv) {
        this.rdvs.add(rdv);
        Collections.sort(this.rdvs);  // Pour la réponse à 2/
    }
}

// Classe de définition d'un rendez-vous
//
public class RendezVous implements Comparable
{
    // Les attributs d'un rendez-vous
    public String dateDebut;
    public String dateFin;
    public String lieu;
    public String objet;

    // Le constructeur qui crée un rendez-vous à partir
    //  de toutes les caractéristiques
    //
    public RendezVous(String dateDebut,
                      String dateFin,
                      String lieu,
                      String objet)
```

```
      {
          this.dateDebut = dateDebut;
          this.dateFin   = dateFin;
          this.lieu      = lieu;
          this.objet     = objet;
      }

      // Pour afficher un rendez-vous dans la console
      //
      public String toString()
      {
          return( this.dateDebut + "\n" +
                  this.dateFin + "\n" +
                  this.lieu + "\n" +
                  this.objet );
      }

  }
```

2/

Il faut que la classe RendezVous implémente l'interface Comparable :

```
  public class RendezVous implements Comparable
```

Il faut implémenter la méthode compareTo qui sera utilisé dans la méthode *sort* de tri

```
      //
      public int compareTo(Object o)
      {
          RendezVous rdv = (RendezVous)o;
          if (getTime()>rdv.getTime()) return(1);
          else if (getTime()<rdv.getTime()) return(-1);
          else return(0);
      }
```

On trie les rendez-vous à chaque fois que l'on ajoute un nouveau rendez-vous :

```
  // Ajoute un rendez-vous dans l'agenda
      public void add(RendezVous rdv) {
          this.rdvs.add(rdv);
          Collections.sort(this.rdvs);  // Pour la réponse à 2/
      }
```

3/

```
      // Méthode qui retourne en millisec la date de début du rdv
      //
      public long getTime()
      {
          String[] ts  = this.dateDebut.split(" ");
          String[] tsd = ts[0].split("/");
          String[] tst = ts[1].split(":");
          Calendar cal = Calendar.getInstance();
          cal.set(Integer.parseInt(tsd[2]),
                  Integer.parseInt(tsd[1]),
                  Integer.parseInt(tsd[0]),
                  Integer.parseInt(tst[0]),
                  Integer.parseInt(tst[1]),
                  0);
          return(cal.getTimeInMillis());
      }
```