IPST-CNAM
Programmation JAVA
NFA 001
Mercredi 4 Février 2015

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

1ère Session NFA 031

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 (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 un langage orienté objet, un principe fort est que les attributs (ou données) non statiques sont alloués dans une instance d'une classe appelée un objet. Ainsi,		Q 1.
1	les données d'une instance peuvent être directement (sans passer par une méthode) modifiées par toutes les autres instances du programme	
2	Une instance peut "protégés" ses données en ne les rendant pas directement accessibles par les autres instances du programme	X
3	les données du programme sont réparties dans toutes les instances du programme	X

Dans un langage orienté objet, un objet est :		Q 2.
1	une structure de données valuées qui répond à un ensemble de « messages » (ou « méthodes »)	X
2	une vision virtuelle d'un paquet d'information qui est extérieur au programme	

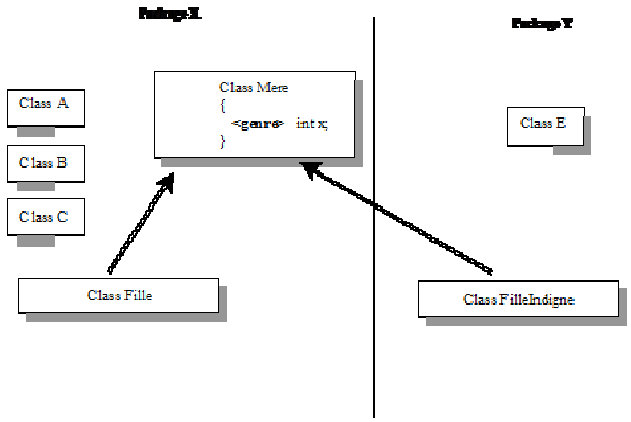
Un avantage du langage Java est que le code généré contenu dans les fichiers .class est exécutable par différents Système d'exploitation (PC, Mac, Linux)		Q 3.
1	OUI	X
2	NON	

Soit l'arborescence d'un programme java suivante : projetNFA031 bin src fr/cnam/projet/Projet.java (main) fr/cnam/ihm/Formulaire.java compil.bat run.bat Le fichier de compilation compil.bat est exécuté dans le répertoire projetNFA031. Les fichiers compilés sont générés dans le répertoire bin Le fichier compil.bat peut être :		Q 4.
1	javac –d bin –classpath "." fr/cnam/projet/Projet.java	
2	javac –d bin –classpath "src" src/fr/cnam/projet/Projet.java	X
3	cd src javac –classpath "." -d ../bin fr/cnam/projet/Projet.java	X

Soit l'arborescence d'un programme java suivante : projetNFA031 src fr/cnam/projet/Projet.java (main) fr/cnam/ihm/Formulaire.java La classe Projet utilise la classe Formulaire. Le fichier Formulaire.java contient la ligne : package fr.cnam.ihm; Le fichier Projet.java doit contenir la ligne d'import suivante :		Q 5.
1	import Formulaire;	
2	import fr.cnam.ihm.*;	X
3	import fr.cnam.ihm.Formulaire;	X

Pour exécuter un programme Java, on exécute la commande suivante : **java Programme**		Q 6.
1	dans ce cas, la JVM exécute le constructeur par défaut de la classe Programme	
2	dans ce cas, la JVM exécute la méthode main de la classe Programme	X

En JAVA, un répertoire qui contient des fichiers .java est nécessairement un package		Q 7.
1	OUI	
2	NON	



Si <genre> est private alors :		Q 8.
1	Seules les classes Fille, FilleIndigne et Mere peuvent accéder à l'attribut x	
2	Seules les classes A, B, Fille et Mere peuvent accéder à l'attribut x	
3	Seule la classe Mere peut accéder à l'attribut x	X

Soit la classe C1 et la classe C2 dont tous les attributs sont privés. Les deux classes C1 et C2 appartiennent au même package. Dans ce cas, les méthodes de C1 peuvent accéder directement aux attributs d'une instance de C2		Q 9.
1	OUI	
2	NON	X

Une méthode public d'une classe A peut utiliser directement les attributs publics de la classe B		Q 10.
1	OUI	X
2	NON	

Soit la classe C1 dont tous les attributs sont publics et statiques. Soit la classe C2 appartenant à un autre package que C1. Dans ce cas, les méthodes de C2 peuvent accéder aux attributs de C1		Q 11.
1	OUI	X
2	NON	

Dans une classe JAVA, le constructeur d'une classe permet de :		Q 12.
1	intialiser les attributs de la classe	X
2	d'allouer en mémoire les attributs d'objet de la classe	X
3	de démarrer le programme dès l'appel du constructeur	

Le code suivant est un exemple correct d'un constructeur : Q 13.

```
public class Individu
{
  private String nom;
  private String prenom;
  private int age;

  public Individu Individu(String nom, String prenom, int age)
  {
    this.nom=nom;  this.prenom=prenom;  this.age = age;
    return this;
  }
}
```

1	OUI	
2	NON	X

Soit le code JAVA suivant Q 14.

```
public class Constructeur {
    public static void main(String args[])
    {
        Exemple1 ex = new Exemple1();
        ex.tab[10] = 22;
    }
}

class Exemple1
{
    public int[] tab;

    public void Exemple1()
    {
        tab = new int[5];
    }
}
```

Ce code s'exécute correctement :		
1	OUI	
2	NON	X

Soit le code suivant : Q 15.

```
ArrayList<Individu> tab = new ArrayList<Individu>();
Individu ind = new Individu("LAFONT","Pierre");

tab.add(ind);
tab.add(ind);
System.out.println("Nombre : ", tab.size());
```

L'exécution de ce code :		
1	déclenche une erreur car deux objets identiques (de même adresse mémoire) ne peuvent être ajoutés dans un même tableau	
2	affiche: 1	
3	affiche : 2	X

Soit une classe C qui contient qu'un seul constructeur public C(int x) Il est possible d'écrire C c = new C() qui créée un objet		Q 16.
1	OUI	
2	NON	X

Soit le code suivant : Q 17.

```
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";
livres.add(l);
System.out.println(livres.get(0).nom);
System.out.println(livres.get(1).nom);
```

Ce code affiche :

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

Soit le code suivant qui teste si v est premier ou non: Q 18.

```
int v=4;
boolean premier=true;
for(int k=2;k<v;k=k+1)
   if (v%k == 0) premier = false;
   else premier = true;
if (premier) System.out.println("PREMIER");
else System.out.println("NON PREMIER");
```

Ce code :

2	affiche "NON PREMIER"	
3	affiche "PREMIER"	X

Soit le code suivant : Q 19.

```
int i=0;;
int[] tab = new int[3];
for(int v : tab)
{
   tab[i] = v*10;
   i = i+1;
}
for(int j=0;j<tab.length;j++)
   System.out.print(tab[j]+" ");
```

Ce code affiche :

1	0 10 100	
2	0 0 0	X

En JAVA, le type de retour d'une méthode est toujours la référence d'un objet ou void		Q 20.
1	OUI	
2	NON	X

En JAVA, pour qu'un objet puisse être passé en paramètre d'une méthode, il faut qu'il soit différent de null		Q 21.
1	OUI	
2	NON	X

Le code suivant permet d'augmenter la taille du tableau t1 : Q 22.

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

t1 = augmenterTailleTab(t1,100);
```

Avec :

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

1	OUI	X
2	NON	

Il est possible de modifier les caractères de la chaîne de caractère contenu dans un objet de classe String		Q 23.
1	OUI	
2	NON	X

Soit le code suivant : Q 24.

```
String str1[] = new StringTokenizer("AA;BB;CC),";");
for(String s: str1)
   System.out.println(s);
```

Ce code ::

1	affiche : AA BB CC	
2	n'affiche rien	
3	ne se compile pas correctement	X

La classe StringTokenizer :		Q 25.
1	a un constructeur dont la signature est : StringTokenizer(String str, String tokens)	X
2	a un constructeur dont la signature est : StringTokenizer()	
3	a un constructeur dont la signature est : StringTokenizer(String str)	X

Soit le code JAVA suivant : Q 26.

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

Ce code :affiche :
un_deux_trois_quatre_cinq_six_sept_huit_

1	OUI	X
2	NON	

Dans la classe String la méthode d'objet ***void set(int i, char c)*** change le i-ième caractère d'une chaine de caractère avec la valeur c.		Q 27.
1	OUI	
2	NON	X

Entre autre, une différence entre un tableau java [] et un ArrayList est :		Q 28.
1	il n'y a pas de limite de capacité à mettre des éléments dans un ArrayList	X
2	il n'y a pas de limite de capacité à mettre des éléments dans un tableau	

Soit le code JAVA suivant : Q 29.

```
ArrayList<String> tab = new ArrayList<String>(4);
tab.add("UN"); tab.add("DEUX");
for(String s : tab){System.out.println(s);}
```

1	Ce code n'est pas correcte (erreur de compilation)	
2	Ce code affiche : UN DEUX	X
3	Ce code affiche : UN DEUX null null	

En JAVA, la déclaration d'un tableau comme suit : Q 30.

```
Individu[] tab_ind = new Individu[10]
```

contient :

1	10 éléments dont les valeurs sont toutes à *null*	X
2	10 éléments dont les valeurs sont égales au résultat de l'exécution du constructeur *Individu()*	

Les tableaux java [] ne peut contenir que des données de type primitif (int, double, ...)		Q 31.
1	OUI	
2	NON	X

En JAVA, les tableaux de dimension [N] sont indicés de :		Q 32.
1	1 à N	
2	0 à N-1	X

Dans la classe String la méthode d'objet ***void replace(String replaced, String new)*** permet de remplacer dans l'objet String passé en paramètre toutes les occurrences de la chaîne ***replaced*** par la chaîne ***new***. Q 33.

1	OUI	
2	NON	X

Soit le code suivant : Q 34.

```
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 peuvent être remplacés par :

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

Soit le code suivant : Q 35.

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

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

Ce code :

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

(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

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

Dans un L.O.O, un objet est une structure de donnée dans laquelle sont également définis les traitements. On appelle ces données des attributs. On appelle ces traitements des méthodes.
Le cycle de vie d'un objet est le suivant :
- un objet est créé par instanciation d'une classe
- les attributs sont initialisés par le constructeur
- les méthodes de l'objet sont appelés par d'autres objet. Ces méthodes modifient ou retournent la valeur de ces attributs.

Q 2

Soit une classe A qui contient une méthode m (public ou privée) quelconque qui a des paramètres.
Enumérez et précisez les éléments du langage Java que la méthode peut utiliser dans son corps de méthode.

Une méthode de la classe A utilise, dans son corps, les éléments suivants :
- les paramètres de la méthode
- les variables locales de la méthode
- les attributs statiques de la classe A
- les attributs non statiques de l'objet si la méthode n'est pas statique
- les méthodes de la classe A
- les attributs des autres objets du programme (soit directement soit à travers des méthodes publiques)
- les méthodes des autres objets du programme
- les attributs et les méthodes statiques des autres classes

Q 3

Citez les 3 différences importantes qu'il existe entre l'utilisation d'un tableau Java [] et d'un ArrayList<>

1 /
Quand on crée un tableau il est indispensable de préciser sa taille physique à priori. Ainsi, il est possible de faire déborder un tableau. Alors que pour le ArrayList on ne précise pas de taille et on peut ajouter autant d'élément que nécessaire

2/ Dans un tableau il est possible de mettre tous les type Java : type primitif et type référence (objet) alors que le ArrayList ne peut contenir que des types référence

3/
L'accès au i-ème élément d'un tableau se fait par la notation [] tab[i], alors que pour les ArrayList il faut passer per l'utilisation d'une méthode tab.get(i)

Fin de la 1ère partie

2ème PARTIE : PROGRAMMATION (avec document)

Exercice [10 points]

```
// Cette methode statique prend en entree un ArrayList<RendezVous> et qui
//  retourne
//  tous les rendez-vous qui sont en intersection horaire.
// Les rendez-vous en entree ont tous la meme date.
//
public static ArrayList<RendezVous> calculerIntersection(
                                    ArrayList<RendezVous> rdvs,
                                    String heureDebut,String heureFin)
{
    ArrayList<RendezVous> rdvtmp = new ArrayList<RendezVous>();

    int debutI = heure(heureDebut);
    int finI   = heure(heureFin);

    for(RendezVous rdv : rdvs)
        {
            int debut = heure(rdv.getHeureDebut());
            int fin   = heure(rdv.getHeureFin());
            if ( ( (debut>debutI) && (debut<finI)) ||
                 ( (fin>debutI) && (fin<finI) ) )
                rdvtmp.add(rdv);
        }
    return rdvtmp;
}

// Methode qui convertit en minutes une heure ecrite en "HH:mm"
//
public static int heure(String heure)
{
    if (heure.equals("")) return 0;
    String[] elts = heure.split(":");
    int h = Integer.parseInt(elts[0]);
    int mn = Integer.parseInt(elts[1]);
    return h*60+mn;
}
```

Probleme [25 points]

```
public class Agenda
{
    /* Les rendez-vous sont stockes dans un tableau dont chaque element
       contient les rendez-vous d'une semaine (numero de semaine) */
    private RendezVousSemaine[] lesRendezVous;

    // Constructeur
    public Agenda(String nomFichier)
    {
        // Creation de tableau des semaines
        lesRendezVous = new RendezVousSemaine[54];

        // On recupere tous les rendez-vous du fichier
        ArrayList<RendezVous> tousLesRdv = lireFichierAgenda("data/Agenda.txt");

        // On parcours tous les rendez-vous lus depuis le fichier
        //  et on les stocke par semaine
        //
        for(RendezVous rdv : tousLesRdv)
            {
                // Calcul du numero de semaine de la date du rendez-vous
                String datestr = rdv.getDate();
                Calendar date = dateStringToCalendar(datestr);
                int numeroSemaine = date.get(Calendar.WEEK_OF_YEAR);

                // On cree la liste des rdv par semain si pas deje cree
                if (lesRendezVous[numeroSemaine]==null)
                    lesRendezVous[numeroSemaine]=new RendezVousSemaine(
                                                          numeroSemaine);

                // Ajout du rdv dans la liste des rdv par semaine
                lesRendezVous[numeroSemaine].ajouterRdv(rdv);
            }
    }

    // Methode qui retourne sous la forme d'une chaine de caractère
    //  tous les rendez-vous de l'agenda qui appartiennent
    //  à la date passe en parametre
    //
    public String getDateRendezVous(String dateRdv)
    {
        String res="";

        // Calcul du numero de semaine de la date en entree
        Calendar date = dateStringToCalendar(dateRdv);
        int numeroSemaine = date.get(Calendar.WEEK_OF_YEAR);

        // On recupere la liste des rendez-vous de la semaine
        RendezVousSemaine rdvsem = lesRendezVous[numeroSemaine];
        ArrayList<RendezVous> rdvs = rdvsem.getRdvs();

        // On parcours la liste des rendez-vous de la semaine et on
        //  et on ne concatene que les rndez-vous de la date en entree
        for(RendezVous rdv : rdvs)
            if (rdv.getDate().equals(dateRdv))
                res= res+ rdv.toString()+"\n";
        return res;
    }

    // Methode qui retourne sous la forme d'une chaine de caractère
    //  tous les rendez-vous de l'agenda qui appartiennent
    //  à un numero de semaine donnée
    //
    public String getDateRendezVous(int numeroSemaine)
    {
        String res="";
```



```
        // On recupere la liste des rdv de la semaine
        RendezVousSemaine rdvsem = lesRendezVous[numeroSemaine];
        if (rdvsem==null) return "Pas de rdv pour cette semaine";
        ArrayList<RendezVous> rdvs = rdvsem.getRdvs();

        // Concatenation de tous les rdv de la semaine
        for(RendezVous rdv : rdvs)
            res= res+ rdv.toString()+"\n";

        return res;
    }
```

```
// La définition de la classe qui contient les rendez-vous d'une semaine
//
public class RendezVousSemaine
{
    private int numeroSemaine;   // Numero de la dite semaine
    private ArrayList<RendezVous>  rdvs;  // Les rdvs de la semaine

    // Constructeur
    public RendezVousSemaine(int numeroSemaine)
    {
        this.numeroSemaine=numeroSemaine;
        rdvs = new ArrayList<RendezVous>();
    }

    public void ajouterRdv(RendezVous rdv){rdvs.add(rdv);}
    public ArrayList<RendezVous> getRdvs(){return rdvs;}
    public int getNumeroSemaine(){return numeroSemaine;}
}
```

(Fin du sujet)