jIPST-CNAM
Programmation JAVA
NFA 031
Mercredi 8 Février 2017

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 à 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 le code en Java.

1ère PARTIE : COURS (sans document) 1h15mn

---

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, une classe est la déclaration d'un type de donnée structurée qui contient également des traitements informatiques qui permettent de modifier ce type de donnée		Q 1.
1	OUI	**X**
2	NON	

Dans un langage orienté objet, un objet représente un état d'information. Cet état est constitué de :		Q 2.
1	l'ensemble des attributs de l'objet	**X**
2	uniquement l'ensemble des attributs publiques de l'objet	
3	uniquement l'ensemble des attributs privés de l'objet	

Dans un langage orienté objet, un objet peut être composé d'autres objets qui peuvent être eux-mêmes composés de sous-objets.		Q 3.
1	OUI	**X**
2	NON	

En JAVA, on doit soi-même s'occuper de la destruction des objets qui ne sont plus utilisés.		Q 4.
1	OUI	
2	NON	**X**

En JAVA, pour compiler un programme (ensemble de plusieurs fichiers .java et donc constitué de plusieurs classes), il suffit de compiler la classe principale (contenant la méthode main)		Q 5.
1	OUI	**X**
2	NON	

Soit le fichier suivant le fichier CodeJava\exercices\cnam\util\Terminal.java.
Le fichier Terminal.java contient en 1ère ligne : **package exercices.cnam.util;**
Dans CodeJava\programme se trouve le fichier Prog.java suivant:
`import exercices.cnam.util.*;`
`public class Prog {`
`  public static void main(String… args) {`
`  Terminal.ecrireStringln("Bonjour");}`
`}`
On est dans le répertoire CodeJava/programme, et on veut compiler le programme.
Quelle(s) commande(s) est(sont) valide(s) :
		Q 6.

1	javac –classpath "." Prog.java	
2	javac Prog.java	
3	javac –classpath ".." Prog.java	**X**

Le compilateur Java (javac) permet de créer un ensemble de fichier .class qui sont ensuite interprétés par une JVM		Q 7.
1	OUI	**X**
2	NON	

En Java, l'exécution d'un programme peut se faire d'autant de façons différentes qu'il existe de méthode main dans les classes du programme		Q 8.
1	OUI	**X**
2	NON	

La commande **java**		Q 9.
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, les méthodes déclarées en dehors d'une classe sont appelées des méthodes statiques		Q 10.
1	OUI	
2	NON	**X**

En JAVA, une méthode statique d'une classe peut utiliser les attributs non statiques de la même classe:		Q 11.
1	OUI	
2	NON	**X**

Soit le code JAVA suivant : Q 12.

```
public class C {
   private int attribut;
   public C(){attribut=10;}
   public static void main(String[] args)
   {
      C  objet = new C();
      objet.attribut = Integer.parseInt(args[0]); // LIGNE 7
   }
}
```

Le code de la LIGNE 7 est correct

1	OUI	
2	NON	**X**

Les méthodes public de la classe A		Q 13.
1	peuvent utiliser directement les attributs privés de la classe A	**X**
2	peuvent utiliser directement les attributs privés d'une autre classe B	
3	peuvent utiliser directement les attributs publiques de la classe A	**X**

Soit la classe C1 dont tous les attributs sont privés 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 14.
1	OUI	
2	NON	**X**

Soit le code JAVA suivant : Q 15.

```
public class Exemple
{
   ????? int var_x;
   public static void main(String a_args[])
      {
         var_x = Integer.parseInt(a_args[0]);
         System.out.println("X=" + var_x);
      }
}
```

On exécute ce programme avec la commande : java Exemple 123
Pour que ce code soit correct, il faut remplacer ????? par :

1	private	
2	public	
3	static	X

Soit le code JAVA suivant : Q 16.

```
public class Bidon {
   public int valeur;
   public Bidon(int v){valeur=v;}
}
public class Exemple{
   public static void main(String[] args){
      Bidon obj = new Bidon(12);
      traitement(obj,100)
      System.out.println(obj.valeur);
   }
   static void traitement(Bidon bid,int v){
      bid.valeur=v;
}
```

L'exécutiuon de ce programme est :

1	affiche la valeur 100	X
2	affiche la valeur 12	

En JAVA, une méthode privée (private) d'une classe a accès aux attributs publics d'une autre classe Q 17.

1	OUI	X
2	NON	

En JAVA, pour créer un package, il suffit de : Q 18.

1	préfixer le nom d'un répertoire par le mot "package" et de mettre dans ce répertoire les fichiers java contenant les classes devant appartenir au package	
2	créer un répertoire de nom quelconque et de mettre dans ce répertoire les fichiers java contenant les classes devant appartenir au package	
	créer un répertoire de nom quelconque, de mettre dans ce répertoire les fichiers java contenant les classes devant appartenir au package et de déclarer le nom du package en entête de chacun des fichiers java appartenant à ce répertoire	X

En JAVA, dans une classe, il faut toujours créer un constructeur sauf si la classe contient que des méthodes statiques. Q 19.

1	OUI	
2	NON	X

Dans la programmation objet, en JAVA, le rôle du constructeur d'une classe est de : Q 20.

1	créer un objet en mémoire de la JVM et d'initialiser les attributs	X
2	créer une classe en mémoire de la JVM	

En JAVA, dans un constructeur il est possible d'initialiser les attributs privés de la classe mais pas les attributs publics de la classe		Q 21.
1	OUI	
2	NON	**X**

Soit la classe suivante :

```
public class Truc{
  private int x;
  public Truc(int x){ this.x = x; }
  public Truc(){ this.x = -1; }
}
```

L'instruction suivante :

```
Truc chose  = new Truc();
```

est valide et la valeur de x de chose a la valeur -1

		Q 22.
1	OUI	**X**
2	NON	

Soit la classe suivante :

```
public class Truc{
  public int x;
  public static Truc Truc(){
     Truc c=new Truc();return c; }
  public Truc(){ this.x = -1; }
}
```

L'instruction suivante :

```
Truc  chose  = Truc.Truc();
```

est valide

		Q 23.
1	OUI	**X**
2	NON	

Quand une classe définit un constructeur avec des paramètres alors le constructeur par défaut de Java (sans paramètre) est accessible		Q 24.
1	OUI	
2	NON	**X**

Soit la classe suivante :

```
public class A{
  private ArrayList<String> liste;
  public A(){
    ArrayList<String> liste = new ArrayList<String>();
  }
  public void add(String s){liste.add(s);}
}
```

Un programme réalise le code suivant:

```
A exemple = new A();
exemple.add("LAFONT");
```

Le code du programme s'exécute normalement :

		Q 25.
1	OUI	
2	NON	**X**

En JAVA, le type de retour d'une méthode est soit void, soit un type primitif, soit la référence d'un objet		Q 26.
1	OUI	**X**
2	NON	

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 peuvent être remplacés par :

		Q 27.
1	**A** ➔ `tab_int.length` **B** ➔ `tab_int[i]`	**X**
2	**A** ➔ `tab_int.size()` **B** ➔ `tab_int.get(i)`	

En JAVA, pour qu'un objet puisse être passé en paramètre d'une méthode, il faut qu'il soit différent de null		Q 28.
1	OUI	
2	NON	**X**

En JAVA, il est possible de créer des tableaux à 4 dimensions.		Q 29.
1	OUI	**X**
2	NON	

Soit le code suivant :

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

Ce code ::

		Q 30.
1	affiche : AA BB CC	
2	n'affiche rien	
3	ne se compile pas correctement	**X**

En Java, il est possible de modifier le contenu d'un tableau passé en paramètre d'une méthode		Q 31.
1	OUI	**X**
2	NON	

En Java, un tableau [] ne peut contenir que des données de type primitif (int, double, ...) alors qu'un ArrayList peut contenir aussi des objets		Q 32.
1	OUI	
2	NON	**X**

Soit le code JAVA suivant :		Q 33.

	`String[] tab = new String[2];`
	`tab[0]="UN"; tab[1]="DEUX";`
	`for(String s : tab){System.out.println(s);}`	

1	Ce code affiche : UN DEUX	**X**
2	Ce code affiche : UN DEUX null null	

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 34.
1	OUI	**X**
2	NON	

Ce code est correct :		Q 35.

	`ArrayList<String> tab = new ArrayList<String>();`
	`tab.array[0] = "LAFONT";`
	`tab.array[1] = "DUPONT";`
	`tab.size = 2;`	

1	OUI	
2	NON	**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

Faire la comparaison de l'utilisation des tableaux Java [] et des ArrayList() : déclaration, type des éléments, création, utilisation.,…

Les tableaux, comme les ArrayList , doivent être déclarés avant de les créer et les utiliser.
La création d'un tableau nécessite de préciser la taille maximale d'élément alors que la création d'un ArrayList ne nécessite pas de préciser une taille maximal.
Ainsi, dans un tableau on ne peut pas mettre plus d'élément que la taille maximale alors que dans un ArrayList il n'y a pas de limite au nombre d'élément (limité quand même à la taille de la JVM).
Un tableau Java peut contenir des éléments de type primitif ou de type référence
Par contre, un ArrayList, ne peut contenir que des types références.
On accède à un élément en demandant le i-ième élément, dans les tableaux comme dans les ArrayList, mais avec une syntaxe différentes (t[i], list.get(i)).

Le type tableau n'est pas une Classe, alors que ArrayList est une classe qui a donc de nombreuses méthodes pour l'utiliser.

Q 2

a) Expliquez pourquoi, il est fortement conseillé de déclarer les attributs privés.
b) Expliquez pourquoi, il ne faut pas abuser des attributs et des méthodes statiques.
c) Expliquez pourquoi, on a besoin de créer des méthodes privées.

a) parce que cela permet de les protéger de l'extérieur de la classe contre des mises à jour intempestives de ces attributs qui forment une certaine cohérence de l'état de l'objet. (sauf si il existe des setteurs sur ces attributs mais qui peuvent faire un certain contôle)
b) parce que sinon on ne fait plus de programmation orientée objet.
c) parce que cela permet de factoriser et/ou découper des méthodes publiques de la classe qui ne doivent donc pas être visible de l'extérieur.

Q 3

Décrivez précisément ce qu'est une méthode en Java (de quoi elle est composée, de son rôle, de ses natures, …)

En Java, une méthode est de la forme :
<nature> <type> <nom méthode>(<paramètres>) { <code> }
où
<nature> permet de préciser si la méthode est non statique ou statique, et/ou publique ou privée.
<type> est le type de retour de la méthode qui peut être void si pas de valeur de retour
<nom méthode> est le nom de la méthode
<paramètres> est la liste des paramètres d'entrées de la méthode qui peut être vide.
Le rôle d'une méthode est d'exécuter un traitement au sein de la classe. Si la méthode est non statique elle peut accéder et modifier les attributs objet de la classe.

Fin de la 1ère partie

2ème PARTIE : PROGRAMMATION (avec document) 1h15mn

Problème 1 [15 points]

```
  public Produit produitsMoinsChere(double minPrix,
                    double maxPrix, String recherche)
  {
    Produit produitMin=null;
    double prixCourant = maxPrix;
    for(Produit p : stock)
      {
        if (p.getNom().contains(recherche))
          if ((p.getPrix()>=minPrix) && (p.getPrix()<=maxPrix))
            if (p.getPrix()<=prixCourant)
              {
                prixCourant = p.getPrix();
                produitMin = p;
              }
      }
    return produitMin;
  }
```

Problème 2 [35 points]

```
public class Site
{
  private ArrayList<Reservation> reservations;
// Les produits du stock

  // Constructeur
  //
  public Site()
  {
    // Initialisation des attributs par defaut
    //
    reservations = new ArrayList<Reservation>();

    initialiserReservations("data/Reservations.txt");
  }

  private void initialiserReservations(String nomFichier)
  {
    // Lecture de tout le fichier
    String[] lignes = Terminal.lireFichierTexte(nomFichier);
    // Pour chaque ligne du fichier
    for(String ligne :lignes)
      {
        System.out.println(ligne);
        // On decode la ligne
        String[] champs = ligne.split("[;]",4);
        int numero       =  Integer.parseInt(champs[0]);
        String dateDebut  = champs[1];
        String dateFin    = champs[2];
        int    nbPersonne =  Integer.parseInt(champs[3]);
```

```
        Reservation r = new Reservation(numero,
                                        dateDebut,
                                        dateFin,
                                        nbPersonne
                                    );
        reservations.add(r);
      }
  }

  public String ajouterReservation(int numeroChambre,
                                   String dateDebut,
                                   String dateFin,
                                   int nbPersonne)
  {
    reservations.add(new Reservation(numeroChambre,
                                     dateDebut,
                                     dateFin,
                                     nbPersonne));
    return listerToutesReservations();
  }

  public String reservationsChambre(int numeroChambre)
  {
    String res ="";
    for(Reservation r : reservations)
      {
        if (r.getNumeroChambre()==numeroChambre)
          res=res+r.getDateDebut()+" "+r.getDateFin()+"\n";
      }
    if (res.equals("")) return "La chambre n'est pas reservee";
    else return res;
  }

  public String listerToutesReservations()
  {
    String res="";
    for(Reservation r : reservations)
      res=res+r+"\n";
    return res;
  }

  public boolean valide(int numeroChambre, String dateDebut, String
dateFin)
  {
    for(Reservation r : reservations)
      if (r.getNumeroChambre()==numeroChambre)
        {
          int nbJDeb = DateString.jours(dateDebut);
          int nbJFin = DateString.jours(dateFin);
          int nbJDebRev = DateString.jours(r.getDateDebut());
          int nbJFinRev = DateString.jours(r.getDateFin());
          if ( ( (nbJDeb>=nbJDebRev) && (nbJDeb<=nbJFinRev) ) ||
               ( (nbJFin>=nbJDebRev) && (nbJFin<=nbJFinRev) ) ||
               ( (nbJDeb<nbJDebRev) && (nbJFin>nbJFinRev) ) )
            return false;
        }
    return true;
  }


}
```

```
public class Reservation
```

```
{
  private int    numeroChambre;
  private String dateDebut;
  private String dateFin;
  private int    nbPersonne;

  // Constructeur
  //
  public Reservation(int numeroChambre,
                     String dateDebut,
                     String dateFin,
                     int nbPersonne)
  {
    this.numeroChambre = numeroChambre;
    this.dateDebut     = dateDebut;
    this.dateFin       = dateFin;
    this.nbPersonne    = nbPersonne;
  }

  // Conversion en chaine
  //
  public String toString()
  {
    return String.format("%d %s %s
%d",numeroChambre,dateDebut,dateFin,nbPersonne);
  }

  // Les getteurs et setteurs
  public int    getNumeroChambre(){return numeroChambre;}
  public String getDateDebut(){return dateDebut;}
  public String getDateFin(){return dateFin;}
  public int    getNbPersonne(){return nbPersonne;}

}
```

(Fin du sujet)