IPST-CNAM
Programmation JAVA
NFA 001
Mercredi 13 Février 2013

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:

Un langage de programmation orienté objet est :		Q 1
1	un langage dont les données créées et utilisées sont des paquets d'information décrites par un cadre de description appelé une "classe"	X
2	un langage permettant de créer des objets ou instance de classe	X
3	un langage spécifique permettant de créer des solutions internet	

Dans un langage orienté objet, comme Java, les traitements informatiques sont portés par les classes et les objets du langage		Q 2
1	OUI	X
2	NON	

Le compilateur Java (javac) permet de créer un exécutable qui ne s'exécute que sur le type de machine sur laquelle le programme a été compilé		Q 3
1	OUI	
2	NON	X

Pour exécuter un programme Java il est nécessaire, entre autre, d'installer et utiliser une JVM		Q 4
1	OUI	X
2	NON	

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 5
1	OUI	
2	NON	X

Soit une classe contenant les méthodes m1 et m2. m1 est une méthode static et m2 n'est pas une méthode static :		Q 6
1	la méthode m1 static peut utiliser les attributs non statics de la classe	
2	la méthode m2 non static peut utiliser les attributs statics de la classe	X
3	la méthode m1 static peut utiliser les attributs statics de la classe	X

Dans la programmation objet, en JAVA, le rôle du constructeur d'une classe est de :		Q 7
1	affecter les valeurs des attributs de la classe	X
2	allouer en mémoire du programme les attributs de l'objet	X
3	construire la classe (ou .class) qui permet à un autre programme de créer les objets de la classe	

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 signature d'une méthode **main** de la class **C1** permettant l'exécution d'un programme JAVA est de la forme **public static void main(String… args).** Cette méthode est static parce la commande **java C1 a1 a2** consiste à demander à la JVM Java d'exécuter l'instruction JAVA suivante : **C1.main(a1,a2)**		Q 9
1	OUI	X
2	NON	

Soit le code suivant : Q 10

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

Ce code affiche :

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

Un attribut protected d'une classe C1 appartenant au package P est un attribut qui est visible depuis :		Q 11
1	une classe C2 qui n'hérite pas de C1 et qui appartient à un autre package	
2	une classe C3 qui hérite de C1 et qui appartient à un autre package	X

Un attribut privé d'une classe C1 est quand même visible depuis les classes qui héritent de C1		Q 12
1	OUI	
2	NON	X

Il est déconseillé de créer des classes dont tous les attributs sont privés et pour laquelle il n'existe aucune méthode permettant de modifier ses attributs car une telle classe n'a pas d'utilité.		Q 13
1	OUI	
2	NON	X

Soit le code suivant : Q 14

```
int v=13;
boolean premier=true;
for(int k=2;k<v;k=k+1)
   if (v%k == 0) premier = false;
if (premier) System.out.println("PREMIER");
else System.out.println("NON PREMIER");
```

Ce code :

1	contient une erreur	
2	affiche "NON PREMIER"	
3	affiche "PREMIER"	X

La commande **javac** C1.java		Q 15
1	crée le fichier C1.class	X
2	compile la classe C1 et exécute la méthode main de la classe C1	

En JAVA, un tableau (tab[])		Q 16
1	peut contenir des éléments de type primitif	X
2	peut contenir des références d'objet JAVA	X
3	ne peut pas contenir de références d'objet JAVA	

En JAVA, les méthodes déclarées en dehors d'une classe sont appelées des méthodes statics		Q 17
1	OUI	
2	NON	X

Une méthode public d'une classe peut utiliser les attributs privés de la classe		Q 18
1	OUI	
2	NON	X

Une méthode static est une méthode dont le contenu (le code) reste inchangé durant tout le temps d'exécution du programme Java		Q 19
1	OUI	
2	NON	X

La classe StringBuffer de Java, comme la classe String, ne permet pas de modifier les caractères de la chaîne de caractère		Q 20
1	OUI	
2	NON	X

La classe StringTokenizer est une classe qui permet de créer des chaines de caractères et contient des méthodes permettant de modifier les caractères de la chaine de caractères.		Q 21
1	OUI	
2	NON	X

En Java, il est possible de modifier le contenu d'un tableau passé en paramètre d'une méthode		Q 22
1	OUI	X
2	NON	

En Java, il est possible de modifier le contenu d'un objet passé en paramètre d'une méthode		Q 23
1	OUI	X
2	NON	

En JAVA, un objet est un pointeur		Q 24
1	OUI	X
2	NON	

Si une classe B qui hérite d'une classe A et si B n'a pas de constructeur alors :		Q 25
1	lors de la création d'un objet de type B, une erreur d'exécution se produit si la classe A n'a pas défini de constructeur	
2	lors de la création d'un objet de type B, les attributs privés de A sont alloués en mémoire et initialisés par le(s) constructeur(s) hérité(s) de A	X
3	Il n'est pas possible de créer un objet de B car la classe B n'a pas de constructeur	

La boucle **for** dite "énumérative" permet :		Q 26
1	d'incrémenter une valeur scalaire de type enum et de réaliser un traitement à chaque valeur	
2	de parcourir tous les éléments d'une instance de ArrayList	X
3	de parcourir tous les éléments d'un tableau java (tab[])	X

Soit la classe suivante : Q 27

```
public class C1{
  private int x;
  public C1(int x){ this.x = x; }
}
```

L'instruction suivante : C1 c1 = new C1(); est valide et la valeur de x est la valeur par défaut de Java 0

1	OUI	
2	NON	X

Soit la classe suivante : Q 28

```
public class C1{
  private int nb;
  private ArrayList<String> liste;
  public C1(int nb){ this.nb = nb; }
  public void add(String s){liste.add(s);}
}
```

Le code suivant s'exécute sans erreur

```
C1 c1 = new C1(0);
c1.add("TOTO");
```

1	OUI	
2	NON	X

En JAVA, une exception est un objet à part entière qui est, notamment, une instance de la classe prédéfinie **Exception** Q 29

1	OUI	X
2	NON	

Le code suivant est correct : Q 30

```
public void action(int parametre)
{
   if (parametre==0)
      throw new Exception("Erreur");
   else
       faireLeTraitement();
}
```

1	OUI	
2	NON	X

Le code suivant est correct : Q 31

```
public void action(int parametre)
{
   if (parametre==0)
      throw new RuntimeException("Erreur");
   else
       faireLeTraitement();
}
```

1	OUI	X
2	NON	

Soit le fichier suivant C:\CodeJava\exercices\cnam\util\Terminal.java. Q 32
Le fichier Terminal.java contient en 1ère ligne : **package exercices.cnam.util;**
Le répertoire C:\bin est vide.
Dans C: on réalise la commande suivante :
javac -d C:\bin C:\CodeJava\exercices\cnam\util\Terminal.java
On obtient le résultat suivant :
C:\bin\ exercices\cnam\util\Terminal.class

1	OUI	X
2	NON	

Soit le fichier suivant C:\CodeJava\exercices\cnam\util\Terminal.java. Q 33

Le fichier Terminal.java contient en 1ère ligne : **package exercices.cnam.util;**

Dans C:\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 C:\CodeJava\programme, et on veut compiler le programme.

Quelle(s) commande(s) est(sont) valide(s) :

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

En JAVA, l'instruction suivante permet de déclencher une exception Q 34

```
throw new Exception("Impossible de faire l'action");
"
```

1	OUI	X
2	NON	

Le code suivant permet d'augmenter la taille du tableau t1 : Q 35

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

augmenterTailleTab(t1,100);
```

Avec :

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

1	OUI	
2	NON	X

(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

Expliquer ce qu'est la notion d'héritage dans le cadre des langages de programmation orientée objet.
Donnez un exemple type et commentez.

Soit la classe A qui hérite de B, la notion d'héritage est une propriété qui permet que quand on crée une instance de A alors l'objet créé contient également tous les attributs de la classe B.
De plus toutes les méthodes publics de la classe B peuvent être utilisées à travers une instance de A.

Exemple :

```
class B {
   private int x ;
   public B(int x)
      {
         this.x = x ;
      }
   public void m1() {…}
}

class A extends B {          // A hérite de B
   private int y ;
   public A(int x,int y)
   {
      super(x) ;             // appel du constructeur hérité de B
      this.y = y ;
   }
   public void m2() {
      m1() ;                 // m1 utilise la méthode héritée m1
   }
}
```

Q 2

En JAVA, il existe la classe Exception et la classe RuntimeException.
Expliquez précisément la différence qu'il existe dans l'utilisation de ces deux classes.
Donner une définition d'une exception.

La différence entre ces 2 classes et que quand une excecption de type Exception remonte il faut préciser dans l'en-tête de la méthode que cette exception peut remonter, en ajoutant ***throws Exception***
Ce qui n'est pas le cas dans le cas où c'est une exception de type RunTimeException qui remonte.

Une exception est une instance de la classe Exception (ou toute classe qui hérite de Exception) que nous déclenchons en cas d'erreur, en utilisant l'instruction **throw**. Ainsi l'exception remonte à l'appelant qui a le choix, soit de laisser passer l'exception (throws), ou soit de la capturer (try-catch).

Exemple de déclenchement : throw new Exception(« Ceci est une erreur ») ;

Q 3

En JAVA, quels sont les rôles du constructeur ?
Dans le langage Java expliquez la contrainte qui existe dans l'utilisation du constructeur par défaut et du constructeur sans paramètre.

Les rôles d'un constructeur sont, dans l'ordre:
- appeler le constructeur de la classe héritée (il existe toujours au moins celui de la classe Object dont toutes les classes héritent)

- allouer tous les attributs d'objet de la classe (les attributs de classe sont déjà alloués lors du chargement de la classe)
- initialiser les attributs de l'objet avec des valeurs soit passées en paramètre du constructeur soit calculées par le constructeur sinon avec les valeurs par défaut de JAVA

En Java si une classe définit que des constructeurs avec des paramètres alors le constructeur par défaut de Java de la classe n'est plus accessible (erreur de compilation).

(Tourner la page)

2ème PARTIE : PROGRAMMATION (avec document)

Exercice [15 points]

Le fichier cnam.exos.Exercice.java :

```
// Pour compiler depuis le répertoire "cnam" :
//   javac -d "../bin" -classpath ".." exos/Exercice.java
// Ppour exécuter depuis le répertoire "bin" :
//   java -classpath "." cnam.exos.Exercice
//
package cnam.exos;

import cnam.util.*;

public class Exercice
{
    public static void main(String... args)
    {
        // On saisit autant de cercle que l'on veut et on
        //  affiche le périmètre du cercle
        //
        String rep="";
        do{
            Cercle c1 = Cercle.saisir();
            Terminal.ecrireStringln("Perimetre de c1 : "+c1.perimetre());
            Terminal.ecrireString("Continuer (o,n)?");
            rep = Terminal.lireString();
        }while(rep.equals("o"));
    }
}
```

Le fichier cnam.util.Cercle.java :

```
package cnam.util;

public class Cercle
{
    // Constante de PI
    static final double PI = 3.141592654;

    private double x;      // Centre en x du cercle
    private double y;      // Centre en y du cercle
    private double rayon;  // Rayon du cercle

    // Constructeur du cercle
    //    initialisation des attributs
    //
    public Cercle(double x,double y,double rayon)
    {
        this.x=x;
        this.y=y;
        this.rayon=rayon;
    }

    // Retourne le périmètre du cercle
    //
    public double perimetre()
    {
        return ( PI * 2.0 * rayon );
    }

    // Méthode static pour saisir les attributs d'un cercle.
    // La mathode retourne le cercle créé.
    //
    public static Cercle saisir()
    {
```

```
        Terminal.ecrireString("x : ");
        double x = Terminal.lireDouble();
        Terminal.ecrireString("y : ");
        double y = Terminal.lireDouble();
        Terminal.ecrireString("rayon : ");
        double rayon = Terminal.lireDouble();

        Cercle c = new Cercle(x,y,rayon);
        return(c);
    }
}
```

Exercice [25 points]

```
import java.util.*;
// Classe de définition d'un compte bancaire
//
public class CompteBancaire
{
    private double soldeInitial;           // Le solde initial
    private ArrayList<Ecriture> ecritures; // La liste des écritures

    // Constructeur
    //   - initialisation du sole initial et de la liste des écritures
    //     en lisant le fichier et en décodant chacun des lignes
    //
    public CompteBancaire(String fichier) throws Exception
                                       // Exception si erreur de décodage
    {
        // Création de la liste des écritures
        ecritures = new ArrayList<Ecriture>();

        // Lecture du fichier
        String[] lignes = Terminal.lireFichierTexte(fichier);

        // Decodage de chacune des lignes du fichier
        soldeInitial=0.0;
        for(int i=0;i<lignes.length;i++)
            {
                try{
                    if (i==0)
                        {
                            // Decodage de la 1ere ligne contenant le solde
                            // initial du compte
                            StringTokenizer strtok=
                                 new StringTokenizer(lignes[0]);
                            strtok.nextToken();
                            if (strtok.nextToken().equals("CREDIT"))
                                soldeInitial =
                                 Double.parseDouble(strtok.nextToken());
                            else
                                soldeInitial =
                                 - Double.parseDouble(strtok.nextToken());
                        }
                    else
                        {
                            // Decodage de chacune des écritures qui sont
                            // ajoutées dans
                            //   la liste des écritures
                            StringTokenizer strtok = new
                                           StringTokenizer(lignes[i]);
                            Ecriture ecr = new Ecriture(
                                 strtok.nextToken(),
                                 Double.parseDouble(strtok.nextToken()),
                                 strtok.nextToken());
                            ecritures.add(ecr);
                        }
                }
                // Quelque soit l'erreur d'interprétation des lignes
                //   du fichier texte
```

```
                //   pas le bon nombre de mots,
                //   la valeur n'est pas un double, ...
                // alors on remonte une exception en indiquant la ligne
                //  où se trouve l'erreur
                catch(Exception ex){
                    int num = i+1;
                    throw new Exception("Erreur dans le fichier "+fichier+
                                        " a la ligne "+num);
                }
            }
    }

    // Méthode qui calcule le solde du compte
    public double calculerSolde()
    {
        double solde = 0.0;
        for(Ecriture e:ecritures)
            {
                if (e.credit) solde=solde + e.valeur;
                if (e.debit)  solde=solde - e.valeur;
            }
        return solde + soldeInitial;
    }
    // Méthode qui calcule la somme des dépenses ou
    //     crédits d'une nature donnée
    public double calculerSommeNature(String nature)
    {
        double somme = 0.0;
        for(Ecriture e:ecritures)
            {
                if (e.nature.equals(nature))
                    {
                        if (e.credit) somme=somme + e.valeur;
                        if (e.debit)  somme=somme - e.valeur;
                    }
            }
        return somme;
    }

    // Programme principal qui calcul le solde de chacun des comptes
    //    d'une famille
    //  et le solde total de tous les comptes cumulés
    public static void main(String... args)
    {
        double soldeTotal = 0.0; // Le solde total cupulé des compte

        // Pour chaque compte on va créé un CompteBancaire et
        //  obtenir le solde
        // Si une erreur est remontée alors on trace l'erreur et
        //   on passe au compte suivant
        //
        // Premier compte
        try{
            CompteBancaire c1 = new CompteBancaire( "COMPTE_JOINT.txt");
            double solde = c1.calculerSolde();
            System.out.println("Le solde de COMPTE_JOINT est : "+ solde);
            soldeTotal=soldeTotal+solde;
        }catch(Exception ex)
            { System.out.println(ex.getMessage()); }

        // Deuxieme compte
        try{
            CompteBancaire c1 = new CompteBancaire( "COMPTE_EPARGNE.txt");
            double solde = c1.calculerSolde();
            System.out.println("Le solde de COMPTE_EPARGNE est : "+ solde);
            soldeTotal=soldeTotal+solde;
        }catch(Exception ex)
            { System.out.println(ex.getMessage()); }

        // Troisieme compte
```

```
        try{
            CompteBancaire c1 = new CompteBancaire("LIVRET_A.txt");
            double solde = c1.calculerSolde();
            System.out.println("Le solde de LIVRET_A est : "+ solde);
            soldeTotal=soldeTotal+solde;
        }catch(Exception ex)
            { System.out.println(ex.getMessage()); }

        // Affichage du solde total
        System.out.println("Le solde total est de : "+ soldeTotal);
    }
}

// Classe interne qui définit une écriture d'un compte bancaire
class Ecriture
{
    boolean credit;  // true si l'écriture est un crédit sinon false
    boolean debit;   // true si l'écriture est un débit sinon false
    double  valeur;  // valeur de l'écriture
    String  nature;  // nature de l'écriture

    // Constructeur
    //
    public Ecriture(String debitCredit,double valeur,String nature)
    {
        this.credit = false;
        this.debit = false;
        if (debitCredit.equals("CREDIT")) this.credit = true;
        if (debitCredit.equals("DEBIT"))  this.debit  = true;
        this.valeur=valeur;
        this.nature=nature;
    }
}
```

(Fin du sujet)