IPST-CNAM
Programmation JAVA
NFA 001
Mercredi 10 Février 2016

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

1ère Session NFA 031

***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)

## 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: |
|---|---|

| En JAVA, il est possible d'écrire des méthodes en dehors de toute classe. On écrit les méthodes dans un fichier .java. Elles sont toutes des méthodes statiques. | | Q 1. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| En JAVA, une classe contient la déclaration des attributs et la déclaration des méthodes. | | Q 2. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Dans un langage orienté objet, un objet appartient toujours à une classe. | | Q 3. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Dans un langage orienté objet, un principe fort est que les données sont encapsulées dans une structure d'accès appelée une classe. | | Q 4. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| En programmation objet, la relation d'agrégation entre une classe A et B est utilisée pour décrire qu'un objet de type A se décompose en un sous-objet de type B. La vie du sous-objet B dépend de la vie de l'objet A auquel il appartient. | | Q 5. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Soit le fichier suivant C:\CodeJava\exercices\fr\cnam\util\Terminal.java.<br>Le fichier Terminal.java contient en 1ère ligne : **package exercices.fr.cnam.util;**<br>Le répertoire C:\bin est vide.<br>Dans C: on réalise la commande suivante :<br>**javac -d C:\bin C:\CodeJava\exercices\fr\cnam\util\Terminal.java**<br><br>On obtient dans le répertoire bin la création suivante : | | Q 6. |
|---|---|---|
| 1 | fr/cnam/util/Terminal.class | |
| 2 | exercices/fr/cnam/util/Terminal.class | |

| La commande javac prend en entrée un ou plusieurs fichiers .java. Elle les compile et crée un fichier exécutable binaire qui ensuite pourra être lancé sur le système d'exploitation. | | Q 7. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Pour exécuter un programme JAVA, il faut créer une classe de nom **Main** dans laquelle on écrit le code du programme principal. | | Q 8. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Pour exécuter un programme JAVA, il faut créer une méthode **main** dans n'importe quelle classe et dont la signature peut être : | | Q 9. |
|---|---|---|
| 1 | public void main() | |
| 2 | public static void main() | |
| 3 | public static void main(String[] args) | |

| Le fichier C1.java contient 2 classes : une classe publique C1 et une classe privée C2.<br>La commande **javac** C1.java | | Q 10. |
|---|---|---|
| 1 | crée qu'un fichier : C1.class | |
| 2 | compile la classe C1 et exécute la méthode main de la classe C1 | |
| 3 | crée deux fichiers : C1.class et C2.class | |

Soit le code suivant : Q 11.

```
public class Exemple {
  public static void main(String args[]) {
    String str = args[0];
    System.out.println(str);
}}
```

Commande d'exécution : java Exemple toto

| | | |
|---|---|---|
| 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 | |
| 3 | L'exécution de ce programme affiche à l'écran la chaine de caractère passée en argument: toto | |

| Une classe C appartient à un package **fr.cnam.exemple** si le fichier **C.java** se trouve dans le répertoire **fr/cnam/exemple** et si en en-tête du fichier **C.java** il y a la ligne : **package fr.cnam.exemple;** | | Q 12. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Soit une classe contenant les méthodes **mstat1** et **m2**. mstat1 est une méthode statique et m2 n'est pas une méthode statique : | | Q 13. |
|---|---|---|
| 1 | la méthode mstat1( statique) peut utiliser les attributs non statiques de la classe | |
| 2 | la méthode m2 (non statique) peut utiliser les attributs statiques de la classe | |
| 3 | la méthode mstat1 (statique) peut utiliser les attributs statiques de la classe | |

Soit le code JAVA suivant : Q 14.

```
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]);
  }
}
```

La partie de code en gras est correct

| | | |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

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

| Le paramètre **-classpath** ou la variable d'environnement **CLASSPATH** est utilisée pour désigner une liste de plusieurs chemin (ou path) d'accès à des répertoires. Chacun de ces répertoires contient les fichiers .class ou les packages utilisés dans la compilation ou dans l'exécution d'un programme JAVA. | | Q 16. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Un attribut privé (private) d'une classe A peut être directement utilisé par une méthode d'une autre classe B qui appartient au même package que A. | | Q 17. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Un attribut protégé (protected) d'une classe A peut être directement utilisé par une méthode d'une autre classe B qui appartient au même package que A. | | Q 18. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Soit la classe C1 qui contient un attribut privé. Une méthode de la classe C2 prend en paramètre une instance de la classe C1 et veut changer la valeur de cet attribut.<br>Cela est possible que s'il existe un setteur sur cet attribut dans la classe C1. | | Q 19. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Dans la programmation objet, en JAVA, le rôle du constructeur d'une classe est de : | | Q 20. |
|---|---|---|
| 1 | initialiser les attributs de la classe | |
| 2 | déclarer de nouveaux attributs | |
| 3 | construire la classe (ou .class) qui permet à un autre programme de créer les objets de la classe | |

Soit la classe suivante : Q 21.

```
public class C1{
  private int[] tab;
  public C1(int[] tt){ tab = tt; }
}
```

L'instruction suivante :

```
C1 c1 = new C1();
```

est valide, et la valeur de tab est la valeur par défaut de Java : null

| | | |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Dans le cadre du développement d'un programme Java, on définit une classe contenant aucune méthode et contenant que des attributs statics et publics. | | Q 22. |
|---|---|---|
| 1 | Cela n'est pas possible. | |
| 2 | Cela est possible mais inutile. | |
| 3 | Cela permet de déclarer des variables globales à tout le développement du programme Java. | |

Soit le code JAVA suivant : Q 23.

```
public class Exemple
{
  ????? int var_x;
}

public class C
{
   public void traitement(Exemple ex)
   {
      ex.var_x = 100;
   }
```

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

| | |
|---|---|
| 1 | private |
| 2 | public |

| Dans le constructeur d'une classe, il est possible d'appeler une méthode (statique ou non statique) de la même classe. | | Q 24. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

Soit le code JAVA suivant : Q 25.

```
public class C {
  private int attribut;
  public C(int param){attribut=param;}
  public int getAttribut(){return attribut;}
  public static void main(String[] args)
  {
    C  objet = new C();
    System.out.println(objet.getAttribut());
  }
}
```

Ce code est correct

| | |
|---|---|
| 1 | OUI |
| 2 | NON |

| En JAVA, une classe peut contenir plusieurs constructeurs | | Q 26. |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

Soit le code JAVA suivant : Q 27.

```
public class Livre
{
    // Le titre du livre
    private String titre;

    // Le constructeur
    public Livre(String titre){
      setTitre(titre);}

    // Setteur
    public void setTitre(String titre){
      this.titre=titre; }
}
```

L'utilisation (en gras) de setTitre dans le constructeur :

| | | |
|---|---|---|
| 1 | déclenche une erreur de compilation | |
| 2 | déclenche une erreur d'exécution | |
| 3 | est correct | |

En JAVA, le type prédéfini primitif **int** est un type référence (pointeur). Q 28.

| | | |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

Soit le code suivant : Q 29.

```
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 | Face aux feux du soleil | |
| 2 | Les cavernes d'acier | |

En Java, il est possible de modifier la référence d'un objet passé en paramètre d'une méthode. Q 30.

| | | |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

Soit le code JAVA suivant: Q 31.

```
public class C {
  public static void initTab(int nb, int[] tab){
    tab = new int[nb]; }
}
```

Et par ailleurs :

```
int[] tableau;
C.initTab(10,tableau);
System.out.println(tableau.size());
```

Ce code :

| | | |
|---|---|---|
| 1 | déclenche une erreur d'exécution (nullPointerException) | |
| 2 | affiche la valeur : 0 | |
| 3 | affiche la valeur: 10 | |

Le code suivant permet d'augmenter la taille du tableau t1 : Q 32.

```
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 | |
| 2 | NON | |

| L'avantage d'utiliser un ArrayList à la place d'un tableau [] est : | | Q 33. |
|---|---|---|
| 1 | Le ArrayList permet de stocker des objets alors que le tableau [] ne peut stocker que des types primitifs (int, double, ….) | |
| 2 | Le ArrayList permet de stocker un nombre indéterminé d'objet | |

Soit le code JAVA suivant : Q 34.

```
String[] tab = new String[4];
tab[0]="UN"; tab[1]="DEUX";
for(String s : tab){System.out.println(s);}
```

| | | |
|---|---|---|
| 1 | Ce code affiche :<br>UN<br>DEUX<br>null<br>null | |
| 2 | Ce code affiche :<br>UN<br>DEUX | |

Dans la classe String la méthode static ***void replace(String old, String new, String target )*** permet de remplacer dans la chaine ***target*** toutes les occurrences de la chaîne ***old*** par la chaîne ***new***. Q 35.

Exemple:

```
String cible = "un deux un quatre un";
String.replace("un","trois",cible);
System.out.println(cible);
// affiche: trois deux trois quatre trois"
```

| | | |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

**(Tournez 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**

Expliquez pourquoi en JAVA, il est possible de modifier le contenu d'un objet qui est passé en paramètre d'une méthode. Justifier votre réponse.

**Q 2**

En JAVA, il existe deux classes différentes, **String** et **StringBuffer**, qui permettent de gérer les chaînes de caractères.
Quel est la différence importante entre ces deux classes ?
Citez un exemple original (pas celui de la classe Terminal vu en cours) pour lequel vous utiliseriez la classe StringBuffer.

**Q 3**

Expliquez ce qu'est un "**package**" en JAVA.

**Fin de la 1ère partie**

# 2ème PARTIE : PROGRAMMATION (avec document)

## Problème 1 [15 points]

Soit les classes **Compte** et **Operation** vues dans le cadre du projet de cette année (voir l'annexe ci-après pour rappel).

Ecrire le code de la méthode de la classe **Compte** suivante :

```
public String rechercher(String chaine)
```

Cette méthode retourne sous la forme d'une chaîne de caractère la concaténation de toutes les opérations du Compte dont le champ **libelle** contient au moins un des mots de **chaine** (les mots sont les groupes de caractères espacés par des caractères blancs).

## Problème 2 [35 points]

On se propose de créer une classe **Banque** qui permet de gérer la liste de comptes bancaires des clients de la banque. Ces comptes sont définis par la classe **CompteClient**,

Le fichier data/Banque.txt contient la liste des comptes bancaires (exemple) :
31/10/2013;00123456;LAFONT Paul;31/10/1960;8912G
12/12/2015;00124001;DUPONT Andre;23/03/1972;3456D

Chaque ligne de ce fichier correspond à un **CompteClient** et contient :

- la date de création du compte
- le numéro client
- les nom et prénom du titulaire du compte
- la date de naissance du titulaire du compte
- le nom du compte qui correspond au fichier data/Compte<*nom du compte*>.txt contenant les opérations du compte.

1/ Ecrire les attributs et le constructeur de la classe **Banque** et **CompteClient**. Le constructeur de la classe Banque lit un fichier de banque (exemple "data/Banque.txt") et initialise la classe Banque avec les informations lues dans le fichier.

2/ Ecrire la méthode de la classe Banque suivante :

```
public Compte lireCompte(String nomDuCompte)
```

Cette méthode recherche si le nom du compte existe dans la Banque. S'il existe la méthode retourne un **Compte** (la classe Compte vu dans le projet de cette année. (voir l'annexe ci-après)) contenant toutes les opérations du compte, sinon retourne null.

3/ Ecrire la méthode de la classe Banque suivante:

```
public ArrayList<String> comptesClient(String numeroClient)
```

Cette méthode retourne la liste des comptes qu'un client possède dans la banque.

## Annexe

Pour rappel (le code de ma correction) :

```
public class Compte
{
    private ArrayList<Operation> elements;  // La liste des operations

    public Compte(String nomFichier)
    {
       initialisation du compte avec la lecture du fichier
       […]
    }
}

public class Operation
{
    private String  date;           // date de l'operation
    private double  montant;        // le montant
    private String  type;           // "CB","CHEQ", "VIRT",
                                       "PRVT", "DEPT" ou ""
    private boolean budgete;        // vrai si operation budgete
                                          sinon faux
    private String  num_cb;         // numero de la CB de
                                       l'operation
    private int     num_cheque;     // numero du cheque de
                                       l'operation
    private String  libelle;        // libelle de l'operation
    private int     numero;         // numero de l'operation

    public Operation(int     numero,
                     String  date,
                     double  montant,
                     String  type,
                     boolean budgete,
                     String  num_cb,
                     int     num_cheque,
                     String  libelle)
    {
        //Initialisation des caracteristiques de l'operation
        this.numero      = numero;
        this.date        = date;
        this.montant     = montant;
        this.type        = type;
        this.budgete     = budgete;
        this.num_cb      = num_cb;
        this.num_cheque  = num_cheque;
        this.libelle     = libelle;
    }

    // Tous les getteurs
    […]
}
```

NB: Vous n'avez pas besoin d'écrire les getteurs, setteurs sur les attributs des classes et ni les méthodes toString() des classes. Vous considérez que ces méthodes vous sont données.

***(Fin du sujet)***