IPST-CNAM
Programmation JAVA
NFA 001
Mercredi 19 Février 2014

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 la programmation orientée objet, un objet est :		Q 1
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	

La difficulté dans la programmation objet est la maitrise de la destruction des objets		Q 2
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

Soit le fichier suivant C:\CodeJava\exercices\fr\cnam\util\Terminal.java. Le fichier Terminal.java contient en 1ère ligne : **package exercices.fr.cnam.util;** Le répertoire C:\bin est vide. Dans C: on réalise la commande suivante : **javac -d C:\bin C:\CodeJava\exercices\fr\cnam\util\Terminal.java** On obtient la création du fichier suivant : C:\bin\exercices\fr\cnam\util\Terminal.class		Q 3
1	OUI	X
2	NON	

Les fichiers .class générés par la compilation javac sont des fichiers intermédiaires qui contiennent un programme exprimé dans un autre langage intermédiaire (P-CODE) et qui est ensuite :		Q 4
1	traduit en fichier binaire directement exécuté par le microprocesseur	
2	interprété par un interpréteur de code P-CODE	X

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 5
1	OUI	X
2	NON	

En Java, l'exécution d'un programme, via la commande "java", est possible par création de la méthode **main**. Cette méthode peut avoir la signature suivante :		Q 6
1	public void main()	
2	public static void main()	
3	public static void main(String… args)	X

Soit le code suivant : Q 7

```
public class Exemple {
  public static void main(String args[]) {
    String str = args[1];
    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	X
3	L'exécution de ce programme affiche à l'écran la chaine de caractère passée en argument"	

Soit le code JAVA suivant : Q 8

```
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

Une méthode static est une méthode dont le contenu (le code) reste inchangé durant tout le temps d'exécution du programme Java		Q 9
1	OUI	
2	NON	X

Le constructeur dans une classe d'objet permet :		Q 10
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

Soit la classe suivante : Q 11

```
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

Quand une classe définit un constructeur avec des paramètres alors le constructeur par défaut (sans paramètre) n'est plus accessible		Q 12
1	OUI	X
2	NON	

Soit le code suivant : Q 13

```
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

En Java, il est possible de modifier le contenu d'un objet passé en paramètre d'une méthode		Q 14
1	OUI	X
2	NON	

La boucle **for** dite "énumérative" permet :		Q 15
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 le code suivant : Q 16

```
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

En JAVA, un tableau (tab[])		Q 17
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	

De ces 3 classes quelles sont les classes qui permettent de modifier une chaîne de caractère		Q 18
1	String	
2	StringBuffer	X
3	StringTokenizer	

La classe StringTokenizer :		Q 19
1	a un constructeur dont la signature est : StringTokenizer(String str, String tokens)	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 (indice) des tokens contenus dans la chaîne de caractère str	

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	

Le code suivant permet d'augmenter la taille du tableau t1 : Q 21

```
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

Une différence entre un tableau java [] et un ArrayList est : Q 22

1	un tableau ne peut pas contenir d'objet alors qu'un ArrayList peut contenir des objets	
2	un tableau a une capacité fixe alors qu'un ArrayList a une capacité dynamique	X

En JAVA, la déclaration d'un tableau comme suit : Q 23

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

contient :

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

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 24

1	OUI	
2	NON	X

La commande **javac** C1.java Q 25

1	crée le fichier C1.class	X
2	compile la classe C1 et exécute la méthode main de la classe C1	

Soit le code JAVA suivant : Q 26

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

Ce code est correct

1	OUI	X
2	NON	

Soit la classe suivante :		Q 27

```
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, le type de retour d'une méthode est soit void, soit un type primitif, soit la référence d'un objet		Q 28
1	OUI	X
2	NON	

En JAVA, le passage des paramètres se fait :		Q 29
1	par référence	
2	par valeur	X

Soit le code suivant : Q 30

```
public class Test2 {
    public static void main(String a_args[])
    {
        int v = 10;
        proc(v);
        Terminal.ecrireIntln(v);
    }

    static void proc(int p)
    {
        p = 100;
    }
}
```

Le résultat est :

1	10	X
2	100	
3	erreur d'exécution	

En JAVA, il est possible de définir deux méthodes de même nom dans la même classe		Q 31
1	OUI	X
2	NON	

Soit le code JAVA suivant — Q 32

```
public class Test {
    public static void main(String args[])
    {
        Exemple1 ex = new Exemple1();
        ex.tab[0] = 22;
    }
}

class Exemple1
{
    private int[] tab;

    public void Exemple1()
    {
        tab = new int[10];
    }
}
```

Ce code s'exécute correctement :

1	OUI	
2	NON	X

Le constructeur de la classe ArrayList suivant : public ArrayList<String>(int taille) crée une collection de chaine de caractère dont le tableau en interne est dimensionné à *taille* éléments et il n'est donc pas possible de mettre plus de *taille* éléments dans la collection — Q 33

1	OUI	
2	NON	X

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	

La classe StringBuffer de Java, est une variante de la classe String. Elle permet de créer des chaines de caractère dans lesquelles il est possible d'ajouter, supprimer et insérer des caractères. — Q 35

1	OUI	X
2	NON	

(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

Expliquez le rôle et le contenu d'une classe dans la programmation objet.

La classe a le rôle de définir le cadre de définition d'un objet.
Elle définit les attributs de l'objet (sa structure) et définit les méthodes qui sont les traitements qu'il est possible d'appeler et qui s'appliquent sur l'objet.
Elle permet de créer, par instanciation, des objets dont le type d'appartenance est la classe. Pour cela elle définit des méthodes particulières appelées "constructeur".
Elle peut aussi contenir des méthodes et attributs (dits "static") qui sont appelés à travers la classe.

Q 2

Expliquer à quoi sert la classe ArrayList<E>

La classe ArrayList<E> est une collection d'éléments de type E. E peut être un objet appartenant à n'importe quelle classe.
Elle permet d'ajouter un nouvel élément, de supprimer un élément donné et d'insérer un élément donné.
Les éléments sont rangés par ordre d'ajout. Ils sont indicés c'est-à-dire que chaque élément est à un indice qui va de 0 à n-1 où n est le nombre d'élément ajoutés ou insérés dans la collection.
Cette collection n'a pas de limite de taille.

Q 3

Qu'est ce qu'un *attribut d'objet* (non static) dans une classe.
Citez où dans la classe sont utilisés ces attributs.

Un attribut d'objet dans une classe est une valeur typée (ex: Individu ind) qui représente une caractéristique d'un objet du type de la classe.
Ainsi un attribut peut être un type primitif (ex int char ...) ou un type référence (tableau, classe)
On le trouve :
- dans sa déclaration (privé ou public) qui précise le nom de l'attribut et son type
- dans un constructeur où il est initialisé par une valeur interne ou par une valeur passée en paramètre
- dans une méthode où il est utilisé ou modifié

2ème PARTIE : PROGRAMMATION (avec document)

Exercice 1 [25 points]

```
class ComptesClient
{
    // Compteur utilisé pour identifier chaque client
    //   de manièr eunique
    private compteur;

    // Les cartes clients
    private ArrayList<CarteClient> clients;

    // Constructeur
    public ComptesClient()
    {
        compteur = 1;
        clients = new ArrayList<CarteClient>();
    }

    // Traitement de creation d'un nouveau client
    /* Algorithme :
       Debut
         Determiner l'identificateur du client
         Creer le nouveau client
         Ajouter le nouveau client
       Fin
     */
    public void creerNouveauClient()
    {
        String identificateur = compteur+"";
        compteur = compteur + 1;
        CarteClient cc = new CarteClient(identificateur);
        clients.add(cc);
    }

    // Traitement de passage en caisse
    /* Algorithme :
       Debut
         Rechercher l'indice du client dans la collection
         Si le client est trouvé alors
           soit cc le client trouvé;
           calculer le solde et les points de fidélité;
           mettre à jour cc avec le solde et les points;
         Sinon
           Erreur
         Finsi
       Fin
     */
    public void passerEnCaisse(String identificateur,
                               String fichierCaisse)
    {
        int indice = -1;
        for(int i=0;i<clients.size();i++)
            {
                CarteClient cc = clients.get(i);
                if (cc.getIdentificateur().equals(identificateur))
                    {
                        indice = i;
```

```
                        break;
                    }
            }
        if (indice != -1)
            {
                CarteClient cc = clients.get(indice);
                double[] r = traiterCaisseClient(identificateur,
                                                 fichierCaisse);
                cc.setSolde(r[0]);
                cc.setPoints(t[1]);
            }
        else
            Terminal.ecrireStringln("Erreur: non trouve
"+identificateur);
    }

    // Traitement d'un fichier caisse pour un client donné
    //
    private double[] traiterCaisseClient(String identificateur,
                                         String fichierCaisse)
    {
        String[] lignes = lireFichierTexte(nomFichier);

        double solde  = 0.0;
        double points = 0.0;
        for(String l : lignes)
            {
                StringTokenizer strtok = new StringTokenizer(" ");
                String numero_s  = strtok.nextToken();
                String date_s    = strtok.nextToken();
                String nombre_s  = strtok.nextToken();
                String montant_s = strtok.nextToken();
                String points_s  = strtok.nextToken();
                // Inutile de lire le reste de la ligne (non
utilise)

                if (numero_s.equals(identificateur))
                    {
                        double nb = Double.parseDouble(nombre_s);
                        double prix = Double.parseDouble(montant_s);
                        solde = solde + nb * prix;
                        points = points +
Double.parseDouble(points_s);
                    }
            }
        double[] resultat = new double[2];
        resultat[0] = solde;
        resultat[1] = points;

        return resultat;
    }
}

class CarteClient
{
    // Identidicateur du client
    private String identificateur;

    // Le solde courant du client
    private double solde;

    // Les points de fidélité accumulés
    private double points;
```

```
    // Constructeur
    public CarteClient(String identificateur)
    {
        this.identificateur = identificateur;
        this.solde = 0.0;
        this.points = 0.0;
    }
}
```

Exercice 2 [15 points]

```
public afficherRdv(String dateCourante,
                   ArrayList<String> rdvs)
{
    // Decodagde de la date en entree
    String[] datec = dateCourante.split(" ");
    String jour = datec[0];
    String date = datec[1];

    // Les jours de la semaine
    String[] jours = {"lundi","mardi","mercredi","jeudi",
                      "vendredi","samedi","dimanche"};

    // Parcourir tous les rendez-vous
    for(String rdv : rdvs)
        {
            String[] elements=rdv.split(";");
            String drdv  = elements[0];
            String heure = elements[1];
            String texte = elements[2];

            boolean toujours=true;

            // Si la date du rdv est le jour d'entrée
            for(String j:jours)
                {
                    if ( (j.equals(jour)) &&
                         (drdv.equals(jour) ) )
                        {
                            System.out.println(heure+" "+texte);
                            toujours = false;
                        }
                }

            // Si la date du rdv est la date d'entrée
            if (drdv.equals(date))
                {
                    System.out.println(heure+" "+texte);
                    toujours = false;
                }

            // Si la date n'en est pas une
            if (toujours)
                System.out.println(date+" "+" "+texte);

        }
}
```