wwjIPST-CNAM
Programmation JAVA
NFA 031
Mercredi 14 Février 2018

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) 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:

Un Langage Orienté Objet est un langage de programmation informatique qui permet de rassembler dans une même structure (la classe) le code et les données du programme.		Q 1.
1	OUI	
2	NON	

Dans un Langage Orienté Objet les attributs sont :		Q 2.
1	les données du programme	
2	le code du programme	

Dans un Langage Orienté Objet, un objet est :		Q 3.
1	une instance d'une classe	
2	une classe	
3	un package	

Soit le fichier suivant C:\CodeJava\exercices\fr\cnam\util\Terminal.java. Le fichier Terminal.java contient en 1ère ligne : **package 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 4.
1	OUI	
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 5.
1	OUI	
2	NON	

En Java, soit une classe A qui utilise une autre classe B. Si on compile la classe A alors la classe B est automatiquement compilée		Q 6.
1	OUI	
2	NON	

En Java, l'exécution d'un programme, via la commande "java", exécute la méthode **main**. Cette méthode peut avoir la signature suivante :		Q 7.
1	public void main()	
2	public static void main()	
3	public static void main(String… args)	

Le paramètre **-classpath** ou la variable d'environnement **CLASSPATH** est utilisée pour désigner une liste de plusieurs 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 8.
1	OUI	
2	NON	

Soit le code suivant : Q 9.

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

Commande d'exécution : java Exemple 1 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 : "TOTO"	

En JAVA, un répertoire qui contient des fichiers .java est nécessairement un package		Q 10.
1	OUI	
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 11.
1	OUI	
2	NON	

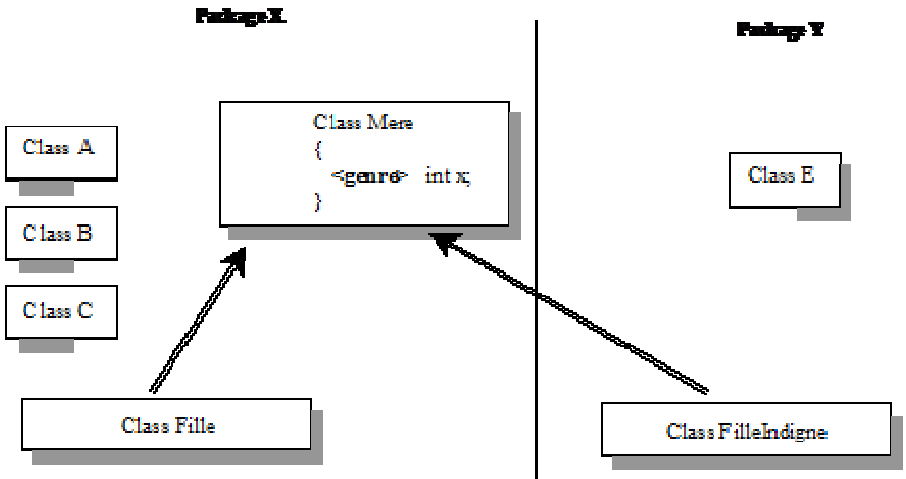
Soit une classe A contenant les méthodes mstat1 et m2. mstat1 est une méthode statique et m2 n'est pas une méthode statique :		Q 12.
1	la méthode mstat1 peut utiliser les attributs statiques de la classe A	
2	la méthode m2 peut utiliser les attributs statiques de la classe A	
3	la méthode mstat1 peut utiliser les attributs non statiques de la classe A	

Soit le code JAVA suivant : Q 13.

```
public class C {
     private int attribut;
     public C(){attribut=10;}
     public void setAttribut(int a){attribut=a;}
     public int getAttribut(){return attribut;}
     public static void main(String[] args)
     {
           C  objet = new C();
           objet.setAttribut(Integer.parseInt(args[0]));
           System.out.println(""+objet.getAttribut());
     }
}
```

L'exécution de ce code est : java C 100
Il affiche :

1	10	
2	100	



Q 14.

Si <genre> est private alors :

1	Seules les classes Fille, FilleIndigne et Mere peuvent accéder à l'attribut x	
2	Seules les classes A, B, C, Fille et Mere peuvent accéder à l'attribut x	
3	Seule la classe Mere peut accéder à l'attribut x	

En JAVA, le constructeur d'une classe est une méthode particulière de la classe dont le rôle est de créer une instance de la classe.		Q 15.
1	OUI	
2	NON	

En JAVA, une instance de la classe est un paquet de code élémentaire, appelé aussi bibliothèque, qui permet de centraliser les données de même nature dans un même container.		Q 16.
1	OUI	
2	NON	

Soit la classe suivante : Q 17.

```
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	

Dans un constructeur de la classe A, il est possible d'appeler des méthodes de la classe A mais il faut que ces méthodes soient publiques.		Q 18.
1	OUI	
2	NON	

Soit le code JAVA suivant — Q 19.

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

class Exemple1
{
    public int[] tab;

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

Ce code s'exécute correctement :

1	OUI
2	NON

Soit la classe suivante : — Q 20.

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

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

  public Individu(String nom, String prenom)
  {
    this.nom=nom;  this.prenom=prenom;  this.age = 0;
  }
}
```

On peut écrire les codes suivants :

1	Individu ind = new Individu("LAFONT","Pierre",45);
2	Individu ind = new Individu();
3	Individu ind = new Individu("LAFONT","Pierre");

Soit la méthode JAVA suivante : — Q 21.

```
public static void ajouter(String s,
Arraylist<String> l){l.add(s);}
```

Soit le programme JAVA qui utilise cette méthode:

```
ArrayList<String> liste=new ArrayList<String>();
liste.add("ZERO");
ajouter("UN",liste);
ajouter("DEUX",liste);
ajouter("TROIS",liste);
for(String s:liste)System.out.print(s+" ");
```

On obtient l'affichage suivant :

1	ZERO UN DEUX TROIS
2	ZERO

En JAVA, quand on passe un objet en paramètre d'une méthode, le langage passe en valeur la référence de l'objet.		Q 22.
1	OUI	
2	NON	

Soit le code suivant : Q 23.

```
int v=5;
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"

En JAVA, le type de retour d'une méthode est toujours la référence d'un objet ou void	Q 24.
1 OUI	
2 NON	

Soit le code suivant : Q 25.

```
public class Exemple
{
  private int n;
  private ArrayList<String> arr;
  public Exemple(){n=0;arr=new ArrayList<String>();}
  public add(String e){arr.add(e);}
  public String next(){return arr.get(n++);}
}
```

Dans un programme Java:

```
  Exemple ex = new Exemple();
  ex.add("UN"); ex.add("DEUX"); ex.add("TROIS");
  Terminal.ecrireString(ex.next()+" ");
  Terminal.ecrireString(ex.next()+" ");
  Terminal.ecrireString(ex.next());
```

Ce code :

1	ne fonctionne pas correctement
2	affiche : "UN DEUX TROIS"

Soit le code suivant : Q 26.

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

De ces 3 classes quelles sont les classes qui permettent de modifier une chaîne de caractère		Q 27.
1	String	
2	StringBuffer	
3	StringTokenizer	

Soit le code : String s = "ABCDEF"; s.setChar(3,'X'); La chaine s contient alors : "ABCXEF"		Q 28.
1	OUI	
2	NON	

En JAVA, pour augmenter la taille physique d'un tableau il faut créer un autre tableau de taille plus grande, recopier tous les éléments de l'ancien tableau dans le nouveau et changer la référence de l'ancien pour qu'elle devienne celle du nouveau.		Q 29.
1	OUI	
2	NON	

Soit le code JAVA suivant : Q 30.

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

Ce code :affiche :

1	un#deux#trois#quatre#cinq#six#sept#huit#
2	un##deux_trois##quatre_cinq#six_sept_huit#
3	un::deux#trois,:quatre#cinq:six#sept#huit#

Soit le code JAVA suivant :	Q 31.

`String s1="EXEMPLE"; String s2=new String(s1);`
`if (s1==s2) System.out.println("Les deux chaines sont egales");`
`else System.out.println("Les deux chaines ne sont pas egales");`	

L'exécution de ce code est :	
1 `Les deux chaines sont egales`	
2 `Les deux chaines ne sont pas egales`	

Si on compare un tableau java [] et un ArrayList,	Q 32.
1 un tableau peut contenir des valeurs de types primitives (int, double, …) alors que le ArrayList ne peut contenir que des objets (instances de classe).	
2 un tableau a une taille fixe alors que le ArrayList n'a pas de taille fixe.	
3 les éléments d'un tableau, comme ceux d'un ArrayList, sont rangés et indexés.	

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 ce qui n'est pas possible avec la classe String.	Q 33.
1 OUI	
2 NON	

Soit le code suivant :	Q 34.

`int[][] matrice = new int[5][5]`
`matrice[0] = new int[3];`
`matrice[1] = new int[2];`	

A l'issu de cette exécution :	
1 matrice contient 25 éléments	
2 matrice contient 15 éléments	
3 matrice contient 20 éléments	

Soit le code JAVA suivant :	Q 35.

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

Ce code affiche : UN DEUX null null	
1 OUI	
2 NON	

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 d'un **attribut** d'une classe dans la programmation orientée objet.

Q 2

Expliquez la différence qui existe entre une méthode statique et une méthode non statique.
Pourquoi utiliser des méthodes statiques ?

Q 3

Expliquez à quoi servent les getteurs et les setteurs dans la programmation objet.

Fin de la 1ère partie

2ème PARTIE : PROGRAMMATION (avec document) 1h15mn

Problème 1 [15 points]

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

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

```
public String premier()
```

Cette méthode retourne l'identification du joueur qui a fait le meilleur score (différences des scores du joueur et de l'adversaire).

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

```
public String deuxieme(String identPremier)
```

Cette méthode retourne l'identification du joueur qui est deuxième dans le classement, c'est-à-dire celui qui a le meilleur score sans prendre en compte le premier dont l'identificateur est *identPremier*. S'il n'existe pas de deuxième alors la méthode retourne "".

Problème 2 [35 points]

On se propose de créer une plateforme de jeu (classe **PFJeu**) qui gère des jeux (classe **Jeu**) auxquels les joueurs peuvent jouer.
Le fichier texte, **data/Jeux.txt**, contient des jeux permettant d'initialiser la plateforme.
Exemple :
Othello;2;8;8
Puissance4;3;7;6
Dame;2;10;10
Morpion;4;120;200

Chaque ligne de ce fichier correspond à un **Jeu** et contient :
- le nom du jeu
- le nombre de joueur
- le nombre de colonne de la grille
- le nombre de ligne de la grille

1) Ecrire le code de la classe **PFJeu** et écrire la classe **Jeu** (ne pas écrire les getteur, setteur) permettant de :

   - lire le fichier data/Jeux.txt et de charger en mémoire les jeux;

   - écrire la méthode **jeux** de la classe PFJeu qui retourne la liste des noms des jeux auxquels il est possible de jouer :

```
public String[] jeux()
```

2) Soit le code de la classe **Puissance4** suivant :

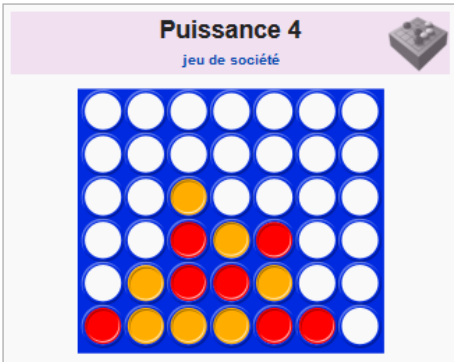
```
public class Puissance4 {
  final public static int NB_LIGNE=6;    // coord y
  final public static int NB_COLONNE=7;  // ccord x
  private Jeu jeu;
  private int[][] grille; //0=vide;1=rouge;2=jaune
  public Puissance4(Jeu jeu){
    this.jeu=jeu;
    grille =new int[jeu.getNbX()][jeu.getNbY()];
  }
}
```

L'attribut grille contient les valeurs des jetons.

- écrire la méthode **coupValide** de la classe Puissance4 qui retourne vrai si le coup joué en x,y est valide, sinon faux. *joueur* est la couleur du joueur : 1 ou 2.

```
public boolean coupValide(int joueur, int x, int y)
```

Règles du jeu Puissance4 :
Le but du jeu est d'aligner une suite de 4 pions de même couleur sur une grille comptant 6 rangées et 7 colonnes. Chaque joueur dispose de 21 pions d'une couleur (par convention, en général jaune ou rouge). Tour à tour les deux joueurs placent un pion dans la colonne de leur choix, le pion doit être à la position la plus basse possible dans la dite colonne à la suite de quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le premier un alignement (horizontal, vertical ou diagonal) consécutif d'au moins quatre pions de sa couleur.



Un début de partie de Puissance 4.

Annexe :

```
public class PFJeu
{
  private Othello  othello;
  private IHMPFJeu ihm;
  private ArrayList<Joueur> joueurs;
  private ArrayList<Partie> parties;
[…]
}

public class Partie
{
  private int     numero;
  private String  identJoueur;
  private String  date;
  private String  nomJeu;
  private int     partieTerminee;
  private int     score;
  private int     scoreAdversaire;
[…]
  public String getIdentJoueur(){return identJoueur;}
  public int getScore(){return score;}
  public int getScoreAdversaire(){return scoreAdversaire;}
}
```

(Fin du sujet)