

IPST-CNAM
Programmation JAVA
NFA 032
Mercredi 4 Juillet 2018

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

1^{ère} Session NFA 032

L'examen se déroule en deux parties. Une première partie de 1h15mn, sans document, consacrée à des questions de cours, et une deuxième partie de 1h 15mn, 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.

1^{ère} PARTIE : COURS (sans document)
Durée: 1h15

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
 - -1/2 pour chaque réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est fausse
 - + 1/2 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 programmation objet, le lien d'héritage est un lien de relation entre deux classes B et A (B hérite de A) qui permet, entre autre, que les méthodes de B, privées ou publiques, peuvent utiliser <u>directement</u> les méthodes publiques de A		Q 1.
1	OUI	
2	NON	

En programmation objet, le lien d'héritage est un lien de relation entre deux classes B et A (B hérite de A) qui permet, entre autre, que les méthodes de B, privées ou publiques, peuvent utiliser <u>directement</u> les attributs privés et publiques de A		Q 2.
1	OUI	
2	NON	

Un des usages important de l'héritage est de factoriser les attributs communs à plusieurs classes dans une autre classe dont elles héritent.		Q 3.
1	OUI	
2	NON	

Soit B qui hérite de A, alors tout constructeur de B hérite toujours d'un constructeur de A (B appelle, via super, un constructeur de A).		Q 4.
1	OUI	
2	NON	

Si le constructeur d'une classe B utilise l'instruction "super();" alors cela signifie que B hérite d'une classe A dans laquelle il n'existe pas de constructeur ou il existe un constructeur sans paramètres.		Q 5.
1	OUI	
2	NON	

En JAVA, le mot clef "super" n'est utilisé dans une classe que pour appeler les constructeurs de la classe dont elle hérite.		Q 6.
1	OUI	
2	NON	

En JAVA, une classe B qui hérite de A, ne peut surcharger que les méthodes privées de la classe A.		Q 7.
1	OUI	
2	NON	

En JAVA, toutes les classes, prédéfinies ou non prédéfinies, héritent de la classe prédéfinie Object.		Q 8.
1	OUI	
2	NON	

En programmation objet JAVA, le polymorphisme d'une collection d'élément est possible quand toutes les classes d'appartenance des éléments héritent d'une même classe abstraite.		Q 9.
1	OUI	
2	NON	

En programmation objet JAVA, une méthode générique est une méthode qui appartient à une classe qui implémente au moins une interface.		Q 10.
1	OUI	
2	NON	

Soit le schéma suivant :

```

classDiagram
    class Container {
        +ArrayList<ClasseRacine> elements
        +void traitement() {
            for (ClasseRacine e : elements)
            e.methode1();
        }
    }
    class ClasseRacine {
        +void methode1()
    }
    class Classe1 {
        +void methode1()
    }
    class Classe2 {
    }
    class Classe3 {
        +void methode1()
    }
    Container "1" *-- "*" ClasseRacine : elements
    ClasseRacine <|-- Classe1
    ClasseRacine <|-- Classe2
    ClasseRacine <|-- Classe3
    
```

		Q 11.
1	La classe "ClasseRacine" peut être une classe quelconque	
2	La méthode "méthode1" de la classe "ClasseRacine" doit être une méthode abstraite	
3	la méthode "traitement" est un traitement générique	

Dans une classe abstraite toutes les méthodes doivent être des méthodes abstraites.

		Q 12.
1	OUI	
2	NON	

En programmation objet, soit une méthode dont un paramètre est une interface alors on doit passer en paramètre une instance d'une classe qui doit implémenter cette interface.

		Q 13.
1	OUI	
2	NON	

Soit une classe B qui hérite d'une classe A.
A possède un seul constructeur codé avec des paramètres.
B n'a pas de constructeur codé.
Cela provoque une erreur de compilation.

		Q 14.
1	OUI	
2	NON	

En JAVA, il est possible de créer une instance d'une classe abstraite.

		Q 15.
1	OUI	
2	NON	

Soit les classes quelconques A et B qui héritent de C. C n'est pas une classe abstraite.
Soit la classe Stock contenant l'attribut : ArrayList<C> elements;
On peut écrire le code suivant :

```

elements.add( new A() )
elements.add( new B() )
elements.add( new C() )
    
```

		Q 16.
1	OUI	
2	NON	

En JAVA, une exception est un objet

		Q 17.
1	OUI	
2	NON	

En JAVA, une exception ne peut être utilisée que dans les méthodes privées.

		Q 18.
1	OUI	
2	NON	

Le code suivant est correct :		Q 19.
<pre> public void action(int parametre) { if (parametre==0) throw new Exception("Erreur"); else faireLeTraitement(); } </pre>		
1	OUI	
2	NON	

Soit le code JAVA suivant :		Q 20.
<pre> public void traitement(String nom) { Individu ind=null; try { ind = rechercher(nom); System.out.println(ind.toString()); } catch(NonTrouveException ex) { throw new RuntimeException("non trouve"); } } </pre> <p>avec :</p> <p>la méthode 'rechercher' qui retourne l'exception 'NonTrouveException' si le nom de l'individu n'est pas trouvé.</p> <p>Alors :</p>		
1	si l'individu que l'on veut ajouter n'est pas trouvé alors la méthode retourne l'exception NonTrouveException	
2	si l'individu que l'on veut ajouter n'est pas trouvé alors la méthode retourne l'exception RuntimeException	
3	Si l'individu que l'on veut ajouter n'est pas trouvé alors la méthode ne retourne pas d'exception	

Soit le code suivant :		Q 21.
<pre> try{ System.out.println("AAA"); call(); System.out.println("BBB"); } catch(Exception ex) { System.out.println("DDD"); } catch(MyException ex) { System.out.println("CCC"); } </pre> <p>avec la méthode call qui déclenche l'exception MyException.</p> <p>Ce code affiche :</p>		
1	AAA CCC	
2	AAA DDD CCC	
3	AAA DDD	

En Java, la classe Collections permet de trier les éléments de n'importe quelle collection (classe qui implémente l'interface List) grâce à la méthode Collections.sort(List<T> list)		Q 22.
Cette méthode fonctionne si la classe d'appartenance, ici T, des éléments de la collection implémente l'interface :		
1	Comparator	
2	Comparable	
3	Compare	

En Java, la méthode statique static public void sort(List<E> liste) de la classe ArrayList permet de trier les éléments de la collection <i>liste</i> passée en paramètre		Q 23.
1	OUI	
2	NON	

En Java, pour rechercher un élément dans un tableau de Contact, ArrayList<Contact>, on peut utiliser la méthode prédéfinie public boolean contains . Dans ce cas il faut que :		Q 24.
1	la méthode public int compareTo(Object o) soit implémentée dans la classe Contact	
2	la méthode public boolean comparer(Object o) soit implémentée dans la classe Contact	
3	la méthode public boolean equals(Object o) soit implémentée dans la classe Contact	

En JAVA, pour rechercher un élément la collection Hashset est plus performante que la classe ArrayList		Q 25.
1	OUI	
2	NON	

En JAVA, la classe Hashtable<K,V> est une classe de collection permettant de stocker des éléments de type V qui sont ordonnés par ordre croissant sur l'index K		Q 26.
1	OUI	
2	NON	

En JAVA, la classe File permet de gérer les fichiers et les répertoires.		Q 27.
1	OUI	
2	NON	

Soit le code suivant :		Q 28.
<pre>File f; f = new File("/home/jl", "fic.txt");</pre>		
Ce code crée un fichier de nom "fic.txt" dans le répertoire "/home/jl". Le fichier créé est vide (taille = 0).		
1	OUI	
2	NON	

Le code JAVA suivant, liste les noms des fichiers et les noms des répertoires qui se trouvent dans le répertoire "/home/jl"		Q 29.
<pre>File varfile; varfile = new File("/home/jl"); for(String nom : varfile.list()) System.out.println(nom);</pre>		
1	OUI	
2	NON	

Pour écrire et lire <u>dans un fichier</u> des chaînes de caractère, des entiers, des doubles, ... on peut utiliser les classes prédéfinies <code>DataOutputStream</code> et <code>DataInputStream</code> .		Q 30.
1	OUI	
2	NON	

Pour écrire et lire <u>dans un socket</u> des chaînes de caractère, des entiers, des doubles, ... on peut utiliser les classes prédéfinies <code>DataOutputStream</code> et <code>DataInputStream</code> .		Q 31.
1	OUI	
2	NON	

Le code suivant permet de lire une chaîne de caractère dans la console d'exécution d'un programme Java :		Q 32.
<pre> BufferedReader in = new BufferedReader(new InputStreamReader(System.in)); String str = in.readLine(); </pre>		
1	OUI	
2	NON	

Le socket est un canal de communication permettant de faire communiquer deux JVM qui s'exécutent sur une même machine.		Q 33.
1	OUI	
2	NON	

En JAVA, la classe <code>ServerSocket</code> est une classe qui hérite de <code>Collection</code> . Elle permet de gérer et stocker des sockets.		Q 34.
1	OUI	
2	NON	

En JAVA, la méthode accept de la classe <code>ServerSocket</code> est en attente de la création d'un socket par une application distante. Elle retourne une instance de la classe <code>Socket</code> .		Q 35.
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.*

QUESTION 1 :

Expliquez comment il est possible de créer une collection polymorphe en Java.

QUESTION 2 :

Le principe de communication par socket entre deux applications Java repose sur les classes **Socket**, **ServerSocket**, **DataOutputStream**, **DataInputStream**.

Expliquez, dans ce contexte, le rôle de chacune de ces classes.

QUESTION 3 :

Expliquez le rôle de l'**ajout** d'une nouvelle méthode dans le graphe d'héritage de classes.

Expliquez le rôle de la **surcharge** d'une méthode dans le graphe d'héritage de classes.

FIN DE LA 1^{ère} PARTIE

2^{ème} PARTIE : PROGRAMMATION (avec document)**Durée: 1h15****PROBLEME [50 points]**

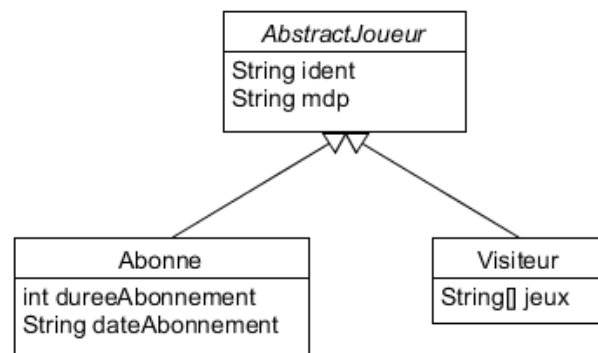
On veut faire évoluer le projet réalisé cette année afin de gérer deux types de joueurs :

- les joueurs abonnés (payant)
- les joueurs visiteurs (non payant)

Les joueurs abonnés sont caractérisés par : son identificateur, son mot de passe, la durée de son abonnement (nombre de mois), la date d'abonnement.

Les joueurs visiteurs sont caractérisés par : son identificateur, son mot de passe, la liste des jeux auxquels il peut jouer (un visiteur est limité).

On obtient l'arborescence de classe suivante :



La classe de plate-forme de jeu PFJeu gère la liste des joueurs (collection polymorphe) :

ArrayList<AbstractJoueur> joueurs;

Le fichier texte contenant la liste des joueurs est dans le format suivant :

```

ABONNE;jack31;1mdptdap;12;12/02/2018
ABONNE;killer;dougDoug;6;03/12/2017
VISITEUR;JeanDupont;$aze_2;Othe1lo;Puissance4
VISITEUR;Claude;$j132;Go;Puissance4;Othe1lo
  
```

1/ Ecrire les classes **AbstractJoueur [6 points]**, **Abonne [6 points]**, **Visiteur [6 points]** permettant de répondre aux questions 2/ et 3/

2/ **[15 points]** Ecrire la méthode de la classe PFJeu : **public void importerJoueur(String nomFichier)** qui initialise la liste des joueurs avec le contenu du fichier texte passé en paramètre.

Rappel : le code `String[] champs = ligne.split(";")` retourne dans `champs` tous les mots entre les ; et `champs.length` est le nombre de champs décodés.

3/ **[2 points]** (les méthodes `toString` sont notées dans 1/) Ecrire la méthode **public String listerJoueurs()**; qui retourne sous la forme d'une chaîne de caractères tous les joueurs (valeurs de tous les attributs).

4/ **[15 points]** Ecrire la méthode de la classe PFJeu qui retourne les joueurs abonnés dont l'abonnement a expiré **public ArrayList<AbstractJoueur> abonneesExpirees()**; Pour cela vous utilisez les deux méthodes prédéfinies suivantes :

public static long differenceJours(String date1, String date2) qui retourne le nombre de jours entre `date1` et `date2`;

public static String aujourd'hui(); qui retourne la date d'aujourd'hui.

Nous considérons qu'un mois fait 30 jours.

NB : tous les attributs sont privés. Vous n'avez pas besoin d'écrire les getteurs et les setteurs des attributs. Vous considérez qu'ils existent pour tous les attributs.

(Fin du sujet)