IPST-CNAM
Programmation JAVA
NFA 032
Septembre 2017

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

2ème 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.

1ère 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
  - –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.

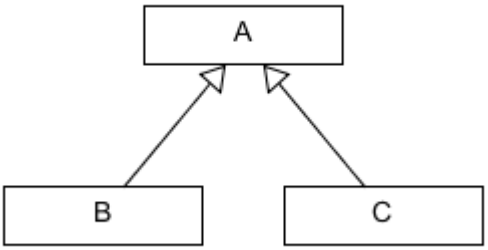
N'oubliez pas d'inscrire en en-tête du QCM, votre nom et prénom.

Vous avez droit à **4 points** négatifs sans pénalité.

NOM:	PRENOM:

En programmation objet, le principe d'héritage permet de spécialiser une classe en créant une classe dérivée qui surcharge certaines des méthodes de la classe d'héritage		Q 1.
1	OUI	
2	NON	

Une classe abstraite peut hériter d'une autre classe abstraite		Q 2.
1	OUI	
2	NON	

Soit le symbolisme UML suivant :  Cela signifie que B et C héritent de A.		Q 3.
1	OUI	
2	NON	

Soit le code suivant :: `public class A extends C` `{` `  public void faire() {` `  B b;` `  b.x = 100;` `  y = 200;` `  }` `}` Les classes A, B et C sont toutes des classes public.		Q 4.
1	si x est un attribut public de B et y est un attribut protected de C alors ce code est correct	
2	si x est un attribut pubic de B et y est un attribut privé de C alors ce code est correct	
3	si x est un attribut privé de B et y est un attribut public de C alors ce code est correct	

Une classe A peut accéder directement à un attribut d'une autre classe B si A hérite de B et que cet attribut est déclaré protected		Q 5.
1	OUI	
2	NON	

Si B hérite de A, alors A doit définir un constructeur		Q 6.
1	OUI	
2	NON	

On a une classe A qui contient uniquement le constructeur suivant : `public A(int x){ }` On a la classe B qui hérite de A, et qui contient le constructeur suivant : `public B(){` `  attr = 100;` `}` Ce code est correct		Q 7.
1	OUI	
2	NON	

Soit une classe B qui hérite d'une classe A. Soit A et B qui possèdent toutes les deux un constructeur sans paramètre. Lors de la création d'un objet de B, automatiquement, le constructeur de A est invoqué.		Q 8.
1	OUI	
2	NON	

En programmation objet, une collection polymorphe est une collection dont les éléments sont d'un type donné. En JAVA, ce type peut être une classe quelconque.		Q 9.
1	OUI	
2	NON	

Soit une classe abstraite C qui contient le constructeur suivant : `public C () { }` Il est possible d'écrire le code suivant : C o = new C();		Q 10.
1	OUI	
2	NON	

Une classe abstraite peut avoir un constructeur.		Q 11.
1	OUI	
2	NON	

Tous les attributs d'une classe abstraite doivent être **protected**.		Q 12.
1	OUI	
2	NON	

Soit trois classes A, B et C qui héritent de la classe abstraite H et H implémente l'interface I. Pour créer une collection polymorphe, on peut faire ArrayList<A,B,C> elements		Q 13.
1	OUI	
2	NON	

Soit les classes quelconques A et B qui héritent de C. C est une classe abstraite. Soit la classe Stock contenant l'attribut : ArrayList<**C**> elements; On peut écrire : `elements.add( new A() )` `elements.add( new B() )` `elements.add( new C() )`		Q 14.
1	OUI	
2	NON	

Soit une interface Java **InterfaceEx** et soit une méthode de la classe **A** dont la signature est la suivante : `public String unTraitement(InterfaceEx i)` La classe A doit implémenter les méthodes de l'interface **InterfaceEx**		Q 15.
1	OUI	
2	NON	

En JAVA, une exception est un objet à part entière qui est une instance de la classe prédéfinie **Exception** ou une instance d'une classe qui hérite de la classe **Exception**.		Q 16.
1	OUI	
2	NON	

En JAVA, une exception est un objet dont la classe d'appartenance hérite de la classe **RuntimeException**		Q 17.
1	OUI	
2	NON	

Soit le code suivant : ``` public void action(int parametre) { if (parametre==0) throw new Exception("Erreur"); else faireUnTraitement(); } ``` Ce code est correct.		Q 18.
1	OUI	
2	NON	

En JAVA, l'instruction suivante permet de déclencher une exception `throw new RuntimeException("Impossible de faire l'action");`		Q 19.
1	OUI	
2	NON	

La déclaration de la méthode suivante : `public void traitement(String s) throws MyException` précise que la méthode est susceptible de retourner l'exception 'MyException'.		Q 20.
1	OUI	
2	NON	

Soit le code suivant : Q 21.

```
try{
   System.out.println("AAA");
   call();
   System.out.println("BBB");
}
catch(MyException ex) {
   System.out.println("DDD");
}
catch(Exception ex) {
   System.out.println("CCC");
}
```

avec la méthode call qui déclenche l'exception **UneAutreException**.

Ce code affiche :

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 Q 22.
Collections.*sort*(List<T> list)
Cette méthode fonctionne si la classe d'appartenance, ici T, des éléments de la collection implémente l'interface Comparator

1	OUI
2	NON

En JAVA, une classe qui implémente l'interface List<E> est une collection. Q 23.

1	OUI
2	NON

En Java, une liste chainée (ex: classe LinkedList) est une liste dont les éléments ne sont pas continus dans la mémoire de la JVM. : Q 24.

1	OUI
2	NON

La classe AbstractList gère des collections dont les éléments sont ordonnés suivant un index d'insertion. Q 25.

1	OUI
2	NON

En Java, on déclare un tableau de la manière suivante : **I tab[] = new B[100]** où **I** est une interface et B implémente l'interface I. Q 26.

1	Cela n'est pas possible
2	Cela est possible et on peut écrire : **t[i]=new I();**
3	Cela est possible et on peut écrire : **t[i]=new B();**

Soit la déclaration suivante : ArrayList mes_elements;		Q 27.
1	Ce code est correct : `Individu ind = new Individu("LAFONT","Pierre");` `mes_elements.add(ind);` car Individu hérite de Object	
2	Ce code est correct : `Individu ind = new Individu("LAFONT","Pierre");` `mes_elements.add((Object) ind);` car Individu hérite de Object	
3	Ce code est correct : `Object ind = new Object("LAFONT","Pierre");` `mes_elements.add(ind);` car Individu hérite de Object	

En Java, la classe Hashtable est une classe de définition d'une Collection qui permet l'accès à ses éléments via une chaine de caractère.		Q 28.
1	OUI	
2	NON	

En Java, pour écrire dans un fichier ou pour écrire dans un Socket, on peut utiliser les mêmes méthodes d'écriture et de lecture.		Q 29.
1	OUI	
2	NON	

En Java, la classe Socket permet, entre autre, de parcourir les fichiers contenus dans un répertoire.		Q 30.
1	OUI	
2	NON	

Le code suivant crée un fichier de nom "exemple.txt" dans le répertoire père du répertoire courant d'exécution. Le fichier est créé, vide de toute information `FileInputStream fis = new FileInputStream(new File("exemple.txt"));` `DataInputSream dis = new DataInputStream(fis);` `dis.close()`		Q 31.
1	OUI	
2	NON	

Soit un serveur qui réalise le traitement suivant : `ServerSocket ssoc = new ServerSocket(9100);` `Socket soc = ssoc.accept();` `DataInputStream dis= new DataInputStream(soc.getInputStream());` `System.out.println(dis.readUTF());` Pour afficher sur le serveur la valeur "HELLO", le client doit écrire le code suivant : `Socket soc = new Socket("localhost",9100);` `OutputStream os=soc.getOutputStream();` `PrintStream ps=new PrintStream(os);` `ps.print("HELLO");` `soc.close();`		Q 32.
1	OUI	
2	NON	

En JAVA, le socket est un moyen permettant d'échanger des données entre des programmes informatiques qui s'exécutent sur une même machine.		Q 33.
1	OUI	
2	NON	

Soit un programme Java P1 (JVM) qui crée un serveur de socket sur le port 9100. Soit un programme Java P2 (JVM) qui crée un serveur de socket sur le port 9100. Le programme P1 s'exécuter sur la machine A, et le programme P2 doit s'exécuter sur une autre machine B.		Q 34.
1	OUI	
2	NON	

En JAVA, sur un socket, on ne peut écrire que des chaînes de caractères.		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 la différence qui existe entre l'utilisation de la classe **Exception** et l'utilisation de la classe **RuntimeException.**

QUESTION 2 :

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

QUESTION 3 :

Quel est le rôle de la classe Socket de Java ? Expliquez son fonctionnement.

FIN DE LA 1ère PARTIE

2ème PARTIE : PROGRAMMATION (avec document)
Durée: 1h15

PROBLEME 2 [50 points]

On se propose de créer un programme Java qui gère une collection de produit.

Un produit est caractérisé par :
- la référence du produit (String)
- la quantité du produit dans le stock (int)
- la description du produit (String)

Ce programme affiche une IHM (et une seule) permettant :
- d'ajouter un nouveau produit dans le stock.
- d'incrémenter (+1) un produit existant du stock.
- de décrémenter (-1) un produit existant du stock.
- de rechercher des produits en recherchant un morceau de chaine dans les descriptions des produits. Si la chaine à rechercher est vide alors l'ihm affiche tous les produits.

La référence du produit est de la forme : <Type>-<Numéro>-2017

Type est une des valeurs d'une liste de chaîne de caractères initialisée en dur dans le programme (exemple : final static String[] types_produit = { "LIVRE", "DVD", "PAPETERIE", "BD" }

Numéro est un numéro unique, automatiquement géré par le programme.

Ecrire la classe **IHMStock**, la classe **GestionProduit** et la classe **Produit**.

La classe **IHMStock** affiche l'IHM. Elle appelle les méthodes de la classe **GestionProduit** qui contient la collection et qui implémente les méthodes de gestion du stock appelées par l'IHM. La classe **Produit** est la classe qui définit un produit.

Au lancement du programme la collection est vide.

Inutile d'écrire les getteurs et les setteurs.

NB : Pour créer les éléments graphiques de l'IHM, vous utiliser la classe Formulaire, vue en cours et en TP.

(Fin du sujet)