Exercices de Révision

Interfaces :

1/ En JAVA, une classe peut implémenter plusieurs interfaces ? OUI / NON

2/ En JAVA, une interface est un moyen de créer des traitements génériques ? OUI/NON

3/ En Java, l' Interface permet de :

- gérer des collections polymorphes (les éléments sont de type d'une interface)
- gérer les fichiers (interface avec le système d'exploitation)
- de passer en paramètre d'une méthode "un traitement" (traitements génériques)

Exceptions :

1/ La déclaration de la méthode suivante : public void traitement(String s) throws MyException précise que la méthode doit capturer l'exception MyException dans le corps de sa méthode : OUI / NON

2/Soit le code JAVA suivant :

```
public void traitement(String nom) throws Exception
{
   Individu ind=null;
   try {
        ind = rechercher(nom);
        System.out.println(ind.toString());
   }
   catch(NonTrouveException ex) {
      throw new Exception("non trouve"); }
}
avec :
    la    méthode    'rechercher'    qui    retourne    l'exception
'NonTrouveException' si le nom de l'individu n'est pas trouvé.
```

- Si l'individu que l'on veut ajouter n'est pas trouvé alors la méthode retourne l'exception NonTrouveException
- Si l'individu que l'on veut ajouter n'est pas trouvé alors la méthode retourne l'exception Exception
- Si l'individu que l'on veut ajouter n'est pas trouvé alors la méthode ne retourne pas d'exception

3/ L'exception JAVA RuntimeException est une exception qui est retournée quand le moteur de la JVM (Runtime) plante ? OUI/NON

4/ En JAVA, la classe RuntimeException est une exception pour laquelle la méthode qui la déclenche n'a pas besoin de préciser qu'elle propage cette exception (l'usage de throws est inutile) ? OUI/ NON

5/ Soit le code suivant :

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

avec la méthode call qui déclenche l'exception MyException, ce code affiche :

- AAA CCC
- AAA CCC DDD
- AAA DDD

6/ En JAVA, une exception est une classe qui hérite toujours directement ou indirectement de la classe Exception ? OUI/ NON

Listes :

1/ En Java, une liste chainée (ex: classe LinkedList) est une liste dont les éléments ne sont pas continus dans la mémoire de la JVM ? OUI/ NON

2/ En Java, une liste non chainée (ex: ArrayList) est une liste dont les éléments sont continus dans la mémoire de la JVM ? OUI/ NON

3/ En Java, les différences entre une liste chainée et une liste non chainée sont :

- Il est plus rapide dans une liste chainée d'ajouter un élément en début de la collection que dans une liste non chainée.
- Il est plus rapide de parcourir tous les éléments dans une liste chainée que dans une liste non chainée.
- Pour un même nombre d'élément, la liste chainée est plus gourmande en mémoire que la liste non chainée.

4/ Soit une collection "liste" définie par la classe ArrayList<Individu>. Nous voulons trier les éléments de cette liste suivant 3 critères de tri différents. Pour cela :

- la classe Individu implémente l'interface Comparable et on code dans la méthode compareTo les 3 critères de tri. Le choix du tri se fait par la valeur d'un attribut statique.
- la classe Individu implémente 3 fois l'interface Comparable.
- pour chacun des tris faire les appels : Collections.sort(liste, comparator1) Collections.sort(liste, comparator2) ou Collections.sort(liste, comparator3) où comparator1, comparator2, comparator3 sont des instances des classe

Comparator1, Comparator2, Comparator3 qui implémentent la méthode compare de l'interface Comparator<Individu>

5/ Comment ajoute-t-on un élément à la fin d'une ArrayList nommée 'liste' ?

- liste.insert(valeur)
- liste.add(valeur)
- liste.push(valeur)
- liste[last] = valeur

6/Pour une ArrayList, quelle méthode retourne le nombre d'éléments ?

- length
- capacity()
- count()
- size()

7/ Comment vide-t-on complètement une ArrayList ?

- liste.clear()
- liste.delete()
- liste.empty()
- liste.reset()

8/ Comment modifie-t-on la valeur à l'index 1 d'une ArrayList ?

- liste.set(1, nouvelleValeur)
- liste[1] = nouvelleValeur
- liste.update(1, nouvelleValeur)
- liste.replace(1, nouvelleValeur)

9/ Que retourne la méthode liste.isEmpty() ?

- Le nombre de cases vides
- La valeur du premier élément
- Une erreur si la liste est pleine
- Un booléen (true/false)

Exercice 2 : Soit le code suivant, déterminer les erreurs qui existent dans ce code et comment peut-on les corriger ?

```
import java.util.LinkedList;
public class GestionListe {
public static void main(String[] args) {
LinkedList<String> noms = new LinkedList<>();

noms.add("Alex");
noms.add("Sara");
noms.add("Youssef");

afficherPremier(noms);
afficherElement(noms, 3);
supprimerNom(noms, "Omar");
ajouterAge(noms);
}

public static void afficherPremier(LinkedList<String> liste) {
System.out.println("Premier élément : " + liste.getFirst());
}

public static void afficherElement(LinkedList<String> liste, int index) {
System.out.println("Élément à l'index " + index + " : " + liste.get(index));
}

public static void supprimerNom(LinkedList<String> liste, String nom) {
liste.remove(nom);
System.out.println(nom + " supprimé avec succès.");
}

public static void ajouterAge(LinkedList<String> liste) {
liste.add(25);
System.out.println("Âge ajouté.");
}
}
```

Exercice 3 :

1/ Donner une définition précise de ce qu'est une " exception ". Quels sont les mots cl importants d'utilisation des exceptions en Java ? Donnez, en une phrase, le rôle de chacun de ces mots clé.

2/ Donner une définition précise de ce qu'est une " interface ". Donnez deux exemples d'utilisation d'une interface (Vous pouvez illustrer ces deux exemples avec du code Java).

Exercice 4 :

Soit les 3 situations suivantes, indiquez à chaque fois quelle structure de données doit-on utiliser et justifiez votre réponse :

- 1ére situation : On souhaite créer un système de gestion de formes géométriques, on peut avoir plusieurs formes : triangle, rectangle etc. Chaque forme doit calculer son aire et doit être affichée.
- 2éme situation : Un enseignant gère une liste d'étudiants, il accède à chaque étudiant par son index, il apporte des modifications dans les bonus et les notes.
- 3éme situation : Dans un train, chaque wagon est relié au wagon précédent et au wagon suivant, on souhaite rajouter ou enlever un wagon au milieu.