Corrigé Interrogation d'informatique
2ème année - Mars 2018

Durée totale : 1h
Documents autorisés : Fiche de synthèse A4 recto/verso.
Les trois exercices sont indépendants
Ils peuvent être traités dans n'importe quel ordre même si l'exercice 3 utilise le code de l'exercice 2

Exercice 1 : for et for each (5pts)

On considère le code suivant (que l'on suppose intégré dans le main d'une classe) :

```
LinkedList<Integer> maListe = new LinkedList<Integer>();
int nbElt = 1000;
for(int i=0; i<nbElt; i++)
      maListe.add(i);

int nRep = 10000;
int s=0;

long tempsAvant = System.currentTimeMillis();
for (int j=0; j<nRep; j++){   // Boucle 1
      for(int v : maListe){
          s += v;
      }
}
long tempsApres = System.currentTimeMillis();
System.out.println("Duree parcours 1 : " + (tempsApres-tempsAvant) + " ms");

s=0;
tempsAvant = System.currentTimeMillis();
for (int j=0; j<nRep; j++){ // Boucle 2
      for(int i=0; i<maListe.size(); i++){
          s += maListe.get(i);
      }
}
tempsApres = System.currentTimeMillis();
System.out.println("Duree parcours 2 : " + (tempsApres-tempsAvant) + " ms");
```

L'exécution du programme conduit à l'affichage de deux temps d'exécution, un pour la Boucle 1 et l'autre pour la Boucle 2.

(Q1-1 – 1 pt) Quelle est la boucle dont l'exécution est la plus rapide ?

La boucle 1 est la plus rapide

(Q1-2 – 2 pts) Justifiez brièvement en expliquant les opérations qui sont effectuées par les méthodes utilisées dans chaque boucle.

Le get(i) part du début de la liste et fait i opérations avant d'obtenir la valeur du ième élement. Cela donne une complexité en 0(n^2) (1+2+3+4+…n).

(Q1-3 – 2 pts) Si `maListe` était un `ArrayList`, pensez-vous que l'exécution de la boucle 2 aurait pris plus de temps, ou moins ? Justifiez.

L'exécution aurait été plus rapide car dans ce cas get(i) est un accès direct en 0(1)

Exercice 2 : Modélisation de bibliothèque (7pts)

On modélise les livres gérés par une bibliothèque. Dans cette modélisation très simplifiée, on mémorise le titre et le nombre de fois où chaque livre a été emprunté. La classe `Livre` est alors définie comme suit :

```
public class Livre {
      private String titre;   // titre du livre
      private int nPrets;     // nombre de fois où le livre a été emprunté

      public Livre(String untitre, int lesPrets){ // Constructeur
            titre = untitre;
            nPrets = lesPrets;
      }

      public String toString(){ // Affichage
            return "[" + titre + " / " + nPrets +"]";
      }

      public String getTitre(){ // Renvoie le titre
            return titre;
      }

      public int getPrets(){ // Renvoie le nombre de prêts
            return nPrets;
      }

      public void setPrets(int prets){ // Met à jour le nombre de prêts
            nPrets = prets;
      }
}
```

Une bibliothèque a un nom et un fond (l'ensemble des livres de la bibliothèque). **On considère que le fond d'une bibliothèque ne peut posséder qu'un seul exemplaire de chaque livre**. La classe `Bibliotheque` est alors définie comme suit:

```
public class Bibliotheque {
      private String nom;
      private TreeSet<Livre> fond;

      public Bibliotheque(String nomBib){ // Constructeur standard
            nom = nomBib;
            fond = new TreeSet<Livre>();
      }


      public Bibliotheque(String nomBib, String nomFichier){
      /* Ce constructeur crée la bibliothèque et remplit le fond.
      On suppose que la méthode readFromFile de la classe « Outils » renvoie
      un TreeSet à partir du fichier dont le nom est passé en paramètre. */

            nom = nomBib;
            fond = Outils.readFromFile(nomFichier);
      }


      public void ajouterLivre(Livre livre){ // Ajoute un livre au fond
```

```
            fond.add(livre);
      }

      public String toString(){ // Affichage
            String res= "[";
            for(Livre l : fond){
                  res+=" " + l;
            }
            return res+" ]";
      }

      public String getNom(){ // Renvoie le nom
            return nom;
      }

      public TreeSet<Livre> getFond(){ // Renvoie le fond
            return fond;
      }
}
```

(Q2.1 – 2 pts) Quelle est la propriété de `TreeSet` qui fait que la classe **`Bibliotheque`** correspond à ce qui est écrit dans l'énoncé (en gras)?

TreeSet est un Set, donc sans doublon, ce qui correspond à l'énoncé

Pour avoir le droit d'utiliser `TreeSet`, il manque deux choses à la classe **`Livre`** : une méthode `compareTo` et une interface standard Java.
(Q2.2 – 2 pts) Quelle est cette interface ? Modifiez en conséquence le code de la classe **`Livre`**.

C'est l'interface Comparable

```
public class Livre implements Comparable<Livre> {
```

(Q2.3 – 3 pts) Proposez une implémentation de la méthode `compareTo` de la classe **`Livre`** qui classe les livres par ordre alphabétique, sans se préoccuper du nombre de prêts.

Pour vous aider, on vous rappelle que la comparaison par ordre alphabétique de deux String A et B est réalisée par la méthode `A.compareTo(B)` qui :

- renvoie un entier négatif si A est située avant B par ordre alphabétique (A<B)
- renvoie 0 si A et B sont identiques (A=B)
- renvoie un entier positif si A est située après B par ordre alphabétique (A>B).

```
public int compareTo(Livre autreLivre){
      return titre.compareTo(autreLivre.titre);
}
```

Exercice 3 : Fusion de bibliothèques (8pts)

On souhaite pouvoir réaliser la fusion cohérente des fonds de 2 bibliothèques modélisées comme dans l'exercice 2. On considère dans ce but 2 méthodes `bibDiff` et `bibInter`. Leurs en-têtes complets sont :

```
public TreeSet<Livre> bibDiff( Bibliotheque autreBib )
public TreeSet<Livre> bibInter( Bibliotheque autreBib )
```

On suppose que la méthode `compareTo` est implémentée dans la classe Livre (comme demandé à Q2.3) et que `L1.compareTo(L2)` renvoie un entier négatif si le titre de L1 est avant le titre de L2 dans l'ordre alphabétique, 0 si c'est le même titre (égalité), un entier positif sinon.

(Q3.1 – 3 pts) La méthode `bibDiff` renvoie l'ensemble des livres présents dans la bibliothèque et qui ne sont pas présents dans la 2ème bibliothèque `autreBib`, passée en paramètre.

Ecrivez la méthode `bibDiff` .

```
public TreeSet<Livre> bibDiff( Bibliotheque autreBib ){
    TreeSet<Livre> listDiff = new TreeSet<Livre>();
    for( Livre ref : fond ){
        if (!autreBib.fond.contains(ref)){
            listDiff.add(ref);
        }
    }
    return listDiff;
}
```

(Q3.2 – 3 pts) La méthode `bibInter` renvoie l'ensemble des livres présents dans la bibliothèque et qui sont aussi présents dans la 2ème bibliothèque `autreBib`, passée en paramètre. Pour chaque couple de livres identiques trouvé, elle choisit le livre ayant le nombre de prêts le plus faible.

Ecrivez la méthode `bibInter` .

```
public TreeSet<Livre> bibInter( Bibliotheque autreBib ){
    TreeSet<Livre> listInter = new TreeSet<Livre>();
    for( Livre ref : fond ){
        for( Livre test : autreBib.fond ){
            if( ref.compareTo(test) == 0 ){
                if( ref.nPrets <= test.nPrets ){
                    listInter.add(ref);
                }else{
                    listInter.add(test);
                }
            }
        }
    }
    return listInter;
}
```

(Q3.3 – 2 pts) Le programme principal réalise la fusion de 2 bibliothèques `bibA` et `bibB` puis l'affiche.

À l'aide des méthodes précédentes, compléter le programme principal donné ci-dessous :

```
public static void main(String[] args){
      Bibliotheque bibA = new Bibliotheque("bibA", "refbib.data");
      Bibliotheque bibB = new Bibliotheque("bibB", "testbib.data");
      Bibliotheque bibFusion;
                              … A COMPLETER (Q3.3) …
      System.out.println("Fusion des bibliothèques : " + bibFusion);
}
```

```
TreeSet<Livre> AdiffB = bib_A.bibDiff(bib_B);
TreeSet<Livre> BdiffA = bib_B.bibDiff(bib_A);
TreeSet<Livre> AinterB = bib_A.bibInter(bib_B);

bib_fusion = new Bibliotheque("bib_fusion");
for( Livre a : AdiffB ) bib_fusion.fond.add(a);
for( Livre a : BdiffA ) bib_fusion.fond.add(a);
for( Livre a : AinterB ) bib_fusion.fond.add(a);
```

(dans le code, il faut lire bibA au lieu de bib_A, de même bibB et bibFusion)

bibFusion.fond.add(a) peut bien sûr aussi être fait par bibFusion.ajouterLivre(a)