Programmation Orientée Objet
Contrôle terminal - 2017-2018

(le barème est donné à titre indicatif)

I. QUESTION DE COURS (3 points)

Illustrez, à partir d'un diagramme de classes, la notion de polymorphisme. Donnez les 3 concepts nécessaires à la mise en œuvre du polymorphisme. Donnez un court exemple de code illustrant ces concepts reprennent les classes de votre exemple.

II. EXERCICE (5 pts)

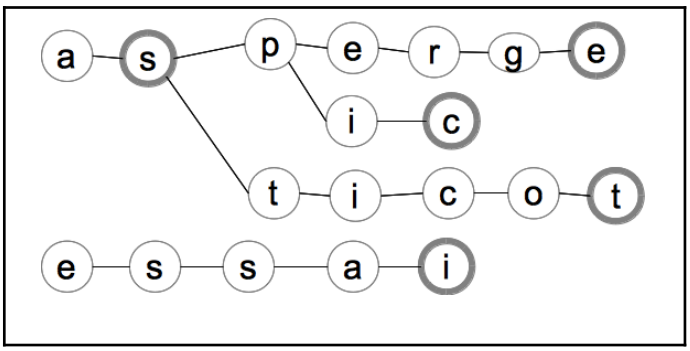
```
public class Machin {
  public  String methode1(String s) {
    int longueur = s.length();
    String resultat = new String();

    for (int i = 0; i < longueur; i++) {
      resultat = resultat + s.charAt(longueur - i - 1);
    }
    return resultat;
  }
  public  boolean methode2(String chaine) {
    String autre = methode1(chaine);
    return (autre.equals(chaine));
  }
  public static void main(String[] arg) {
    Machin machin = new Machin();
    String test = "ada";
    System.out.println(machin.methode2(test));
    test = "java";
    System.out.println(machin.methode2(test));
  }
}
```

Que fait la classe Machin ?
Pour vérifier le rôle de la classe, faites la trace de l'exécution et donnez le résultat à partir des appels donnés dans le main et indiquez son rôle en une phrase courte.

II. PROBLÈME (15 pts)

On souhaite écrire une classe permettant de créer un **dictionnaire** et d'ajouter des mots stockés à l'aide d'une structure ***récursive***. On considère qu'un mot est constitué d'un caractère suivi de la suite du mot. Quand on atteint le dernier caractère du mot, on signale que ce caractère est terminal. Dans le schéma, un caractère terminal est entouré d'un cercle plus épais.

Dans l'exemple ci-dessous, les mots ajoutés dans cette structure de données sont : «as», «asperge», «aspic», «asticot», et «essai».
On constate que tous les mots suivant le caractère initial 'a' commencent par le caractère 's', qui est aussi un caractère terminal. On constate également que ce caractère 's' est suivi de deux mots commençant par 'p' et 't'. Ce caractère 'p' est suivi de deux mots commençant par 'e' et 'i', etc.



1. Complétez le schéma ci-dessus pour donner le résultat de l'ajout des mots : «art», «artiste», «astre», «est».

```
public interface IDico {
	/**
	 * Recherche la présence d'un mot dans un dictionnaire
	 * @param word mot à rechercher
	 * @return true si trouvé, false sinon
	 */
	public boolean containsWords(String word);
	/**
	 * Insère un mot dans le dictionnaire si non déjà présent
	 * @param word mot à insérer
	 * @throws Exception exception si le mot est déjà présent
	 */
	public void addWord(String word) throws Exception;
}
```

On considère l'interface suivante :

2. Ecrire la déclaration de la classe *Dico* qui a comme attributs un caractère, un booléen permettant de signaler que ce caractère est le dernier d'un mot, et une collection (**ArrayList**) de *Dico* correspondant à toutes les suites de mots ajoutés dans le dictionnaire. Ajouter un constructeur qui initialise le caractère à **'\0'** (caractère **nul**) et la collection à vide.
   Surcharger le constructeur qui initialise le caractère à partir d'un caractère passé en paramètre, le booléen à faux et la collection à vide.
3. Ajouter à la classe *Dico* le code de la méthode *int indexOfChar(char c)* qui recherche dans la collection. Cette méthode retourne l'index de l'instance de *Dico* contenant le caractère passé en paramètre si le caractère a été trouvé, et retourne -1 si le caractère n'a pas été trouvé.

4. Ajouter à la classe *Dico* le code de la méthode *int insertChar(char c)* qui ajoute à la collection, en respectant l'ordre alphabétique, une instance de *Words* contenant ce caractère, et retourne son index. Attention, si la collection contient déjà une instance de *Words* contenant ce caractère, la méthode se contente de retourner son index, sans modifier la collection.
5. Ecrire le code de méthode récursive *boolean findWord(String word)* qui recherche la présence du mot *word*. Si mot est de longueur 0 alors il n'a pas été trouvé. S'il est de longueur > 0, utiliser la méthode *indexOfChar(char c)* pour vérifier la présence du premier caractère du mot dans la collection. Si ce premier caractère est bien présent, relancer la recherche sur le reste du mot.

```
String s = word.substring(1);  //renvoie la sous-chaîne privée du premier caractère.
char c = word.charAt(0);   // revoie le premier caractère de la chaîne.
```

On considère qu'on dispose dans la classe *Dico* du code de la méthode permettant d'insérer un mot *void addWord(String word)*. On dispose également de la méthode *String toString()*.

6. Pour tester la classe *Dico*, écrire le code de la méthode main qui utilise la classe *Dico* et qui ajoute dans le dictionnaire les mots de l'exemple ci-dessus.
7. Ajouter à la classe *Dico* le code de la méthode *void addWord(String word)*. Cette méthode renvoie une exception si le mot est déjà présent. Utiliser les méthodes ci-dessus pour tester si le premier caractère du mot est déjà présent et s'il n'est pas présent, pour le rajouter dans la collection du *Dico* courant. Si c'est le seul et dernier caractère du mot, modifier le booléen. Si ce n'est pas le dernier caractère du mot, rappeler récursivement la méthode *addWord* pour ajouter le reste du mot.
8. Vérifier le code de vos classes avec les exemples ci-dessus.

III. ANNEXE

```
public class ArrayList<T> {
    // retourne la taille de la collection
    public int size() {…};
    // ajoute en fin de la collection
    public boolean add(T x) {…};
    // ajoute x à l'index i, et décale les index des val suivantes
    public boolean add(int i, T x) {…};
    // modifie la valeur du ième élément
    public void set(int i, T x) {…};
    // retourne la valeur du ième élément
    public T get(int i) {…};
    // supprime le ième élément et décale les index des val suivantes
    public void remove(int i) {…};
    ...
}
public class String {
     // retourne le caractère à la position donnée par index
     public char charAt(int index);
     // retourne la longueur de la chaîne
     public int length();
     // retourne la sous-chaîne s[index..s.length()-1]
     public String substring(int index)
     ...
}
```