Contrôle de Programmation

Jeudi 7 janvier 2016
Durée : 2h

Ce sujet comporte 3 pages.
Documents autorisés : notes de cours, TD et TP, spécifications des types abstraits.

1 Répertoire téléphonique.

Le but de cet exercice est de permettre à une entreprise de gérer les répertoires téléphoniques de ses employés.

Un répertoire téléphonique permet de mémoriser le nom et le numéro de téléphone de personnes appelées *contacts*. Nous considérons qu'il n'y a pas d'homonymes : ainsi le nom permet d'identifier de manière unique un contact.

Un répertoire téléphonique dispose des opérations suivantes :

```
public interface Repertoire
{
  // déterminer si un nom est présent dans le répertoire
  public boolean estPresent(String nom);

  // obtenir le numéro d'un contact
  // @return null si nom absent
  public String getNumero(String nom);

  // enregistrer un nouveau contact
  // @post sans effet si nom présent
  public void enregistrer(String nom, String numero);

  // modifier le numéro d'un contact
  // @post sans effet si nom absent
  public void modifier(String nom, String numero);

  // supprimer un contact du répertoire
  // @post sans effet si nom absent
  public void supprimer(String nom);
}
```

Implémentation avec une page simple

On considère une première sorte de répertoire téléphonique : une *simple page* sur laquelle les nouveaux contacts sont ajoutés *à la fin de la page*.

Pour cette implémentation on décide d'utiliser une *liste chaînée* (type LinkedList<E>) dont les éléments sont de type Contact (voir Annexe).

Question 1. Donnez la déclaration de la classe PageSimple avec son/ses attribut(s) et programmez un constructeur sans paramètre qui initialise un répertoire vide.

Question 2. Programmez la méthode (protected) chercher qui cherche un nom donné dans le répertoire ; cette méthode renvoie null si le nom cherché est absent ; s'il est présent, elle renvoie un itérateur it tel que it.previous() donne l'élément trouvé.

Question 3. Programmez la méthode estPresent.

Question 4. Programmez la méthode getNumero.

Question 5. Programmez la méthode enregistrer.

Question 6. Programmez la méthode modifier.

Question 7. Programmez la méthode supprimer.

Implémentation avec une page classée

Dans cette implémentation les contacts sont rangés dans l'ordre alphabétique de leur nom, dans une liste de type ArrayList<E>.

Question 8. Indiquez les avantages et inconvénients de ce choix.

Question 9. Parmi les 6 méthodes de la classe PageClassée indiquez celles qui diffèrent des méthodes de la classe PageSimple en *expliquant ce qui change.*

Implémentation avec un carnet

Un carnet est composé de 26 pages ; chaque page correspond à une lettre de l'alphabet et permet de ranger les contacts dont le nom commence par cette lettre. On supposera définie la méthode
protected int numeroPage(String nom);
qui donne le numéro de page ($\in [0 \cdots 26[$) correspondant à la première lettre de la chaîne nom.

Question 10. Donnez la déclaration de la classe Carnet avec son/ses attribut(s) et programmez un constructeur sans paramètre qui initialise un répertoire vide.

Question 11. Indiquez quelles sont les méthodes de la classe Carnet qui diffèrent de celles de PageSimple en *expliquant ce qui change.*

Question 12. Programmez la méthode enregistrer.

Question 13. Dessinez un diagramme de classes (sans attribut ni méthode) qui fait apparaître les entités ci-dessus et les relations qui les lient.

Mutualisation

Question 14. Faites un nouveau diagramme en ajoutant une classe dont le rôle est de mutualiser le plus grand nombre de fonctionnalités des trois classes PageSimple, PageClassée et Carnet ; quelle est la particularité de cette classe ?

Question 15. Donnez la déclaration de cette nouvelle classe et indiquez :

- les méthodes qui *ne peuvent pas* être programmées dans cette classe ;
- les méthodes qui *peuvent* être programmées dans la classe.

Question 16. Modifiez la déclaration de la classe Carnet pour l'adapter à la nouvelle organisation puis listez les méthodes qui *doivent* être programmées dans cette classe.

Question 17. Expliquez à l'aide d'un nouveau diagramme ce qu'il faut modifier dans la hiérarchie de classes pour exprimer qu'un carnet peut être composé de pages simples ou de pages classées mais pas de carnets.

Annexe

class Contact

```
public class Contact {
 // ...
 public Contact(String p_nom, String p_numero) { ... }
 public String getNom()                         { ... }
 public String getNumero()                      { ... }
 public void   setNumero(String p_nouveau)      { ... }
 public String toString()                       { ... }
}
```

Interface ListIterator<E>

Ci-dessous, les commentaires de spécification en français de quelques opérations importantes.

```
// opérations de parcours
// Renvoie vrai s'il reste un élément lors d'un parcours "en marche avant"
boolean         hasNext();

// renvoie le prochain élément de la liste et avance le curseur
// @pre : hasNext() est vérifié
E       next();

// Renvoie vrai s'il reste un élément lors d'un parcours "en marche arrière"
boolean         hasPrevious();

// renvoie l'élément précédent de la liste et recule le curseur
// @pre : hasPrevious() est vérifié
E       previous();

// modification de la liste
// Insère l'élément nouveau à la position du curseur
void    add(E nouveau);

// Remplace l'élément renvoyé par le dernier next() ou previous() par l'élément nouveau
void    set(E nouveau);

// Supprime de la liste l'élément renvoyé par le dernier next() ou previous().
 void   remove();
```

UML light