Travaux pratiques
séance n°6

Classes, fichiers, API

Exercice 1

On veut écrire une programme capable de sauvegarder un texte saisi au clavier sur un fichier texte puis d'afficher ce même fichier sur le terminal et enfin d'afficher l'histogramme des fréquences d'apparition de chaque mot du texte.

Question 1

Définir la classe `Fichier` comportant une variable d'instance mémorisant le nom du fichier sur disque., un constructeur surchargé et 3 méthodes :

- le constructeur permet d'associer le nom quelconque d'un fichier texte à l'instance de la classe `Fichier`
- la méthode `clavier2Disque` enregistre un texte saisi au clavier sous la forme d'un fichier texte du nom associé à l'instance. Pour terminer la saisie, il faut insérer un caractère de fin de fichier. En général le caractère '^d' fait l'affaire.
- la méthode `Disque2Term` affiche le contenu du fichier sur le terminal.
- la méthode `fichier2Mots` qui transforme le contenu du fichier en une instance de la classe Mots (voir question 3).

Le code de ces 3 méthodes est à compléter :

```
public void clavier2Disque(){
       // construire un flot en entrée pour la lecture sur clavier
        ……
        String s = null;
        try{
          // construire un flot en sortie pour écrire sur le fichier
           ……
           do{
              // lire une chaîne de caractères à partir du clavier
              s = ……;
              // écrire cette chaîne de caractères sur le fichier
              ……
           // tant qu'il y a quelque chose à lire
           }while( …… );
        }catch( FileNotFoundException r ){
            System.out.println("le fichier "+nomFichier+"est inconnu");
        }
        finally{
         // fermeture du flot de sortie
         ……
        }
    }
```

```
public void Disque2Term(){
        String ligne = null;
        BufferedReader input = null;
        try{
         // construire un flot en lecture connecté au fichier disque
          input =……
          while( (ligne = input.readLine()) != null ){
            System.out.print( ligne+" " );
          }
        }catch( FileNotFoundException r ){
            System.out.println("le fichier "+nomFichier+"est inconnu");
        }catch( IOException r ){
            System.out.println("erreur d'E/S");
        }finally{
            try{
                input.close();
            }catch( IOException r ){
                System.out.println("erreur d'E/S");
            }
        }
    }

// à compléter après les question 3 et 4
public Mots fichier2Mots(){
        String ligne = null;
        BufferedReader input = null;
        Mots mots = new Mots();
        try{
          // construire un flot en lecture connecté au fichier disque
          input =……
          while( (ligne = input.readLine()) != null ){
            ajouter une instance de la classe Mot à la variable mots
            ……
          }
        }catch( FileNotFoundException r ){
            System.out.println("le fichier "+nomFichier+"est inconnu");
        }catch( IOException r ){
            System.out.println("erreur d'E/S");
        }finally{
            try{
                input.close();
            }catch( IOException r ){
                System.out.println("erreur d'E/S");
            }
        }
        return mots;
    }
```

Rappel :

Pour créer un objet fichier texte dont le nom est `"fichier"`:

```
File file = new File( "fichier" );
```

Avant de lire un fichier texte, il faut lui connecter un flot ("stream") :

```
BufferedReader  input = new BufferedReader( new FileReader(fichier) );
```

Pour lire une chaîne `s`:

```
s = input.readLine();
```

La méthode `readLine` retourne `null` si la fin de fichier est atteinte et déclenche une `IOException` en cas d'erreur d'E/S.

Avant de terminer, il faut fermer le fichier :

```
input.close();
```

Avant d'écrire sur un fichier texte, il faut lui connecter un flot ("stream") :

```
PrintWriter pw = new PrintWriter("fichier");
```

L'exception `FileNotFoundException` est levée si le fichier n'est pas trouvé
Pour écrire une chaîne s:

```
pw.println(s);
```

Avant de terminer, il faut fermer le fichier :

```
pw.close();
```

Question 2

Ecrire un programme de test qui lit un texte au clavier, l'enregistre sur disque et affiche son contenu à l'écran.

Question 3

Définir la classe `Mot` caractérisée par sa chaîne de caractères (variable d'instance `mot`) et sa fréquence d'apparition dans un texte (variable d'instance `frequence`). Ajouter les accesseurs (les ″ getters″ et ″ setters″) à chaque variable d'instance, une méthode pour incrémenter la fréquence du mot et une méthode d'affichage de sa fréquence sous la forme d'une suite de '*'.

```
public void printFrequence(){
    System.out.print( …… );
    for( ……;……;…… );
        System.out.print(……);
    System.out.println();
}
```

Question 4

Définir la classe `Mots` qui rassemble tous les mots d'un texte.
La classe `Mots` est constituée :

- de la variable d'instance (`tabMots`) de type `java.util.HashMap` pour rassembler des instances de `Mot`. La classe `HashMap` représente des tables de hachage. Chaque valeur (instance d'une classe, ici un mot) est repérée par une clé (ici la chaîne de caractères qui correspond au mot).
Conseil : pour créer une instance de `HashMap`, consulter la documentation en ligne.
Chaque mot pourra être ajouté à une table (instance de la classe `HashMap`) grâce à sa méthode V **put**(K key, V value).

```
private HashMap<…,…> tabMots;
```

- d'un constructeur (à compléter) qui crée une instance de la variable d'instance précédente (la table de hachage)

```
public Mots(){
    tabMots = ……;
}
```

- de la méthode à compléter `ajouter` qui ajoute un mot à la table de hachage à condition que la chaîne de caractères qui lui correspond ne soit pas déjà présente et incrémente, si besoin est sa fréquence.

```
// tabMots étant l'identificateur de la table de hachage
public void ajouter( Mot mot ){
    Mot courant = null;
    // test : la table de hachage est-elle vide ?
    if( …… ){
        // ajouter un élément à la table de hachage
        ……
        // incrémenter sa fréquence
```

```
                .....
            }else{
                if( !tabMots.containsKey(mot.getMot()) ){
                    // ajouter un élément à la table de hachage
                     .....
                }
                // à partir de la table de hachage, retourner le mot
                // connaissant sa chaîne de caractères
                 courant = tabMots.get(mot.getMot());
                // incrémenter sa fréquence
                .....
             }
        }
```

- de la méthode `printHistogramme` à compléter qui affiche l'histogramme des fréquences

```
    // affichage de l'histogramme
    public void printHistogramme(){
      System.out.println("\n\n------Histogramme des fréquences--");
      Collection<Mot> collec = tabMots.values();
      Iterator<Mot> iter = collec.iterator();
      Mot courant = null;
      while( iter.hasNext() ){
          courant = iter.next();
          // affichage de la fréquence du mot courant
          .....
      }
    }
```

On utilise le "design pattern" `Iterator` (voir ci-dessous)

L'interface `Iterator` possède les méthodes :

`boolean` **`hasNext`**`()`

Tests if this iteration contains more elements.

`E` **`next`**`()`

Returns the next element of this iteration if this iteration object has at least one more element to provide.

Méthodes d'`HashMap` :

V **put**(K key, V value)

Associates the specified value with the specified key in this map.

V **get**(Object key)

Returns the value to which the specified key is mapped in this identity hash map, or null if the map contains no mapping for this key.

boolean **isEmpty**()

Returns true if this map contains no key-value mappings.

boolean **containsKey**(Object key)

Returns true if this map contains a mapping for the specified key.

Question 5

Ecrire un programme de test dont l'algorithme est le suivant :

```
début
      créer une instance de fichier de nom "temp";
      afficher sur le terminal le message : "entrez votre texte";
      transférer sur le fichier le contenu saisi au clavier;
      afficher le contenu du fichier sur le terminal;
      afficher l'histogramme des fréquences de chacun des mots du fichier;
fin.
```