TP - RECENSEMENT POPULATION

Objectifs du TP

Maintenant que vous avez mis en œuvre les quelques lignes de code qui permettent de lire un fichier, nous allons passer à l'exploitation des données qu'il contient !
Le but de ce TP va être de créer une application simple, dotée du menu suivant :

1. Population d'une ville donnée
2. Population d'un département donné
3. Population d'une région donnée
4. Afficher les 10 régions les plus peuplées
5. Afficher les 10 départements les plus peuplés
6. Afficher les 10 villes les plus peuplées d'un département
7. Afficher les 10 villes les plus peuplées d'une région
8. Afficher les 10 villes les plus peuplées de France
9. Sortir

Description du TP Recensement Population

- Créez le package *fr.epsi.b3.recensement*
- Créez une classe exécutable Application qui sera notre point d'entrée
- Pour réaliser cette application, il existe de nombreuses possibilités avec leurs avantages et leurs inconvénients. Je vous propose plusieurs options et c'est à vous de choisir.
- **Modèle objet simple**
  Dans ce modèle nous avons une classe *Recensement* qui possède la liste de toutes les villes du fichier.
  La classe *Ville* aura les attributs suivants :
  - code région
  - nom de la région
  - code département
  - code de la commune
  - nom de la commune
  - population totale

  Dans ce cas l'objectif va être de construire une variable de type *Recensement* avec sa liste de **35800** instances de villes. Pour certains cas d'utilisation vous aurez sans doute besoin d'autres classes :
  - une classe *Region* pour la recherche des 10 régions les plus peuplées
  - une classe *Departement* pour la recherche des 10 départements les plus peuplés.

- **Consignes spécifiques**
  Afin d'éviter d'avoir des kilomètres de code dans la classe *Application*, le code de chaque option de menu sera développé dans une classe dite « de services ».
  Exemple dans le cas où l'utilisateur recherche la population d'une ville :

```
switch (choix){
	case 1:
		RecherchePopulationVille recherche = new RecherchePopulationVille();
		recherche.traiter(recensement,scanner);
		break;
```

- Dans le package *fr.epsi.b3.recensement* créez une classe *MenuService* abstraite dont hériteront toutes les classes de services.
- La classe *MenuService* **a une méthode abstraite** :

```
public abstract void traiter(Recensement recensement, Scanner scanner);
```

ANNEXES & SUPPORTS

Comptage grâce au *HashMap*

Pour les opérations de comptage, par exemple l'affichage des 10 régions les plus peuplées :

```
package entites;

import java.util.*;

class Article {

   /** nom de l'article */
   private String nom;
   /** référence de l'article */
   private String reference;
   /** catégorie */
   private String categorie;
   /** prix en euros */
   private double prix;

   /** Constructeur
    * @param nom nom de l'article
    * @param reference référence de l'article
    * @param categorie catégorie
    * @param prix prix en euros
    */
   public Article(String nom, String reference, String categorie, double
prix) {
      super();
      this.nom = nom;
      this.reference = reference;
      this.categorie = categorie;
      this.prix = prix;
   }
   /** Getter
    * @return the nom
    */
   public String getNom() {
      return nom;
   }
   /** Setter
    * @param nom the nom to set
    */
   public void setNom(String nom) {
      this.nom = nom;
   }
   /** Getter
    * @return the reference
    */
   public String getReference() {
      return reference;
   }
   /** Setter
    * @param reference the reference to set
    */
   public void setReference(String reference) {
```

```
        this.reference = reference;
    }
    /** Getter
     * @return the categorie
     */
    public String getCategorie() {
        return categorie;
    }
    /** Setter
     * @param categorie the categorie to set
     */
    public void setCategorie(String categorie) {
        this.categorie = categorie;
    }
    /** Getter
     * @return the prix
     */
    public double getPrix() {
        return prix;
    }
    /** Setter
     * @param prix the prix to set
     */
    public void setPrix(double prix) {
        this.prix = prix;
    }
}




/**
 * Application de comptage utilisant les HashMap
 */
public class AppComptage {

    /**
     * Point d'entrée
     *
     * @param args non utilisés ici
     */
    public static void main( String[] args ) {
        List<Article> articles = new ArrayList<>();
        articles.add( new Article( "IPhone 11", "REF001", "Smartphone", 859.0
) );
        articles.add( new Article( "Dragon Quest XI: Les combattants de la
destinée", "REF013", "Jeu vidéo", 44.49 ) );
        articles.add( new Article( "IPhone 11 Pro", "REF002", "Smartphone",
1159.0 ) );
        articles.add( new Article( "Izuku support téléphone voiture",
"REF003", "Accessoires Smartphone", 11.99 ) );
        articles.add( new Article( "Fire TV Stick", "REF004", "Accessoires
High Tech", 11.99 ) );
        articles.add( new Article( "Echo dot enceinte connectée", "REF005",
"Accessoires High Tech", 59.99 ) );
        articles.add( new Article( "FIFA 20", "REF006", "Jeu vidéo", 54.99 )
);
        articles.add( new Article( "Joyguard coque IPhone 11", "REF007",
"Accessoires Smartphone", 5.99 ) );
        articles.add( new Article( "Samsung Galaxy A10 Dual sim", "REF008",
```

```
"Smartphone", 121.0 ) );
      articles.add( new Article( "HETP Ecouteur Bluetooth sans fil",
"REF009", "Accessoires High Tech", 121.0 ) );
      articles.add( new Article( "XIAOMI Redmi Note 7", "REF010",
"Smartphone", 169.94 ) );
      articles.add( new Article( "Rampow adaptateur USB C vers OTG 3.1",
"REF011", "Accessoires High Tech", 6.99 ) );
      articles.add( new Article( "The legend of Zelda: Link'a Awakening",
"REF012", "Jeu vidéo", 41.99 ) );


      // Création du stockage des compteurs:
      // - clé de type String qui va correspondre à la catégorie, exemple:
Jeu vidéo.
      // - valeur de type Integer qui va correspondre au nombre d'articles
dans cette catégorie
      HashMap<String, Integer> compteurs = new HashMap<>();

      // Parcours de la liste des articles
      for ( int i = 0; i < articles.size(); i++ ) {
         Article article = articles.get( i );
         String categorie = article.getCategorie();

         // On recherche le compteur correspondant à la catégorie de
l'article
         Integer compteur = compteurs.get( categorie );

         // Si le compteur n'existe pas on le créé
         if ( compteur == null ) {
            compteur = 0;
         }

         // On l'incrémente de 1
         compteur++;

         // On le remet dans la map avec la nouvelle valeur
         compteurs.put( categorie, compteur );
      }

      // On affiche maintenant les résultats
      Set<String> categories = compteurs.keySet();
      Iterator<String> iterateur = categories.iterator();
      while ( iterateur.hasNext() ) {
         String categorie = iterateur.next();
         Integer valeurCompteur = compteurs.get( categorie );

         System.out.println( "Categorie:" + categorie + " - Nombre
d'articles:" + valeurCompteur );
      }
   }
}
```

Parsing du fichier :

Traiter une ligne de données du recensement :

```
// On commence par découper la ligne en morceaux sur la base du caractère
séparateur « ; » . De plus on ne récupère que les morceaux qui nous
intéressent. En l'occurrence on ignore les morceaux 3 et 4 dont on a pas
besoin dans le TP
String[] morceaux = ligne.split(";");
String codeRegion = morceaux[0];
String nomRegion = morceaux[1];
String codeDepartement = morceaux[2];
String codeCommune = morceaux[5];
String nomCommune = morceaux[6];
String population = morceaux[7];
// Pour la population, avant la conversion en int, il faut d'abord
supprimer les
// espaces qui se trouvent à l'intérieur.
int populationTotale = Integer.parseInt(population.replace(" ",
"").trim());
// On cree maintenant la ville avec toutes ses données utiles
Ville ville = new Ville(codeRegion, nomRegion, codeDepartement,
codeCommune, nomCommune, populationTotale);
```

La classe de recensement qui possède toutes les villes :

```
public class Recensement {
   private List<Ville> villes = new ArrayList<>();
   public List<Ville> getVilles() {
      return villes;
   }
   public void setVilles(List<Ville> villes) {
      this.villes = villes;
   }
}
```

Algorithme :

1. En début de programme, on crée une instance de *Recensement*
2. On lit le fichier et on récupère l'ensemble des lignes.
3. On boucle sur l'ensemble des lignes du fichier :
   a. pour chaque ligne, on crée une nouvelle instance de ville (cf. code ci-dessus)
   b. on ajoute cette ville à la liste des villes du recensement :

```
recensement.getVilles().add(ville);
```

A chaque fois que l'utilisateur va sélectionner une option de menu, on va appeler la méthode *traiter* d'une classe de services à laquelle on passera en paramètre notre instance de *Recensement*.

Le travail de calcul et d'extraction de données se fera sur cette instance de *Recensement* et toutes ses villes.