Université de Montpellier / Faculté des sciences / MPO2 / HAI401I

Itérateurs et streams

Dans ces travaux, nous explorons comparativement les notions d'itérateurs et de streams pour certaines opérations de manipulation d'un objet composite. L'objet composite est ici une *direction de lycée*, qui se compose d'*employés de lycée*.

Les classes correspondantes sont esquissées ci-dessous. Une énumération `Categorie` décrit les différentes catégories des employés, correspondant à un niveau d'études et de rémunération. Un employé (`EmployeLycee`) est décrit par ses nom, prénom, âge, année de recrutement, catégorie, corps, grade et échelon.

```
public enum Categorie {A,B,C}

public class EmployeLycee {
   private String nom = "";
   private String prenom = "";
   private int age;
   private int anneeRecrutement;
   private Categorie categorie;
   private String corps, grade;
   private int echelon;

   public EmployeLycee() {}
   public EmployeLycee(String nom, String prenom, int age, int anneeRecrutement,
            Categorie categorie, String corps,
            String grade, int echelon) {
      this.nom = nom;
      this.prenom = prenom;
      this.age = age;
      this.anneeRecrutement = anneeRecrutement;
      this.categorie = categorie;
      this.corps = corps;
      this.grade = grade;
      this.echelon = echelon;
   }
   public String getNom() {return nom;}
   public void setNom(String nom) {this.nom = nom;}
   public String getPrenom() {return prenom;}
   public void setPrenom(String prenom) {this.prenom = prenom;}
   public int getAge() {return age;}
   public void setAge(int age) {this.age = age;}
   public int getAnneeRecrutement() {return anneeRecrutement;}
   public void setAnneeRecrutement(int anneeRecrutement)
         {this.anneeRecrutement = anneeRecrutement;}
   public Categorie getCategorie() {return categorie;}
   public void setCategorie(Categorie categorie) {this.categorie = categorie;}
   public String getCorps() {return corps;}
   public void setCorps(String corps) {this.corps = corps;}
   public String getGrade() {return grade;}
```

```
   public void setGrade(String grade) {this.grade = grade;}
   public int getEchelon() {return echelon;}
   public void setEchelon(int echelon) {this.echelon = echelon;}
   public String toString() {
         return "\n"+prenom+" "+nom+" "+anneeRecrutement+" "+categorie
                  +" "+ corps +" "+grade+ " "+echelon;
   }
}
```

Une direction de lycée se compose d'un proviseur et de son adjoint, d'un conseiller principal d'éducation, d'un gestionnaire et d'un chef de travaux.

```
public class DirectionLycee{
   private EmployeLycee proviseur, proviseurAdjoint, conseillerPrincipaldEducation,
                              gestionnaire, chefDeTravaux;
   public DirectionLycee() {}
   public EmployeLycee getProviseur() {return proviseur;}
   public void setProviseur(EmployeLycee proviseur) {this.proviseur = proviseur;}
   public EmployeLycee getProviseurAdjoint() {return proviseurAdjoint;}
   public void setProviseurAdjoint(EmployeLycee proviseurAdjoint)
      {this.proviseurAdjoint = proviseurAdjoint;}
   public EmployeLycee getConseillerPrincipaldEducation()
      {return conseillerPrincipaldEducation;}
   public void setConseillerPrincipaldEducation(EmployeLycee conseillerPrincipaldEducation)
      {this.conseillerPrincipaldEducation = conseillerPrincipaldEducation;}
   public EmployeLycee getGestionnaire() {return gestionnaire;}
   public void setGestionnaire(EmployeLycee gestionnaire) {this.gestionnaire = gestionnaire;}
   public EmployeLycee getChefDeTravaux() {return chefDeTravaux;}
   public void setChefDeTravaux(EmployeLycee chefDeTravaux)
      {this.chefDeTravaux = chefDeTravaux;}
   public String toString() {
      String res ="Direction du lycee "+"\nproviseur "+this.getProviseur()
                  +"\nproviseur adjoint "+this.getProviseurAdjoint()
                  +"\ngestionnaire "+this.getGestionnaire()
                  +"\nCPE "+this.getConseillerPrincipaldEducation()
                  +"\nChef de travaux "+this.getChefDeTravaux();
      return res;
   }
   public int ageMoyen() {
         return (this.getProviseur().getAge()+this.getProviseurAdjoint().getAge()+
                  this.getConseillerPrincipaldEducation().getAge()+
                  this.getGestionnaire().getAge()+
                  this.getChefDeTravaux().getAge())/5;
   }
}
```

QUESTION 1 *Manipulation sans itérateur ni stream.*
Sur le modèle de la méthode `ageMoyen` *qui calcule l'âge moyen en considérant un par un les âges de tous les employés, écrivez pour la classe* `DirectionLycee` *les méthodes suivantes :*

- `void afficheNoms()` *qui affiche les noms des employés sur la sortie standard* `System.out`.
- `ArrayList<EmployeLycee> recruteApres(int annee)` *qui retourne la liste des employés recrutés après une certaine année passée en paramètre.*
- `double ageMoyenCategorieAvant(Categorie cat, int annee)` *qui retournant l'âge moyen des membres d'une certaine catégorie recrutés avant une certaine année.*

— *`int anneeRecrutementPlusAnciennePourCorps(String corps)` qui retourne l'année de recrutement la plus ancienne pour les employés d'un certain corps passé en paramètre.*

Que se passe-t-il si on ajoutait un nouveau employé dans la direction (par exemple un correspondant hygiène et sécurité), quel est l'impact sur ces méthodes ?

QUESTION 2 *Manipulation avec un itérateur.*
Munissez la classe `DirectionLycee` d'un itérateur puis, en utilisant cet itérateur, écrivez les méthodes suivantes :

— *`int ageMoyenIte()` qui retourne l'âge moyen des employés de la direction.*
— *`void afficheNomsIte()` qui affiche les noms des employés sur la sortie standard `System.out`.*
— *`ArrayList<EmployeLycee> recruteApresIte(int annee)` qui retourne la liste des employés recrutés après une certaine année passée en paramètre.*
— *`double ageMoyenCategorieAvantIte(Categorie cat, int annee)` qui retourne l'âge moyen des membres d'une certaine catégorie recrutés avant une certaine année.*
— *`int anneeRecrutementPlusAnciennePourCorpsIte(String corps)` qui retourne l'année de recrutement la plus ancienne pour les employés d'un certain corps passé en paramètre.*

QUESTION 3 *Manipulation avec un stream.*
Commencez par écrire deux méthodes pour disposer d'un stream :

— *Une méthode `ArrayList<EmployeLycee> membres()` qui met simplement les membres de la direction dans une liste et la retourne. On peut ensuite appliquer la méthode `stream()` à la liste retournée pour disposer du stream.*
— *Une méthode `Stream<EmployeLycee> streamMembres()` qui appellera la méthode suivante et lui passera un itérateur sur la direction en paramètre pour retourner un stream :*

```
public static <T> Stream<T> iteratorVersStreamSequentiel(Iterator<T> ite)
{
   // tout d'abord on convertit ite en spliterator qui sert d'intermédiaire
   Spliterator<T> spitr
         = Spliterators.spliteratorUnknownSize(ite, Spliterator.NONNULL);
   // puis on convertit le spliterator en stream sequentiel
   // "false" indique que le stream ne sera pas parallele
   return StreamSupport.stream(spitr, false);
}
```

Puis écrivez les méthodes suivantes :

— *`int ageMoyenStream()` qui retourne l'âge moyen des employés de la direction.*
— *`void afficheNomsStream()` qui affiche les noms des employés sur la sortie standard `System.out`.*
— *`ArrayList<EmployeLycee> recruteApresStream(int annee)` qui retourne la liste des employés recrutés après une certaine année passée en paramètre.*
— *`double ageMoyenCategorieAvantStream(Categorie cat, int annee)` qui retourne l'âge moyen des membres d'une certaine catégorie recrutés avant une certaine année.*
— *`int anneeRecrutementPlusAnciennePourCorpsStream(String corps)` qui retourne l'année de recrutement la plus ancienne pour les employés d'un certain corps passé en paramètre.*