Rappels Java Exercices (2)

Ynov - Lyon – 2021/2022

Exercices - Instructions

- Créer un projet Java simple : **j2ee-td2**
- Pour chaque exercice :
  - Créer un package contenant la ou les classes associées à l'exercice : **com.j2ee.td2.ex1**, **com.j2ee.td2.ex2, …**
  - Dans la classe principale, créer une méthode « main() » permettant de vérifier l'exercice
  - Pour tester la classe, dans Eclipse, faire clic-droit → Run As… → Java Application
- Pour les exercices avec Scanner : on utilise également une méthode « main », mais les paramètres d'entrée seront à entrer dans la console (Console.in). Une fois la classe entrée, il faut copier les paramètres d'entrée dans la console et appuyer sur Entrée.

Collections - Exercice 1

- Créer une classe Exercice1 et une méthode qui permet de lister tous les caractères différents d'une chaîne de caractères, passée en argument :
  - Utiliser un Scanner pour passer les arguments
  - Créer la méthode :

  public static Collection<Character> extraireCaracteres(String string);

  - Réfléchir au type de *Collection* à utiliser pour éviter d'avoir des doublons

Pour tester : *extraireCaracteres("exercice")* doit retourner ['e', 'x', 'r', 'c', 'i']

Collections - Exercice 2

- Créer une classe Exercice2 sur le modèle de l'exercice precedent, mais la méthode doit cette fois-ci retourner les caractères tries par ordre alphabétique

  public static Collection<Character> extraireCaracteresTries(String string);

Pour tester : *extraireCaracteresTries("exercice")* doit retourner ['e', 'c', 'i', 'r', 'x']

Collections - Exercice 3

- Créer une classe Exercice3 sur le modèle de l'exercice precedent, mais la méthode doit cette fois-ci retourner également le nombre d'occurences de chaque lettres.

  public static Map<Character, Integer> extraireCaracteresOccurences(String string);

Pour tester : *extraireCaracteresOccurences("exercice")* doit retourner ['e' : 3, 'x' : 1, 'r' : 1, 'c' : 2, 'i' : 1]

Collections - Exercice 4

- Créer une classe Exercice4 sur le modèle des exercices précédents, avec une méthode permettant d'analyser une chaîne de caractère. Le résultat de cette méthode doit permettre de récupérer le caractère le plus fréquent, la liste des caractères triés par occurrence, la liste des caractères triés par ordre alphabétique, ainsi que la liste complète des caractères avec le nombre d'occurence.

  public static ResultatAnalyse analyserString(String string);

- Méthodes de la classe ResultatAnalyse :

  public Character recupererCaracterePlusUtilise()
  public Collection<Character> recupererCaracteresUtilisesTries()
  public Collection<Character> recupererCaracteresLesPlusUtilises()
  public Map<Character, Integer> recupererCaracteresEtOccurences()

POO – Exercice 5

- Boulangerie

Afficher le salaire moyen dans une boulangerie et le chiffre d'affaire total

Une boulangerie est constituée avec plusieurs employés qui sont soit des boulangers, soit des vendeurs. Les boulangers ont un salaire fixe de 1700€, les vendeurs de 1300€ + 10% du chiffre d'affaire qu'ils réalisent.
Créer les classes : *Boulangerie*, *Employe*, *Vendeur*, *Boulanger*

La classe Boulangerie doit avoir les méthodes :

- ajouterEmploye(Employe)
- calculerSalaireMoyen()
- calculerChiffreDaffaireTotal()

La classe Employe doit avoir la méthode :

- calculerSalaire()

```
public class ExerciceBoulangerie {

    public static void main(String[] args) {
        Boulangerie b = new Boulangerie();
        b.ajouterEmploye(new Vendeur("Pierre", 30000));
        b.ajouterEmploye(new Vendeur("Jean", 22000));
        b.ajouterEmploye(new Boulanger("Delphine"));
        b.ajouterEmploye(new Boulanger("Coraline"));
        b.ajouterEmploye(new Boulanger("Anthony"));

        System.out.println("Le salaire moyen dans la boulangerie est de "
                + b.calculerSalaireMoyen() + " euros.");
        System.out.println("Le chiffre d'affaire total de la boulangerie est de "
                + b.calculerChiffreDaffaireTotal() + " euros.");
    }

}
```

Attendu :

```
Le salaire moyen dans la boulangerie est de 2580.0 euros.
Le chiffre d'affaire total de la boulangerie est de 52000.0 euros.
```

Héritage - Exercice 6

- Créer le modèle de données suivant permettant de gérer une école (simplifiée) et les alertes liées à la COVID.
  - L'école est constituée d'utilisateurs : étudiants, et des membres du personnel. Parmi ces derniers, certains sont des professeurs, d'autres des conseillers d'orientation, et enfin certains sont des membres de l'équipe de direction.
  - Les utilisateurs ont tous un nom, un prénom, une date de naissance et une adresse email.
  - Les membres du personnel déclarent toutes leurs journées d'absence
  - Les cours rassemblent un professeur et des étudiants. Ils ont lieu à une date donnée, avec une heure de départ et une heure de fin. On peut savoir pour chaque cours quel utilisateur était présent, et quel utilisateur était absent, et si le cours était à distance ou non
  - Les élèves peuvent avoir des rendez-vous individuels avec les conseillers d'orientation.
  - Les liens à déterminer pour savoir qui est « à risque » :
    - Tous les élèves et professeurs ayant été en cours en présentiel avec un utilisateur positif lors des sept derniers jours sont à risque
    - Tous les membres de l'équipe de direction présents lors des sept derniers jours sont à risque si un membre de l'équipe est positif
    - Les utilisateurs ayant eu un rendez-vous individuel avec un autre utilisateur positif sont à risque

Héritage - Exercice 6 (suite)

- Utiliser la classe *Exercice5* fournie pour initialiser les données et faire les tests
- Classes du modèle de données :
  - Ecole
  - Utilisateur
  - Etudiant
  - MembreDuPersonnel
  - Professeur
  - ConseillerOrientation
  - MembreEquipeDirection
  - JourneeAbsence
  - Cours
  - RendezVousIndividuel

- Créer les constructeurs :

```
public Cours(List<Etudiant>
etudiantsInscrits, List<Etudiant>
etudiantsAbsents, Professeur professeur,
Date dateCours)
```

```
public Etudiant(String nom, String prenom,
Date dateDeNaissance, String email)
```

```
public
RendezVousIndividuel(ConseillerOrientation
conseiller, Etudiant etudiant, Date
dateRdv)
```

```
public JourneeAbsence(Date dateAbsence,
MembreDuPersonnel personnelAbsent)
```

Héritage - Exercice 6 (suite)

- La classe *Ecole* stocke tous les utilisateurs et rendez-vous (cours, journées d'absence, etc.)
  - Créer les attributs suivants :

```
List<Etudiant> etudiants;
List<MembreDuPersonnel> membresDuPersonnel;

List<Cours> cours;
List<RendezVousIndividuel> rendezVousIndividuels;
List<JourneeAbsence> journeesAbsence;
```

  - Créer la méthode :

```
public Collection<Utilisateur> extraireUtilisateursARisque(Utilisateur utilisateurPositif,
Date dateTestCovid)
```

- Penser à faire des méthodes intermédiaires (ex : récupération des utilisateurs présents à un cours, récupération des cours et des rdv ayant eu lieu entre deux dates, récupération des membres du personnels n'ayant pas eu d'absence entre deux dates, …)