Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2018 – 2019
DEV1

DEV1 – Laboratoires Java I

TD 13 – Mise en pratique
La gestion des clients (ébauche)

1 Introduction

Dans ce TD, on vous propose de mettre en pratique les notions vues auparavant pour créer un programme gérant l'enregistrement de nouveaux clients d'une entreprise (imaginaire). De nombreuses données différentes peuvent être associées à un client d'une entreprise donnée : son identifiant (numéro d'identification au sein de l'entreprise), son nom, son numéro de compte courant, son adresse postale, son adresse e-mail, sa date de naissance, sa nationalité, ... Pour simplifier les choses dans ce TD, nous considérerons uniquement les identifiants, noms, prénoms et l'adresse e-mail. Nous considérerons qu'un identifiant client est simplement une suite de 10 chiffres (dans l'intervalle `0-9`).

Cet encodage au clavier n'aurait pas beaucoup d'intérêt si les données ne sont pas enregistrées à la fin de l'exécution du programme. Dans un système professionnel de gestion de clients, celles-ci sont le plus souvent enregistrées dans une base de donnée. Ici, nous simplifierons les choses et enregistrerons simplement les données dans un fichier plat (un fichier texte).

2 Développement

Exercice 1 **Créer la structure Client**

Déclarer une nouvelle structure `Client` comportant les champs repris dans l'introduction : identifiant, nom, prénom, adresse-e-mail.

Vous prendrez soin de déclarer un constructeur approprié ainsi qu'une méthode `toString()` renvoyant une description du client (il suffit pour ce faire de concaténer les différents champs les uns à la suite des autres).

Exercice 2 **Initialiser un tableau de clients**

Écrire une classe `GestionClients` contenant une méthode `main` dans laquelle on demande à l'utilisateur le nombre de nouveaux clients qu'il souhaite encoder. Si ce nombre est n, le programme initialisera alors un tableau de n clients. Pour ce faire, une référence vers le tableau contenant les clients peut-être déclarée dans la classe :

Version du document
26 novembre 2018

```
public class GestionClients
{
   // Déclare une référence destinée à contenir un tableau de structures Client.
   static Client [] clients;

   public static void main (String args[])
   {
      //Assigner un nouveau tableau de taille n à la variable clients ici.
      // n est choisi par l'utilisateur.
   }
}
```

Exercice 3 **Encoder les différents clients dans le tableau**

Après avoir créer un nouveau tableau de n clients, if faut initialiser chaque case du tableau avec un nouveau client dont les données sont entrées au clavier par l'utilisateur. Pour ce faire, le programme demandera n fois à l'utilisateur d'entrer un nom, un prénom, un identifiant et une adresse e-mail.

Exercice 4 **Afficher les clients encodés**

Ècrire une méthode

```
static void afficher()
```

qui affiche à l'écran, pour chaque client du tableau `clients[]` les donnés du clients. Cette méthode fera appel a la méthode `toString()` de la classe `Client`.

Exercice 5 **Vérifier le format de l'adresse e-mail**

L'utilisateur peut se tromper en entrant les données du client. En particulier, une adresse e-mail doit être au format

`s1@s2.s3`

ou s1,s2,s3 sont des chaînes de caractère. Ècrire une méthode

```
boolean verifierEmail(String email)
```

qui renvoie `true` si email a le format d'une adresse e-mail et `false` sinon. Mettre la fonction `main` à jour : si l'utilisateur entre une mauvaise adresse pour un certain client, le programme refuse de créer le nouveau client et lui demande de corriger l'adresse entrée.

Exercice 6 **Vérifier l'absence de doublons**

Bien que ce soit rare, il est possible que deux clients aient le même nom et le même prénom. Par contre, ils ne peuvent avoir le même identifiant au sein de la société. Ecrire une méthode

```
boolean verifierID(String identifiant)
```

qui renvoie `true` si l'identifiant que l'utilisateur vient de rentrer n'a pas déja été utilisé et `false` sinon. Adapter la méthode `main` de sorte que l'utilisateur ne puisse jamais encoder deux clients avec le même identifiant.

Exercice 7 Écrire les données dans un fichier

Il existe plusieurs façons d'écrire des données dans un fichier en Java. Celle que nous vous proposons utilise la classe `FileWriter` :

```
import java.io.FileWriter;

public class Td13
{
   static void Ecrire () throws Exception
   {
      // On va écrire dans le fichier Truc.txt
      // Si le fichier n'existe pas il est créé, sinon il est écrasé
      FileWriter w = new FileWriter ("Truc.txt");
      w.write("Turlututu chapeau pointu !");

      // Une fois qu'on a fini d'écrire, on ferme le fichier.
      w.close();

   }
}
```

Vous aurez remarqué la ligne

```
throws Exception
```

Celle-ci signifie simplement que notre méthode Ecrire peut générer des exceptions. En effet, si, pour une raison ou pour une autre, le fichier "Truc.txt" ne peut être ouvert ou qu'on ne peut écrire dedans (nous n'avons pas les droits d'accès) alors `FileWriter` renvoie un exception à la méthode Ecrire qui doit se charger de la gérer. Le mécanisme de gestion des exceptions en Java sera vu dans un cours ultérieur. Ici, on se contente de dire à Java qu'on ne gère pas l'exception explicitement.

Ecrire une méthode :

```
static void WriteData (String filename) throws Exception
```

qui écrira toute les données du tableau `clients[]` dans le fichier dont le nom est spécifié dans `filename`. Mettre à jour la méthode `main` pour demander un nom de fichier a l'utilisateur et valider l'écriture du fichier de données.