Yann.Secq@univ-lille.fr
Fabien.Delecroix@univ-lille.fr

TD 11

Objectifs: Comprendre la définition de types en ijava.

Implémenter un répertoire téléphonique

Revenons à une époque pas si lointaine où (certains) étudiant·e·s possédaient un téléphone mobile. C'était un appareil qui permettait de téléphoner, d'envoyer des SMS en utilisant un clavier à 12 boutons, et c'est à peu près tout. Ces appareils si convoités à l'époque possédaient aussi un répertoire, c'est à dire une liste de contacts. La mémoire de ces machines étant limitée, on ne pouvait avoir qu'un nombre limité d'ami·e·s. (Pour être exact, on pouvait avoir autant d'ami·e·s qu'on voulait, mais seul un nombre limité avait leur place dans le répertoire.)

Supposons donc que

```
final int TAILLE_REP = 100;
```

est la constante qui définit le nombre de contacts que le répertoire peut contenir.

Nous allons implémenter quelques fonctionnalités d'un tel répertoire, et en particulier une fonctionnalité qui aurait été très avant-gardiste à cette époque, à savoir recevoir un rappel pour les anniversaires de ses amis.

Échauffement: le type Date

Rappelons le type date qui a été vu en cours, ainsi que les quelques fonctionnalités qui lui sont associées.

```
// Définition du type Date, dans un fichier Date.java
class Date {
  int jour;
  int mois;
  int annee;
}

// Fonctionnalités associées au type Date, dans le fichier du programme principal
Date newDate(int jour, int mois, int annee) {
  Date d = new Date();
  d.jour = jour;
  d.mois = mois;
  d.annee = annee;
  return d;
}
String toString(Date d) {
  return d.jour+"/"+d.mois+"/"+d.annee;
}
```

Q1. Écrire la fonction `boolean equals(Date d1, Date d2)` qui retourne vrai si et seulement si`d1` et `d2` représentent la même date, càd ont les mêmes jour, mois et année.

Les contacts

Un *contact* est une entrée du répertoire.

Q2. Écrire le type `Contact` dont les champs sont un prénom, un nom, un numéro de téléphone, et une date de naissance.

Q3. Écrire la fonction `Contact newContact(...)` qui permet de construire un contact à partir des valeurs de ses champs; les ... doivent être remplacés par la définition des paramètres formels.

Q4. Écrire la fonction `String toString(Contact c)` qui donne la représentation d'un contact sous forme de chaine de caractères.

Le répertoire

Comme dit précédemment, un répertoire peut contenir au maximum `TAILLE_REP` contacts. Mais il peut tout à fait en contenir moins.

Ainsi, un répertoire sera défini par une structure qui contient un tableau de taille `TAILLE_REP`, et un entier représentant le nombre d'entrées actuellement stockées dans le tableau.

Q5. Définir le type `Repertoire` comme défini ci-dessus.

Q6. Écrire la fonction `Repertoire newRepertoireVide()` qui crée un répertoire pouvant contenir le nombre maximal d'entrées, mais qui est pour l'instant vide.

Q7. Écrire la fonction `void afficherContacts (Repertoire r)` qui affiche la liste de tous les contacts présents dans le répertoire. Utiliser la fonction `toString` pour les contacts définie plus haut.

Q8. Écrire la fonction `void ajouterContact(Repertoire r, Contact c)` qui ajoute un nouveau contact au répertoire. Si le répertoire a déjà atteint sa taille maximale, cette fonction n'ajoutera pas le contact et laissera le répertoire inchangé.

Q9. Écrire la fonction `void rappelerAnniversaires(Repertoire r, Date d)` qui affiche tous les contacts dont la date d'anniversaire est la même que la date donnée en paramètre. Utiliser les fonctions déjà écrites.

Q10. Écrire une fonction `void algorithm()` qui

1. créer un nouveau répertoire
2. y ajoute le contact Tim Stamp, né le 01/01/1970, dont le numéro de téléphone est 0606060606
3. affiche les contacts
4. affiche le rappel des anniversaires du jour

Prolongements

Groupes. Supposons maintenant que nous voulons un répertoire avec encore une fonctionnalité indispensable: pouvoir classer les contacts dans des groupes. Les groupes sont prédéfinis, et peuvent être *Amis*, *Famille*, *Collègues* et *Services* [1]

Proposer une manière d'enrichir les types définis pour permettre à ce que chaque contact appartienne à exactement un groupe.

Ajouter une fonction permettant d'afficher les contacts d'un groupe particulier.

Répertoire trié. Il est bien plus agréable de voir sa liste de contacts triée. Il n'est pas forcément nécessaire de pouvoir trier le répertoire; il suffit de s'assurer que l'ajout d'un nouveau contact se fait à la bonne place.

On suppose qu'on dispose de la fonction

```
int compareTo (Contact c1, Contact c2)
```

qui permet de comparer deux contacts d'après l'ordre lexicographique de leur prénom, puis de leur nom. `compareTo` retourne un nombre négatif si `c1` est avant `c2`, un nombre positif si `c1` est après `c2`, et retourne 0 si `c1` et `c2` ont le même prénom et le même nom.

Q11. Écrire la fonction `void insererNouveauContact(Repertoire r, Contact c)` qui, en supposant que `r` est trié dans l'ordre croissant, va insérer le contact `c` au bon endroit pour préserver le tri.

Q12. Écrire la fonction `compareTo` définie ci-dessus. Astuce: vous pouvez utiliser la fonction de ijava `int compareTo(Str s1, String s2)` qui permet de comparer deux chaines, et dont le résultat suit la même convention que celle décrite ci-dessus.

1. Comme par exemple le numéro de Transpole pour s'informer sur les pannes de métro, Transpole n'ayant pas de site web à cette époque (que de toute manière on aurait eu du mal à consulter).