I.U.T. Montpellier – Département informatique – 1ère année

Programmation 1 bis

Sujet 1 : classes Fraction et Date

1 Préambule

Dans un répertoire dédié au module (`~/Documents/java/` par exemple), créer de préférence un sous-répertoire pour chaque exercice. Pour l'exercice sur les fractions par exemple, on trouvera un fichier `Fraction.java` décrivant la classe `Fraction`, un fichier `MainFraction.java` permettant de tester la classe `Fraction` et le fichier `Ut.java` (qui peut fournir des fonctions utiles...).

2 Une classe Fraction

Le but de cet exercice est de concevoir une classe `Fraction`. Une instance $f = \frac{p}{q}$ de cette classe sera définie pour $p \in \mathbb{Z}$, $q \in \mathbb{Z}^*$.

Copier dans un nouveau sous-répertoire, les fichiers `Fraction.java` et `MainFraction.java` qui se trouvent sur l'ENT.

1. Etudier ces classes. Les compiler. Les exécuter.
2. Ajouter à la classe `Fraction` une méthode `reduire` permettant de mettre une instance de `Fraction` sous forme réduite. Tester dans `MainFraction`.
   Indication : cette méthode modifie l'objet `this`.
   On pourra utiliser la méthode `pgcd` de la classe `Ut` (récupérable sur l'ENT) qui implémente l'algorithme d'Euclide.
3. Ecrire comme variante une méthode : `public Fraction fractionReduite ()`
   qui retourne une nouvelle fraction correspondant à la fraction réduite de `this` (sans modifier `this`).
4. Etant donnée une seconde fraction f, ajouter à la classe `Fraction` des méthodes (invoquées par les instances de `Fraction`) qui retournent :
   – $this * f$
   – $this + f$
5. Ajouter une méthode `puissance` qui, étant donné un entier naturel n, calcule/retourne $this^n$.

Indication : pour les sous-questions 3, 4 et 5, on codera des méthodes qui ne modifient pas la fraction `this` (ni f), mais qui renvoient une nouvelle fraction.

3 Cahier des charges pour une classe Date

Concevoir une classe `Date` telle que, pour une instance d_1 de cette classe, on puisse :

1. incrémenter la date `this` d'un jour ;
2. connaître la date du lendemain (définir une fonction qui renvoie une nouvelle date) ;
3. afficher la date `this` sous les deux formes, selon l'exemple suivant :
   - 31 janvier 2013
   - 31/01/2013
4. Etant donnée une seconde date d_2, déterminer :
   - si `this` est égale à d_2 ;
   - si `this` est antérieure à d_2 ;
   - si `this` est postérieure à d_2 ;
   - le nombre de jours séparant `this` de d_2.

Indications

- Vous pourrez définir une fonction `nbJoursMois` sans paramètre qui retourne le nombre de jours du mois de `this` (31 pour janvier, 30 pour juin, 28 ou 29 pour février selon que l'année de `this` est bissextile ou non).
- Vous pourrez également définir une méthode :
```
private boolean anneeEstBissextile()
    // pré-requis: aucun
    // résultat:   retourne vrai ssi l'année de this est bissextile
```
- Vous pourrez utiliser le tableau suivant dans certaines de vos méthodes.
```
String[] moisLettres = {"janvier","fevrier","mars","avril", ...};
```

Remarque pédagogique

Pour éviter de déclarer `moisLettres` comme une variable locale dans chacune des méthodes qui l'utilise, il est préférable de la déclarer une seule fois comme variable de *classe*, en haut de la classe `Date`. En pratique, il faut ajouter le mot-clef `static`.

Ainsi, ce tableau de classe est alloué une seule fois dans la mémoire de la classe et n'est pas construit (copié) dans chaque instance de `Date`. Une variable de classe est connue de et partagée par toutes les instances (objets) de la classe.

On suggère aussi de rendre le tableau `moisLettres` constant, c'est-à-dire non modifiable, en ajoutant le mot-clé `final` dans la déclaration.

Ce qui donne en résumé :
```
private static final String[] moisLettres = {"janvier", ...};
```