Développement initiatique

Sujet 6 : Objets et classes : 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. Vous pourrez aussi copier à partir de l'ENT le fichier `Ut.java` qui fournit quelques 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, compiler et exécuter ces classes.
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. En utilisant la méthode de produit de fractions ci-dessus, écrire 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 la forme : 14 mars 2020.
4. Etant donnée une seconde date d_2, écrire des méthodes permettant de 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 utiliser la méthode :
`public static boolean Ut.estBissextile(int an)` ou 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, en haut de la classe `Date`, comme attribut de la classe, aussi appelé *variable de classe*. En pratique, il faut ajouter le mot-clef `static` :

```
private static String[] moisLettres = {"janvier", ...};
```

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.

Dates en Java pour les curieux

Plusieurs classes sont disponibles en Java pour gérer les dates. Nous vous conseillons d'utiliser la classe `LocalDate` du paquetage `java.time`, dont la description se trouve à l'URL :

`https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/time/LocalDate.html`

N'hésitez-pas à parcourir cette documentation par curiosité et avoir des idées d'extension de votre classe `Date`.