

Les fonctions ne sont pas à écrire entièrement, faites simplement un plan. Il est conseillé d'implémenter ces classes chez vous.

On considère l'interface suivante.

```
1 interface Evaluable {
    int toInt();
    // renvoie un entier qui evalue l'instance
    String toString();
    // renvoie une String qui donne la façon de s'afficher de l'objet
2 }
```

Exercice 1 Concevoir une classe `Date` qui implémente `Evaluable` ; elle s'affiche comme `Dimanche 27 septembre 2015` et s'évalue au nombre de jours écoulés depuis le 1er janvier de l'an 1. Elle doit être capable d'incrémenter d'un jour, de vérifier si elle est cohérente avec une autre date : par exemple la date précédente n'est pas cohérente avec `Jeudi 28 Septembre 2015`. (On ignore les années bissextiles.)

Exercice 2 Concevoir une classe `ListeTrie` sur le modèle de `PileExecutable` dont les chaînons doivent implémenter `Evaluable`. Elle doit être capable d'afficher ses éléments par deux méthodes différentes, l'une appelant `toString()`, l'autre `toInt()`. Quand on ajoute un élément, il doit être placé *au bon endroit* en fonction de son évaluation : les éléments de la liste doivent toujours être triés par `toInt()` croissant.

Comment doit-on modifier `Date` pour pouvoir l'utiliser avec cette chaîne ?

Peut-on modifier `ListeTrie` pour pouvoir utiliser `Date` telle quel ?

On considère maintenant le code suivant

```
1 interface Fonction {
    int appliquer(int x);
2 }

3
4
5
6 class FonctionMap extends ArrayList<Fonction> {
    public int appliquer (int x) {
        int y = x;
        for (Fonction f: this)
            y = f.appliquer(y);
        return y;
7     }
8 }

9
10
11
12
13
14 class Carre implements Fonction {
    int appliquer (int x) { return (x*x); }
15 }
16
17 class PlusTrois implements Fonction {
    int appliquer (int x) { return (x+3); }
18 }
19 }
```

Exercice 3 Que fait la classe `FonctionMap` ? Qu'affiche le code suivant ?

```
1 FonctionMap f = new FonctionMap();
2 f.add(new PlusTrois());
  System.out.println(f.appliquer(1));
4 f.add(new Carre());
  System.out.println(f.appliquer(1));
```

Comment ajouter facilement des `Fonction` à `f` ? Ajouter à `f` une fonction qui multiplie par 2 à l'aide d'une classe locale. Ajouter à `f` une fonction qui soustrait 4 à l'aide d'une classe anonyme. Qu'affiche alors `System.out.println(f.appliquer(1))` ;

Exercice 4 On considère le code suivant.

```
1 Class Poly extends FontionMap {
2   enum Atom implements Fonction {
      PlusOne, TimesX;

      static int xvalue = 1;
6     public int appliquer(int x) {
          if (this == PlusOne)
8             return (x+1);
          else
10             return x*xvalue;
        }
12    }
    public int appliquer(int i) {
14        Atom.xvalue=i;
        return super.appliquer(0);
16    }
}
```

Où peut-être placé le code suivant ? dans ce cas, qu'affiche-t-il ?

```
1 System.out.println(Atom.PlusOne.appliquer(1))
```

Qu'affiche le code suivant ?

```
1 FonctionMap g = new FonctionMap();
  g.add( Atom.PlusOne );
3  g.add( Atom.TimesX );
  g.add( Atom.PlusOne );
5  g.add( Atom.PlusOne );
  g.add( Atom.TimesX );
7  System.out.println(g.appliquer(0));
  System.out.println(g.appliquer(3));
```

Qu'est en réalité la fonction `g.appliquer(i)`. Quelle classe de fonctions peut-on réaliser en ajoutant uniquement `Atom.PlusOne` et `Atom.TimesX` ? Écrire le code pour la fonction $i \mapsto (i^2 + 1)^2$.

Exercice 5 L'élève avisé aura remarqué que `FonctionMap` satisfait elle-même l'interface `Fonction`; on considère dorénavant qu'elle a été déclarée comme suit.

```
class FonctionMap extends ArrayList<Fonction> implements Fonction
```

Le code suivant est-il correct et si oui, qu'affiche-t-il ?

```
1 f.add(g);
  System.out.println(f.appliquer());
3 f.add(f);
  System.out.println(f.appliquer());
```