Licence Informatique **UE J1IN4W01**
Université de Bordeaux **Programmation 2**
2015-2016 **Travaux Dirigés - feuille 7**

DS 2012/2013

Exercice 1 : Forme canonique

Ecrivez une classe `Date` qui représente une date de la forme jour/mois/année. Ces trois informations seront représentées par des entiers. On ne vous demande pas de tester leurs intervalles de valeurs.

En plus des méthodes de la forme canonique de classe, vous écrirez la méthode
`boolean before(Date d)` qui teste si la date à laquelle s'applique la méthode est antérieure à la date passée en paramètre.

Exercice 2 : Encapsulation

Soit le programme suivant :

```
public class Birthday {
    private Date date;
    private String name;

    public Birthday(Date date, String name) {
        this.date = date;
        this.name = name;
    }

    public String toString() {
        return "[ Anniversaire de " + name + " le " + date + "]";
    }

    public static void main(String arg[]) {
        Date d = new Date(20, 3, 2013);
        Birthday e = new Birthday(d, "Anatole");
        System.out.println(e);
        d.setMonth(6);
        System.out.println(e);
    }
}
```

Qu'affiche ce programme ? Respecte-t-il le principe d'encapsulation des données ? Si non, comment le corriger ?

Exercice 3 : Une autre classe

Ecrivez une classe `Meeting` qui contiendra les informations liées à une réunion : date, heure de début, heure de fin et description. Les heures seront simplement des entiers. La description est une chaine de caractères. Cette classe doit comporter toutes les méthodes de la forme canonique.

Exercice 4 : Polymorphisme

On veut écrire une classe `Agenda` permettant de stocker un nombre fixe (`MAX_SIZE`) d'évènements. Ces évènements pourront être du type `Birthday` ou `Meeting`. La classe Agenda doit permettre :

- d'ajouter un évènement à un agenda (on ne gèrera pas les éventuels conflits de date ou d'horaire),
- de supprimer tous les évènements ayant lieu avant une date passée en paramètre,
- d'afficher tous les événements ayant lieu à une date donnée.

Avant d'écrire la classe `Agenda`, réfléchissez à l'organisation de votre code. Comment implémenter la notion d'évènement tout en factorisant au mieux le code ? Cela induit-il des modifications des classes `Birthday` et `Meeting` ? Si oui, précisez les. Ecrivez un programme de test de la classe `Agenda`.