Programmation Orientée Objet
M2 CCI 2008–2009
hoenen@icps.u-strasbg.fr

Université de Strasbourg
Département d'informatique
20 février 2009

TP noté

Quelques précisions :
- l'ensemble des classes du TP doivent appartenir à un paquetage `cci.tpnote.nom` où vous remplacez `nom` par votre nom,
- réfléchissez à l'accessibilité à donner à vos attributs et méthodes,
- commentez le code pour décrire les méthodes, surtout celles qui ne sont pas demandées explicitement,
- ce n'est pas grave si toutes vos classes n'ont pas toutes les fonctionnalités, n'hésitez pas à passer à la suite si vous bloquez,
- le répertoire contenant vos fichiers (donc `nom`) doit être compressé par la commande `tar -cvzf nom.tar.gz nom`, lancée depuis le répertoire `tpnote`. Puis cette archive doit être attachée en pièce jointe d'un mail de sujet `[CCI]nom` et de destinataire hoenen@icps.u-strasbg.fr.

1 Un objet qui fait `Date`

On veut pouvoir représenter une date comme un objet qui possède des champs non-modifiables `jour` (type `int`), `mois` (type `Mois`) et `annee` (type `Annee`). Le code de la classe `Mois` se trouve à l'adresse http://icps.u-strasbg.fr/~hoenen/cours/2008-2009/poo/Mois.java. De même, la structure de la classe `Annee` se trouve à l'adresse http://icps.u-strasbg.fr/~hoenen/cours/2008-2009/poo/Annee.java.

1. Complétez la classe `Mois` et la classe `Annee`. La description des différentes méthodes est donnée en commentaires (sauf pour les méthodes classiques).
2. Écrire la classe `Date` avec 4 constructeurs différents :
   - `Date(int j, int m, int a)`
   - `Date(int j, Mois m, Annee a)`
   - `Date(String s)` : la chaîne de caractères contient une phrase du type "20 fevrier 2009", utilisez par exemple la méthode `split("\\p{javaWhitespace}")` de la classe `String` pour en séparer chaque mot.
   - `Date()` : utilisez une instance de la classe `java.util.Calendar` pour récupérer les informations (jour, mois et année) courantes.
3. Implantez les différents accesseurs (`get`), redéfinissez la méthode `toString` et implémentez l'interface `Comparable` pour pouvoir comparer deux dates entre elles.
4. Ajoutez une méthode renvoyant le nombre de jours écoulés entre deux dates.
5. Écrire une classe `DateTest` pour tester les 4 manières différentes de créer un objet `Date` et en vérifier l'affichage et la comparaison.

2 Un `Agenda`

On veut gérer un agenda simple, vu comme une liste d'évènements. Dans cet exercice les classes sont décrites brièvement pour vous laisser un maximum de liberté dans le choix de l'implantation. N'hésitez pas à rajouter des méthodes ou attributs si cela vous facilite le travail et que ça *colle* à l'esprit de la classe.

1. Construisez la classe simple `Horaire` pour représenter une heure à l'aide de champs entiers `heure` et `minute`.
2. Construisez la classe `Evenement`, qui hérite de la classe `Date` et possède un attribut de type `String` pour décrire l'évènement.

3. Construisez enfin la classe `Agenda` qui possède un attribut qui est une liste d'évènement (vous pouvez par exemple utiliser un `Vector` ou toute autre structure).
4. Implantez une méthode d'`Agenda` pour permettre l'ajout d'un évènement supplémentaire dans la liste, puis une méthode qui permet d'afficher le prochain évènement qui aura lieu (par rapport à la date courante).
5. Testez ces fonctionnalités dans un programme `AgendaTest`.

3 Reprise de la classe `Personne`

Reprendre le code de la classe `Personne` (vue dans le TP2) avec notamment des attributs `String nom`, `String prenom`, `int age` et `Adresse adresse`.

1. Modifiez cette classe pour ajouter un champ `Date dateDeNaissance` qui représente la date de naissance de la personne.
2. Écrire une classe `PersonneTest` qui possède une méthode permettant de trier un tableau d'objets de type `Personne` du plus vieux au plus jeune.
3. Rendre la classe `Personne` sérialisable (interface `java.io.Serializable` vue dans le TP5) et ajoutez une méthode à la classe `PersonneTest` pour sauvegarder sur fichiers des objets `Personne` présents dans un tableau (par exemple celui où les personnes sont triées de la plus vielle à la plus jeune).
4. Ajoutez une deuxième méthode pour importer des objets `Personne` qui ont été sérialisés.
5. Ajoutez une méthode `main` à `PersonneTest` pour en tester ses méthodes.