

Jean David Olekhnovitch
jd@ecom.fr

Java

Modélisation et implémentation d'un répondeur-enregistreur

Le but de ce TP est de simuler en Java le fonctionnement d'un répondeur-enregistreur téléphonique.

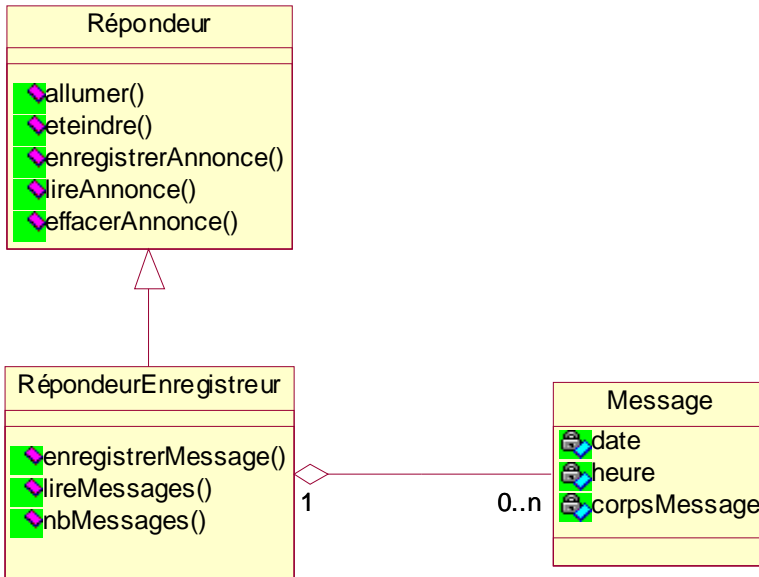
Ce dernier aura dans un premier temps les fonctions suivantes :

- possibilité d'enregistrer un message d'accueil
- fonction d'activation / désactivation du répondeur
- fonction d'appel du répondeur (ce qui se passe lorsqu'un utilisateur appelle sur votre ligne).

Dans une deuxième version (utilisant les notions d'héritage), le répondeur aura en outre les fonctions suivantes :

- enregistrer d'éventuels messages, qui seront stockés avec leur date d'émission
- fonction de consultation des messages reçus
- effacement des messages reçus.

Vous allez devoir passer par une étape de modélisation des classes, via UML. A titre indicatif, voici une proposition (pas forcément complète) :



Il n'est bien sûr pas demandé ici de gérer le stockage d'information sonores. On se contentera de simuler un message par une chaîne de caractères. L'interface d'utilisation sera limitée à quelques écrans textes.

Le stockage des messages va devoir passer non pas par un tableau, mais impérativement par une liste chaînée. Une telle liste est gérée en Java par la classe ArrayList.

Autres élément pouvant vous être utile :

- la classe *java.util.Date* pour le stockage de la date et de l'heure
- la méthode statique `System.currentTimeMillis()` permettant de récupérer la date et l'heure exacte en millisecondes depuis EPOCH
- Pour lire une donnée au clavier, essayez le code suivant :

```
BufferedReader d
    = new BufferedReader(new InputStreamReader(in));
String ligneLue=d.readLine();
```

- Il est demandé de veiller à une parfaite séparation entre les objets métiers et l'interface utilisateur. Dans le cadre de ce TP, l'interface se résumera a des appels de méthodes dans une méthode `main()`.
- Pour inclure une fonction d'affichage à une classe, surchargez la méthode `public String toString()` qui a pour avantage d'être appelée implicitement lorsque l'on cherche à afficher directement une instance.
- Vous allez devoir générer la documentation de vos classes via javadoc. Pour préparer cela, vous allez systématiser les commentaires devant chacune des méthodes de la manière suivante :

```
/** cette méthode est super */
public void uneMethode()
{ ... }
```