TP n°2
Gestion de version : CVS

Introduction

CVS (Concurrent Version System) est un système qui permet de gérer les différentes versions d'un ensemble de documents (code, site web, etc.). Une description sommaire peut être trouvée ici : http://www.ensta.fr/~diam/dev/online/cvs-doc-bouyer/cvs-html. L'essentiel des interactions avec cvs a lieu grâce à la commande `cvs` et ses différents arguments (`man cvs` pour plus de détails).

Mise en oeuvre de base

1. Initialisation de CVS

Les fichiers sont stockés dans un répertoire de dépôt (*repository* en anglais). Avant de lancer CVS pour un projet, il faut donc le paramétrer grâce à deux variables d'environnement, CVSROOT et EDITOR:

- CVSROOT doit désigner le répertoire de dépôt
- EDITOR précise l'éditeur que va appeler CVS

Il faut ensuite initialiser le répertoire de dépôt grâce à la commande `cvs init`.

2. Création d'un module

Avant de pouvoir travailler sur un projet, il faut définir un premier module dont vont faire partir des fichiers du projet. Si on part de zéro, le module se crée **depuis 'un répertoire vide** grâce à la commande :

```
cvs import nomModule tagVendeur tagRelease.
```

3. Ajouter des fichiers dans un module

On commence par récupérer le module par un : `cvs checkout nomModule`

Dans le répertoire correspondant au module, on crée les fichiers voulus, que l'on demande à cvs de gérer grâce à un :

```
cvs add fichier
```

On valide les changements grâce à un `cvs commit`.

Application

1. Créer une classe Etudiant (nom, note) avec une méthode d'affichage et un main d'exemple
2. Créer une classe Promotion (vecteur d'etudiant) permettant d'ajouter des étudiants. Définir une méthode d'affichage de la liste des étudiants, ainsi qu'un main d'exemple.
3. Appeler cette version "initiale" (man cvs)
4. Modifier la classe Etudiant en supprimant la methode affichage() et en ajoutant une methode toString(). Modifier le main et la classe Promotion en consequence.
5. Appeler cette version "base standard"
6. Récuperer la version "initiale"
7. Aspect Transversal des « Tags »
   Après avoir récupéré la version « Base standard », modifier le main de Promotion.java en supprimant l'ajout d'un étudiant ; faire un commit et appeler cette nouvelle version « baseAllegee ». Faire un cvs log. Regarder les noms symboliques des différentes versions des 2 fichiers.
8. Travail à plusieurs sans conflits
   Remonter de 2 répertoires et créer un nouveau répertoire de travail (Travail2) pour simuler un deuxième utilisateur. Aller dans ce répertoire, faire un cvs checkout, modifier le fichier Etudiant.java en rajoutant un commentaire en tête du fichier, puis faire un commit.
   Retourner alors dans le repertoire Travail1, rajouter un commentaire dans le main de Etudiant.java, puis faire un

commit, puis un update

9. Travail à plusieurs avec conflits
   Se placer dans le répertoire Trravail (utlisateur 1)
   Modifier la façon dont fonctionne le « toString » de la classe Etudiant (par exemple, si le séparateur entre le nom de l'étudiant et sa note était « : », le remplacer par une virgule.
   Faire un commit
   Aller dans le répertoire Travail2 (utilisateur 2)
   Modifier différemment la façon dont fonctionne le « toString » de la classe Etudiant (par exemple, si le séparateur entre le nom de l'étudiant et sa note était « : », le remplacer par une tiret.
   Faire un commit, puis un update