TP Gestion de version - Git - GitLab

05 Mai 2017 • adaptation du TP de Yoann Pigné

Le but de ce TP est de prendre en main l'outils de gestion de version GitLab de l'université.

Le travail est réalisé en équipes de deux personnes. Le but est de travailler en parallèle sur deux postes différents, le partage du code se faisant grâce à Git et à la plateforme GitLab.

Avant de commencer : configuration de GitLab

Le GitLab de l'université est hébergé ici : https://iutdoua-git.univ-lyon1.fr/

Lors de la première connexion, on utilise le système d'authentification de l'université.

Dans un terminal on configure le client Git local avec les commandes de configuration suivantes (remplacer les mots `LOGIN`, `PRENOM`, `NOM` et `EMAIL` par ce qui convient) :

```
git config --global credential.https://www-apps.univ-lehavre.fr.username LOGIN
git config --global user.name "PRENOM NOM"
git config --global user.email "EMAIL"
```

Pour ne pas retaper à chaque fois le mot de passe. Sous Linux taper :

```
git config --global credential.helper 'cache --timeout 3600'
```

GitLab est maintenant configuré. Plus besoin d'utiliser le "CAS de L'université" pour se connecter à l'avenir, on peut directement entrer le login (ou l'adresse email) et le mot de passe définit précédemment, sur la page de connexion.

Vous pouvez aussi passer par le logiciel installé sous windows « TortoiseGit » qui vous demandera votre login/password au moment du GitClone.

But et fonctionnement du TP.

Le but est les manipulation de GIT et le travail en équipe. Le travail s'effectue par équipe de deux personnes (deux postes séparés).

Se mettre d'accord avec un collègue pour travailler en binôme.

Dans la suite le binôme dont le nom est premier dans l'ordre lexicographique est appelé **Alice**. L'autre binôme est **Bob**.

La section "travail demandé" va décrire les tâches à remplir dans ce projet. Chaque tâche ou groupe de tâche est attribué soit à Alice, soit à Bob. Alice doit faire les tâches qui la concernent uniquement, de même pour Bob.

On essaye de travailler en parallèle quand cela est possible.

Le projet

On veut réaliser un simple modèle objet en Java qui permet de gérer un carnet d'adresses.

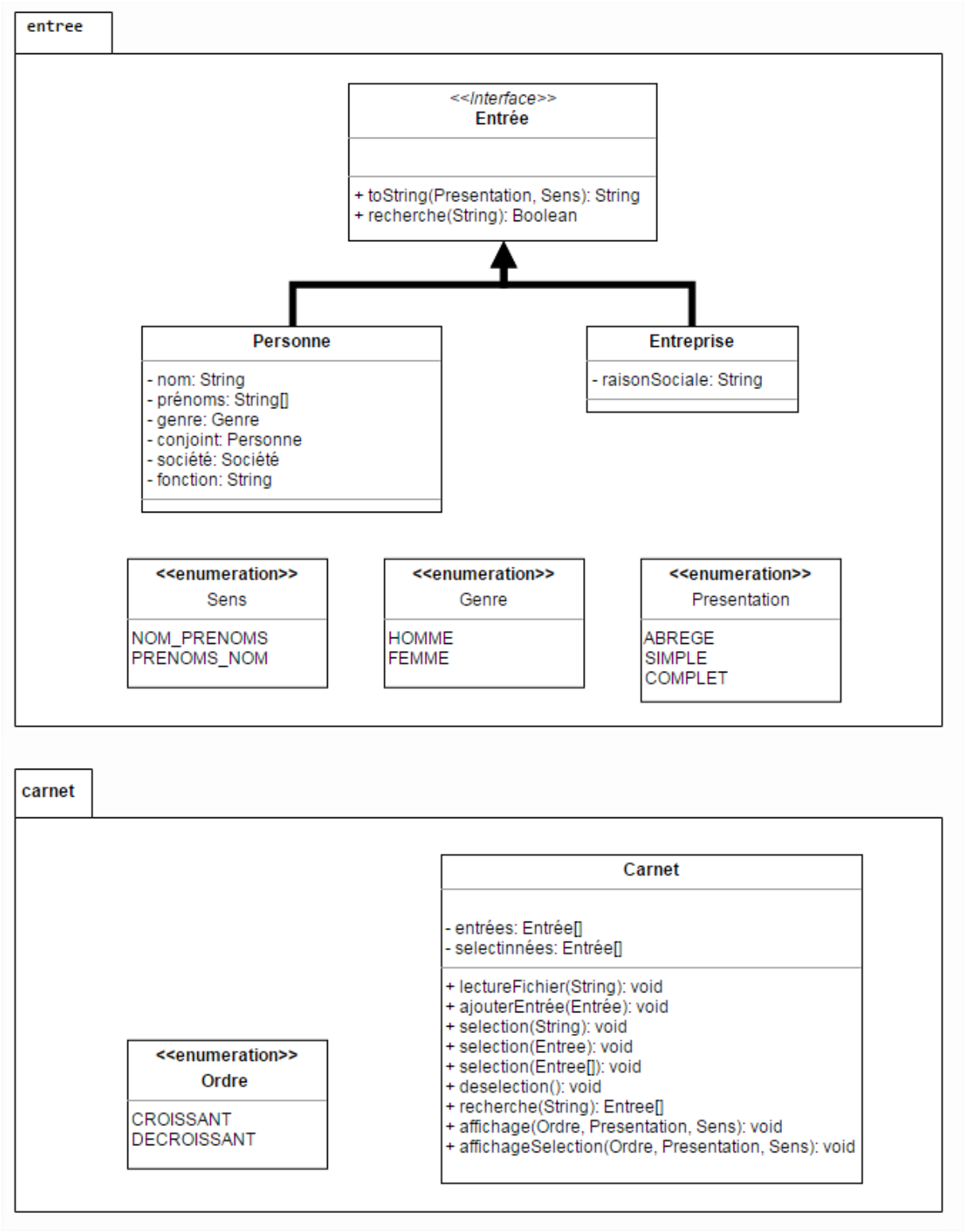
Un carnet d'adresses est constitué d'entrées qui peuvent représenter des personnes ou des sociétés. Les informations et la présentation sont différentes s'il s'agit d'une personne ou d'une société.

Une méthode utilitaire permet de lire les contacts à partir d'un fichier texte.

On peut faire des recherches sur le carnet et sélectionner des contacts pour les afficher.

L'affichage se fait en choisissant l'ordre lexicographique (croissant ou décroissant), le sens d'affichage (nom ou prénom) et le mode présentation (abrégé, simple ou complet).

En plus des méthodes indiquées, toutes les classes possèdent des accesseurs pour leurs champs privés. Le modèle objet est le suivant.


entree
<<Interface>>
Entrée
+ toString(Presentation, Sens): String
+ recherche(String): Boolean
Personne
- nom: String
- prénoms: String[]
- genre: Genre
- conjoint: Personne
- société: Société
- fonction: String
Entreprise
- raisonSociale: String
<<enumeration>>
Sens
NOM_PRENOMS
PRENOMS_NOM
<<enumeration>>
Genre
HOMME
FEMME
<<enumeration>>
Presentation
ABREGE
SIMPLE
COMPLET
carnet
Carnet
- entrées: Entrée[]
- selectinnées: Entrée[]
+ lectureFichier(String): void
+ ajouterEntrée(Entrée): void
+ selection(String): void
+ selection(Entree): void
+ selection(Entree[]): void
+ deselection(): void
+ recherche(String): Entree[]
+ affichage(Ordre, Presentation, Sens): void
+ affichageSelection(Ordre, Presentation, Sens): void
<<enumeration>>
Ordre
CROISSANT
DECROISSANT

Travail Demandé

1. Alice

- Créer le projet sur GitLab
- Dans un terminal, taper les commandes git pour créer un nouveau projet git et écrire et faire un premier commit/push.
- Ajouter Bob comme membre du projet (icon de roue crantée, puis *Members*) et lui donner les droits *Master*.
- Ajouter le prof (`stephane.pequignot`) au projet et lui donner les droits *Reporter*.

2. Bob

- Cloner de dépôt.
- Utiliser votre IDE pour créer un nouveau projet java en spécifiant le dépôt cloné comme *Location* (destination). Ne pas utiliser la destination par défaut.
- Créer un fichier `.gitignore` pour ignorer le dossier `bin`
- sélectionner, valider et envoyer tous les autres fichiers (`.gitignore`, `.project` , `.classpath`, `.settings`)

3. Alice

- Récupérer les modifications.
- Importer le projet dans Eclipse.
- Créer un fichier README contenant le “vrai” nom et l’adresse d’Alice et de Bob.
- Sélectionner, valider et envoyer.

4. Bob

- Créer le package `entree`
- Créer l’interface `Entree`

- Créer les énumérations `Presentaion`, `Sens` et `Genre`
- Sélectionner, valider et envoyer.

5. Alice

- Créer la classe `Personne`.
    - Ne **pas** écrire le corps de la méthode `recherche`.
    - Une personne peut avoir plusieurs prénoms. On les stocke sous forme de tableau.
    - La méthode `toString` prend en paramètre une énumération `Presentation` et `Sens`. Le sens définit si le nom doit être placé avant les prénoms. En fonction de la valeur de `Presentation`, la chaîne retournée sera différente.
        - La présentation *abrégée* affiche uniquement les initiales du prénom et le nom (e.g. `"H. J. Potter"`)
        - La présentation *simple* affiche le titre abrégé (M. ou Mme), le premier prénom suivit des initiales des autres prénoms, le nom, et entre parenthèses le nom de la société si elle existe. Exemple : `"M. Albus P. W. B. Dumbledore (Ecole de sorcellerie Poudlard)"`
        - La présentation *complète* affiche un maximum d'information sur plusieurs lignes si nécessaire. Par exemple :

```
o  M. Albus Perceval Wulfric Brian Dumbledore
o    - Société : Ecole de sorcellerie Poudlard
o    - Fonction: Directeur
```

- Sélectionner, valider et envoyer.

6. Bob

- Créer la classe `Société`.
    - Ne **pas** écrire le corps de la méthode `recherche`.
    - Les paramètres de la méthode `toString` sont ignorés. On affiche toujours de la même façon une société.
- Sélectionner, valider et envoyer

7. Alice

- Dans un package `test` écrire une classe de test (`TestPersonne`) qui contient un `main` qui teste toutes les méthodes de la classe `Personne`.
- Sélectionner, valider et envoyer.

8. Bob

- Dans un package `test` écrire une classe de test (`TestSociete`) qui contient un `main` qui testera toutes les méthodes de la classe `Société`.
- Sélectionner, valider et envoyer.

9. Alice ou Bob

- Dans le package `carnet` créer l'énumération `Ordre` et la classe Carnet **sans** ses méthodes
- Sélectionner, valider et envoyer.

10. Bob

- Dans une branche `lecture` créé la méthode `lectureFichier` qui permet de créer un carnet d'adresse avec ses entrées à partir d'un fichier passé en paramètre. Le fichier contient une entrée par ligne et respecte le format suivant :

```
ID;TYPE;CHAMP1;CHAMP2;CHAMP3
```

Si le TYPE est "PERSONNE" alors les autres champs seront :

```
ID;PERSONNE;PRENOMS;NOM;GENRE;ID_CONJOINT;ID_SOCIETE;FONCTION
```

Si le type est "SOCIETE" :

```
ID;SOCIETE;RAISON_SOCIALE
```

Les prénoms multiples sont séparés par des virgules. Les champs facultatifs peuvent être vides. Quelques exemples :

```
1;SOCIETE;Ecole de sorcellerie Poudlard
2;PERSONNE;Albus,Perceval,Wulfric,Brian;Dumbledore;H;;1;Directeur
3;PERSONNE;Harry,James;Potter;H;4;1;Elève
4;PERSONNE;Ginny;Weasley;F;3;1;Elève
```

- Des tests sont écrits avec un fichier d'exemple dans le package `test` avec la classe `TestLecture`. Dans le `main` de cette classe on test la lecture d'un fichier, on vérifie le nombre d'entités créées, on affiche les entités pour vérifiées qu'elles sont bien créées.

- Sélectionner, valider et envoyer la branche `lecture`.

- Dans l'application Web, faire un `Merge Request` de la branche `lecture` dans `master` et nommer Alice comme responsable de ce `Merge Request`.

11. Alice

- Dans une nouvelle branche `recherche_selection`, écrire la méthode `ajoutEntrée` de `Carnet`, créer toutes les méthodes de sélection et de recherche. Ecrire également le corps des méthodes recherche de `Personne` et `Société`. Pour la classe `Personne`, la méthode de `recherche` doit retourner vrai si la chaine de caractère est contenue dans l'un des prénoms ou dans le nom. Pour les société c'est seulement le champs raison sociale qui est recherché.

- Des tests sont écrits avec un fichiers d'exemple dans le package `test` avec la classe `TestSelection`. Dans le `main` de cette classe on crée (dans le code) un carnet d'adresse et on teste les différentes formes de sélection d'un fichier, on vérifie le nombre d'entités créées, on affiche les entités pour vérifiées qu'elles sont bien créées.

- Sélectionner, valider et envoyer la branche `recherche_selection`.

- Dans l'application Web, faire un *Merge Request* de la branche `lecture` dans `master` et nommer Bob comme responsable de ce *Merge Request*.

12. Bob

- Sur l'application Web GitLab, examiner le *Merge Request* de la branche `recherche_selection` envoyé par Alice. Faire des commentaires si nécessaire pour qu'Alice améliore/corrige le code. Quand tout semble bon, valider le *Merge Request*.

13. Alice

- Sur l'application Web GitLab, examiner le *Merge Request* de la branche `lecture` envoyé par Bob. Faire des commentaires si nécessaire pour que Bob améliore/corrige le code. Quand tout semble bon, valider le *Merge Request*.

14. Alice ou Bob

- Dans la branche `master`, écrire le code des méthodes d'affichage.
- Dans le package `test`, écrire une classe `TestTout` avec une fonction `main` qui propose une manipulation de tout le carnet d'adresse allant de la lecteur à partir d'un fichier, la sélection via recherche de chaîne et l'affichage de la sélection. Tester toutes les combinaisons d'affichage (croissant/décroissant, simple/abrégé/complet, nom_prénoms/prénoms_nom). Oui il en a 12.
- Sélectionner, valider et envoyer.

15. Bob

- Quand tout est fini, créer une étiquette git avec le numéro de version `v1.0`.
- M'envoyer un mail (stephane.pequignot@univ-lyon1.fr) avec le titre "[TP-GIT] Bob et Alice" (en remplaçant Bon et Alice par vos vrai noms) et contenant :
  - l'url du projet
  - une archive de la branche `master` du projet, générée de sur le site (bouton "*download zip*" dans *Repository*/*files*)