TP 3-4 : Système de Gestion de Versions - Git

Préambule

Pour ces 2 TP, vous allez utiliser GIT pour versionner des projets basiques. Le but de ces TP est de vous faire pratiquer :

1. l'ajout d'un projet sur un Gitlab (création du dépôt)
2. la récupération d'un projet existant depuis un dépôt Gitlab
3. la synchronisation des sources d'un projet à plusieurs utilisateurs
4. la résolution des conflits
5. application au projet "geometrie" de votre équipe des TP 1 et 2.

Avant de démarrer le TP, connectez vous sur le Gitlab de l'ISTIC `https://gitlab.istic.univ-rennes1.fr/` (à ne pas confondre avec le Gitlab de Rennes 1 `https://gitlab.univ-rennes1.fr/` sur lequel vous n'avez pas les droits).

TP 3 : Initiation à Git

3.1 Création d'un projet sur un dépôt Gitlab et import initial

Vous aller initialiser un projet sur Gitlab, l'importer en local, ajouter des dossiers/fichiers et propager les modifications sur le Gitlab.

1. Sur le Gitlab : Créez un nouveau projet (*Create blank project*) ;
2. Sur le Gitlab : Ajoutez des participants (par exemple, la personne assise à côté de vous) : Manage>Members ;
3. Trouvez l'URL du projet sur la page principale du projet ;
4. Dans un terminal, importez le projet vide que vous venez de créer (avec **clone**) ;
5. ajouter un dossier "doc", et dans celui-ci un fichier "README.txt" qui contiendra une ligne du type `Les contributeurs de ce projet sont :` suivi à la ligne suivante, de votre nom et e-mail universitaire.
6. Comme vu en cours, ajoutez dossier et fichier aux éléments à versionner (**add**), puis signalez au Git local les modifications (**commit**) et envoyez les modifications au Gitlab (**push**).
7. Dans le Gitlab : Consulter la page Web du dépôt et vérifiez que votre projet a bien été déposé.

3.2 Partagez votre dépôt avec la personne assise à côté de vous

1. Elle consulte la page Web du dépôt et trouve l'URL pour cloner votre projet ;
2. Dans un terminal, elle importe le projet (avec **clone**).

3.3 Modification

Modfifier le fichier "README.txt" que vous venez de cloner en y ajoutant votre nom et e-mail.

3.4 Synchronisation des sources à plusieurs utilisateurs

Pour envoyer vos modifications, utilisez les opérations **add**, **commit**, et **push** de Git. Pour récupérer les modifications, utilisez l'opération **pull**.

3.5 La résolution des conflits

1. Avec votre voisin ou voisine, dans le même fichier, du même projet, modifiez la même ligne de deux façons différentes ;
2. Propagez les modifications sur le projet jusqu'au dépôt **add**, **commit**, **push** ;
3. Récupérez leurs modifications sur le projet ;
4. L'un des deux binômes devrait avoir un conflit ;
5. Le binôme qui voit le conflit doit le résoudre et propager la résolution jusqu'au dépôt : **add**, **commit** et **push** sur le fichier responsable du conflit ;
6. Reproduisez l'opération pour que le second binôme observe, à son tour, un conflit.

TP 4 : Application au projet "geometrie" de votre équipe

4.1 Création d'un projet sur un dépôt Gitlab et import initial

Choissisez un membre de votre équipe (et un seul !) qui va initialiser le projet sur Gitlab, l'importer en local, ajouter des dossiers/fichiers et propager les modifications sur le Gitlab de la manière suivante :

1. Sur le Gitlab : Créez un nouveau projet (*Create blank project*) dont le nom sera L2GEN suivi de numéro de votre groupe de TP suivi de votre nom d'équipe.
2. Sur le Gitlab : Ajoutez l'ensemble des membres de votre équipe **plus votre encadrant de TP**, avec le rôle de "Maintainer".
3. Trouvez l'URL du projet sur la page principale du projet ;
4. Dans un terminal, importez le projet vide que vous venez de créer (avec **clone**) ;
5. y copier le résultat du TP2, c'est à dire le dossier `src` et tout ce qu'il contient (notamment le package `fr.univrennes.istic.l2gen.geometrie`).
6. Ajoutez le dossier dossier `src` et et tout ce qu'il contient aux éléments à versionner (**add**), puis signalez au Git local les modifications (**commit**) et envoyez les modifications au Gitlab (**push**).
7. Dans le Gitlab : Consulter la page Web du dépôt et vérifiez que votre projet a bien été déposé.

4.2 Partagez ce dépôt avec l'ensemble de l'équipe

1. Chacun consulte la page Web du dépôt et trouve l'URL pour cloner votre projet ;
2. Dans un terminal, importer le projet (avec **clone**).

Chacun, y compris votre encadrant de TP doit maintenant avoir une copie locale du votre projet.

4.3 Importation dans VS Code

Importer ce projet dans VS Code (Fichier>Open Folder).

4.4 Éviter les conflits simples liés à la mise en forme du code

Quand vous allez partager du code et modifier le code des autres membres du projet, beaucoup de conflits (inutiles) seront liés aux différences de formatage du code. Par exemple deux commits de la forme :

```
Groupe arbre(IForme figure) {
      if (figure == null) return null;
      Groupe groupe = new Groupe(figure);
      IForme mini = figure.dupliquer();
      mini.redimensionner(0.5, 0.5);
      groupe.ajouter(mini);
      return groupe;
}
```

```
Groupe arbre(IForme figure) {
  if (figure == null)
    return null;
  Groupe groupe = new Groupe(figure);
  IForme mini = figure.dupliquer();
  mini.redimensionner(0.5, 0.5);
  groupe.ajouter(mini);
  return groupe;}
```

seront en conflit à cause d'une mise en forme différente sur un code identique (sauts de lignes, tabulations, espaces etc.). Pour éviter cela, une bonne pratique est que tous les développeurs du projet utilisent une **mise en forme imposée** pour le code. Pour activer cela dans VScode : File>Preferences>Editor>Format on save.

4.5 Mettre en place un `.gitignore`

Par la suite, pour simplifier l'envoi de modifications et éviter de devoir faire **add** sur chaque fichier modifié, vous pouvez utiliser **commit -a** qui fait un `add` et un `commit` sur **tous les fichiers locaux modifiés**. Ceci permet d'envoyer rapidement **toutes** les modifications locales sur les fichiers suivis au Gitlab. Pour cela, vous aurez besoin de dire explicitement à Git quels sont les fichiers **à ne pas suivre**. Cela se fait à l'aide d'un fichier `.gitignore` à ajouter à la racine du projet. Voici un fichier `.gitignore` type pour les projets Java gérés avec VScode :

```
# bloop, metals and vscode
.bloop
.vscode
.bsp
.metals
bin
```

Dans ce fichier, on exclut du versionnage les fichiers de configuration de VScode, Metals, etc `.bloop`, `.metals`, `.vscode`, etc. ainsi que les répertoires utilisés par Java pour enregistrer les versions compilées des projets Java `bin/`. Ce fichier `.gitignore` est à placer par un membre de l'équipe à la racine du projet et **à ajouter au versionnement** avec `git add`, `commit` et `push` comme précédement.

4.6 Messages de Commit

Dans un développement collaboratif le message que vous écrivez lors d'un commit revêt une importance particulière pour la compréhension mutuelle au sein de l'équipe, et aussi pour celle de l'historique du projet. A cet égard on utilise souvent la convention de préfixer le message par les mots clés **fix :** et **feat :** de la manière suivante :

fix : indique que ce commit a corrigé un bug (dire lequel)

feat : indique que ce commit a introduit une nouvelle fonctionnalité (dire laquelle)

Plus de détails sur `https://www.conventionalcommits.org`.

4.7 Comparer deux versions du code

Pour comparer 2 versions d'un fichier, vous pouvez utiliser l'extension Git pour VScode (l'installer si ça n'est pas fait). Puis cliquer sur  . Cliquez sur un fichier sur lequel il y a eu une modification depuis le dernier commit. Il est affiché dans VScode dans 2 fenêtres avant et après modification. Vous avez aussi accès à des boutons pour remplacer le **add** et le **commit**.

4.8 Terminer le TP 2

Vous avez maintenant toutes les armes pour terminer dans de bonnes conditions le TP 2 : ajout de nouvelles fonctionnalités (si pas encore fait), passage à une API fluent, utilisation de l'héritage pour factoriser le code commun.

Répartissez-vous le travail, faites les push dans Git au fur et à mesure que chacun avance et assurez vous que le programme de la question 1 du TP 2 fonctionne correctement. C'est **critique** pour la suite des TP de GEN.

4.9 Générer le Javadoc

Générer le Javadoc de votre package geometrie et rendez-le à votre encadrant de TP.