TP 2 : Prise en main Git
Algorithmique – Programmation FIP (ING39)

V. Aponte, P. Courtieu

Septembre 2020

Ce Tp vous prépare à la méthode de travail à adopter TOUT LE TEMPS par la suite, ET AUSSI au premier Tp noté. Faites le consciencieusement.

MANTRA : *PROJET* GIT ≠ *PROJET* ECLIPSE(Maven//netbeans/vscode)

Relisez le paragraphe suivant jusqu'à l'avoir compris.

RÈGLE 1 : Jamais de **projet git** à l'intérieur d'un autre **projet git**.
RÈGLE 2 : Jamais de **projet Eclipse** à l'intérieur d'un autre **projet Eclipse**
RÈGLE 3 : Jamais **plusieurs projets git** dans **un projet Eclipse**.
RÈGLE 4 : **plusieurs projets Eclipse** dans **un seul projet git** OK mais sous-répertoires **complètement distincts**.

Sur le gitlab du CNAM (gitinfo.cnam.fr), vous pouvez créer des projets personnels. Vous pourrez le faire pour vos projets tout au long de l'année.

Dans ce cours nous faisons le choix suivant : nous avons créé pour vous un projet GIT où nous mettrons *tous vos TP*. Il s'agit donc de respecter la règle 4 ci-dessus : un projet git contenant plusieurs projets Eclipse *dans des sous-répertoires distincts*. En particulier il ne faut pas créer un projet Eclipse ou Maven à la racine du projet git.

Les TPs seront ajoutés (par nos soins) dans votre projet gitlab et il vous suffira de faire `git pull` régulièrement pour voir les ajouts. À chaque fois que vous avez fini une question : vous commiterez et pousserez les fichiers modifiés. Cela vous permettra de ne pas perdre votre travail et cela nous permettra de voir ou vous en êtes (et de vous noter le cas échéant). IL N'Y AURA PAS D'AUTRE FAÇON DE RENDRE VOS TP NOTÉS.

1 Premier pas

Clonez l'archive depuis le terminal :

```
git clone https://gitinfo.cnam.fr/fip_2023/tp_fip1_java_xxxx.git
```

Cette commande crée un répertoire du même nom que le projet Git. Comme expliqué dans le cours ce répertoire contient

— la copie de travail : le projet dans sa version la plus récente,
— tout l'historique (stocké dans le répertoire `.git`).

1.1 Reprise du TP 0

Vous avez récupérer votre archive, elle contient déjà plusieurs sous-répertoires :
— `00-tpbasejava` qui contient le squelette initial du tp 0 (prise en main, code sans objet),

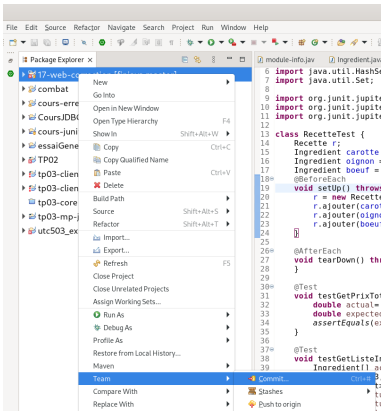
— `01-tptestsansobjet` qui contient le squelette initial du tp 1 (test sans objet).

Ces deux répertoires sont deux projets Maven séparés (dans la même archive git). Ouvrez-les dans Eclipse (Import → Existing Maven Project).

1.2 Un premier commit

Modifiez les classes du tp 0 en reprenant les modifications que vous aviez faites pendant le TP. Vous pouvez procéder soit en faisant un copier-coller, soit en écrasant le fichier par sa nouvelle version.

Une fois que le TP fonctionne (ou au moins que le premier exercice fonctionne et que le reste compile sans erreur), faites un premier commit à l'aide d'Eclipse : clic droit **sur le projet** Tp0 (important, ne pas cliquer sur un seul fichier ou package) ⇒ Teams ⇒ Commit.



Sélectionnez les fichiers que vous voulez commiter (unstages ⇒ staged). La plupart du temps il faut sélectionner tous les fichiers modifiés et décider quels nouveaux fichiers sont à ajouter. Pour ces derniers : éviter de commiter les fichiers de configuration personnel (`.project`, etc).

Mettez un message de commit (obligatoire). ⇒ Commit. Tapez votre mot de passe éventuellement. Les petits sigles (`>`) qui indiquent que les dossiers n'ont pas été « commités » devraient disparaître du navigateur de packages.

Vos modifications sont maintenant conservées dans l'archive locale... **mais pas sur le serveur**. Pour les envoyer au serveur, il faut sélectionner `Teams/Push to origin`. Tapez vos identifiants, validez.

Allez voir sur la forge si votre projet a bien été enregistré.

1.3 Continuez

Finissez le Tp0 et le TP1, COMMITEZ ET POUSSEZ à chaque fois qu'une nouvelle partie fonctionne. N'oubliez pas d'ajouter les tests du TP1.

Vérifiez régulièrement sur le serveur gitlab que vos push ont fonctionné.

Au moins une fois : faites le commit depuis un terminal, demandez de l'aide si nécessaire.

2 Récupération de code

Une fois votre code commité, remplacez le code d'implantation de votre méthode par le code de votre voisin (ne sauvegardez pas votre fichier, git vous permettra de le récupérer). Essayez de voir si vous pouvez y trouver des erreurs en jouant vos tests sur ce code !

Vous pouvez voir que vos fichiers ont éété modifiés depuis le dernier commit avec `git status`.

Lorsque vous avez fini, rétablissez le répertoire de la manière suivante :

```
cd <racinde du projet git>
git checkout 01-tptestsansobjet
```

3 Visualisation de code plus ancien

Dans le répertoire du projet exécutez cette commande :

```
git log --oneline
```

Cette commande affiche la liste des commits de l'archive. Les numéros précédent les message permettent de désigner un commit. Vous pouvez voir le contenu d'un fichier

```
git show HEAD@{numéro de commit}:un/certain/fichier
```

Il existe de nombreuses interfaces graphiques permettant de naviguer dans l'historique. Sur les machines du département d'informatique vous disposez de `gitk`.

4 Finissez

Si ce n'est pas déjà fait, attaquez l'exercice optionnel du Tp1 en suivant la même méthodologie : commitez et poussez à chaque fois qu'un exercice est fini.

RÈGLE IMPORTANTE (surtout en entreprise) ON NE COMMITE QUE QUAND ÇA NE GÊNERA PERSONNE (donc quand ça compile et que ça marche normalement).

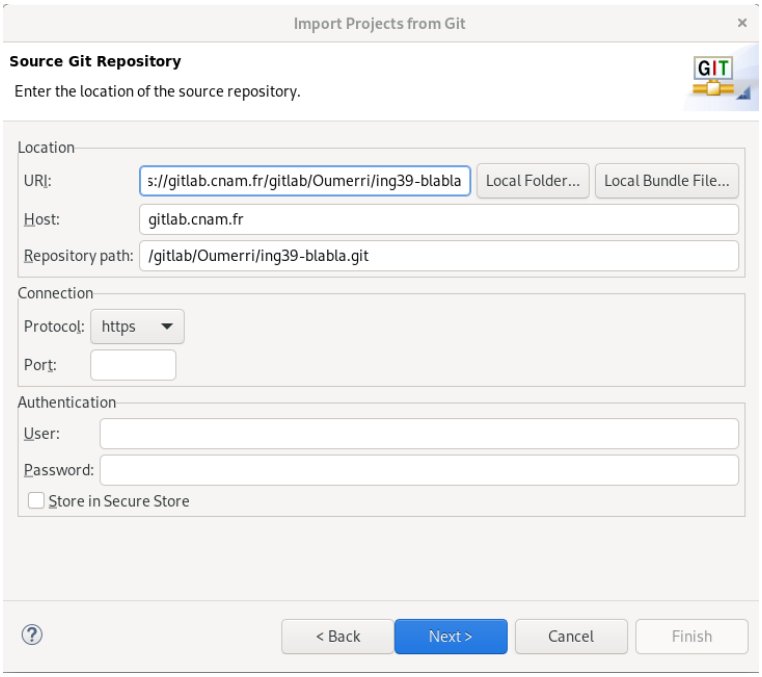
A Gitlab

Marche à suivre pour créer un projet git sur gitlab :
— ouvrez un navigateur web ;
— connectez-vous sur la forge ;
— cliquez sur `New Project` ⇒ `create blank project` pour ajouter un nouveau projet ;
— Cochez la case `Initialize repository with a README`.
— Si vous voulez rendre le projet visible aux enseignants, mettez le dans le group `ing39_<votreannée>` (à ne pas confondre avec `ing39_<votreannée>_user`). Ou bien suivez les étapes ci-dessous.
Pour donner accès à votre projet aux enseignants :
— Sélectionnez votre projet, cliquez sur `Project information` puis `Members`
— ajoutez les utilisateurs Maria-Virginia Aponte, Serge Rosmorduc et Pierre Courtieu comme `Reporter` de votre projet (pour que nous puissions le lire).

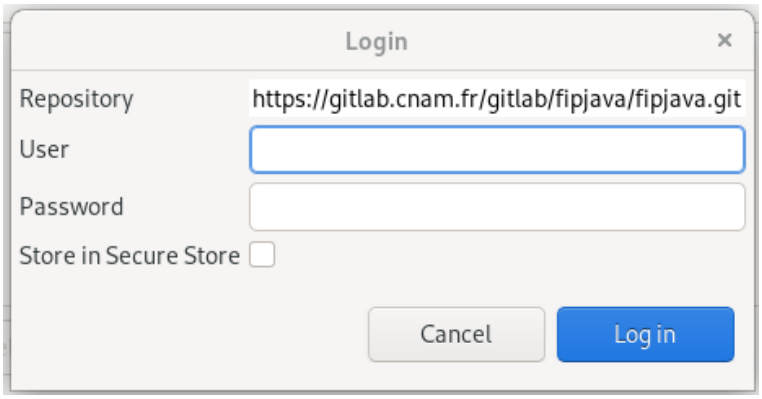
B Eclipse

Cloner un projet depuis Eclipse

Ceci ne marche que si l'archive contient déjà au moins un projet Eclipse.
Menu `Files` ⇒ `Import` ⇒ `Project from git` ⇒ `clone URI`
Remplissez le formulaire :
— URL du projet gitlab (`https` uniquement). L'URL de votre projet vous est donnée sur gitlab (projet nommé `ing39_votrelogin`). Les cases `Host` et `path` se remplissent automatiquement.
— PAS votre login tant que vous travaillez sur un compte local générique
— PAS votre mot de passe tant que vous travaillez sur un compte local générique



⇒ `Next`.
Tapez vos identifiant gitlab (mais ne cochez pas la case « store in Secure Store »).

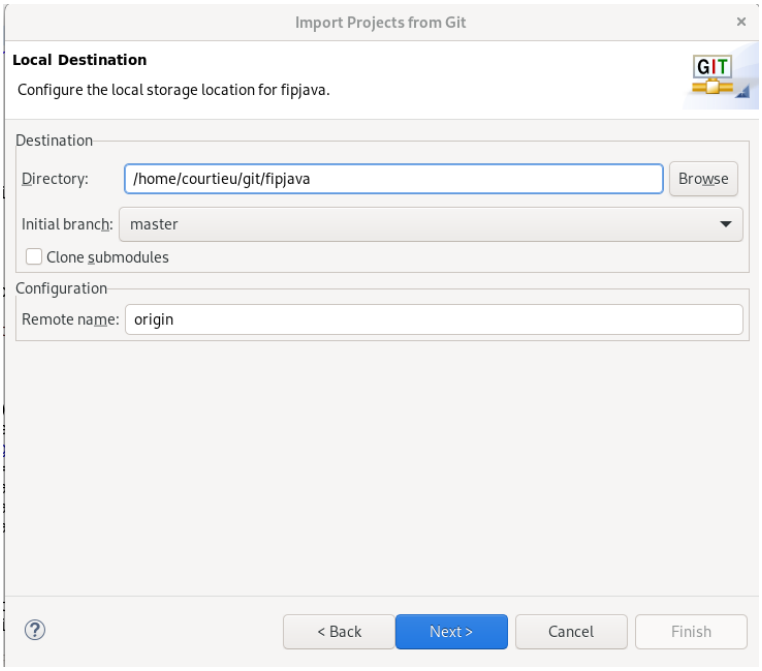


⇒ `Log in`.

2e formulaire : a priori ne touchez à rien.

⇒ `Next`.

3e formulaire : vous pouvez choisir l'emplacement local où cloner (télécharger) l'archive. NE PAS choisir un (sous-)répertoire d'une autre archive git (cf règle 1) ou d'un autre un projet Eclipse (règle 2).

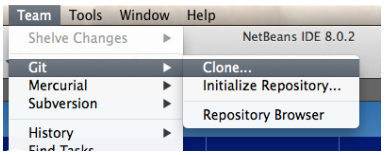


⇒ `Finish`.

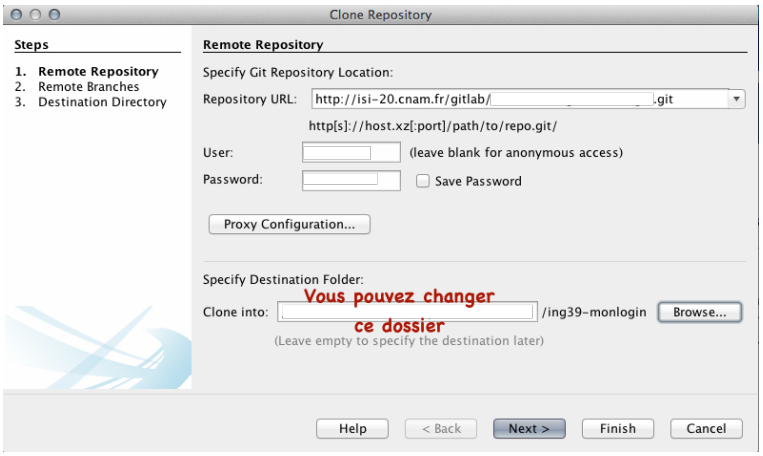
C Netbeans

Cloner un projet depuis Netbeans

Lancez Netbeans. Placez vous dans l'onglet `Services` de la fenêtre en haut à gauche. Cliquez sur `Team/Git/clone`

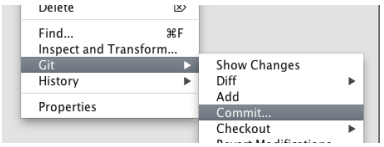


Remplissez le formulaire :



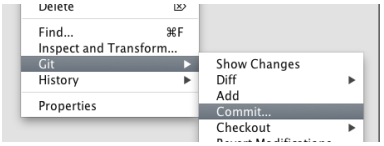
L'URL de votre projet vous est donnée sur gitlab. Cliquez sur finish, et acceptez la création d'un projet, que vous nommerez Tp0.

Pour faire un commit :

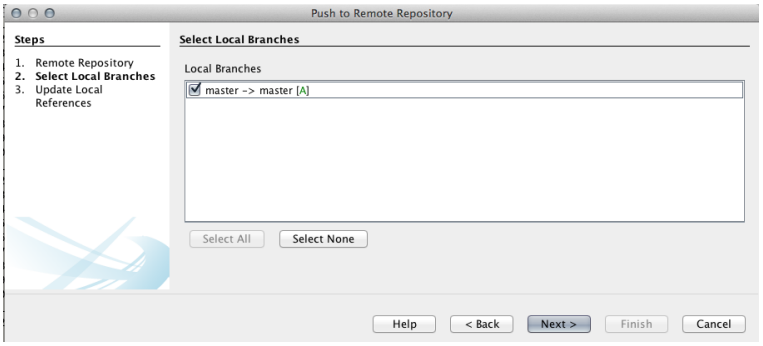


C.0.1 Commiter dans netbeans

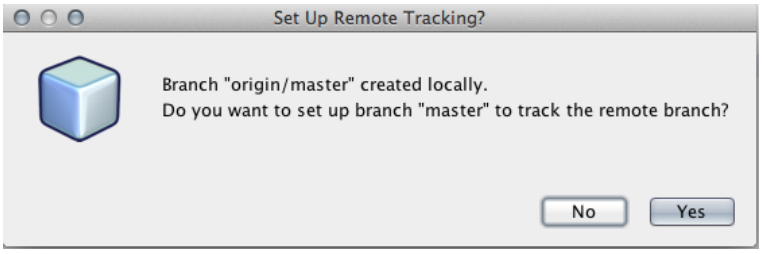
Faites un premier commit de Tp0,



Pour envoyer au serveur le commit, il faut sélectionner Git/Remote/Push. Une fenêtre nommée « Push to Remote Repository » apparaît. Cliquez sur `Next.`



Si la branche « master » n'est pas déjà cochée, cochez-là. Puis `Next` et `Finish`.
La fenêtre suivante s'ouvrira :



cliquez sur `Yes`. (attention, ne pas presser « entrée »)
Allez voir sur la forge si votre projet a bien été enregistré.