DLP : Développement d'un langage de programmation

Master STL, Sorbonne Université

Année 2022–2023

TME 0 – Environnement des TME

Antoine Miné

Ce document décrit l'environnement de travail en TME et leur déroulement général, de l'installation au rendu. L'environnement sera repris lors des épreuves sur machine.

1 Logiciels utilisés

Nous utiliserons les logiciels suivants :

- Le langage **Java**, version 8 ou supérieure (la version 11 installée à la PPTI, par exemple, convient).
- **Eclipse IDE** : un environnement de développement pour Java.
  Cet environnement gérera votre projet localement sur votre compte à la PPTI ou votre ordinateur personnel. Il vous permettra d'éditer les sources, de les compiler et de les exécuter.
- **JUnit 4** : un *framework* de test unitaire.
- **ANTLR 4** : un générateur d'analyseurs syntaxiques.
- **Boehm GC** : un gestionnaire automatique de mémoire (en option).
  S'il est installé, le ramasse-miettes Boehm GC sera utilisé par la bibliothèque d'exécution C du compilateur ILP afin de libérer automatiquement la mémoire.
- **GitLab** : un gestionnaire de projets logiciels.
  Il vous permettra de récupérer les sources d'ILP nécessaires au développement des TME et pour effectuer les rendus de TME. Vos sources locales (gérées par Eclipse) seront synchronisées avec celles du serveur (GitLab) grâce au gestionnaire de versions git, ce qui vous permettra de travailler sur différents ordinateurs et de partager votre projet avec votre binôme et votre chargé de TME, sans jamais perdre vos sources.
  Nous y avons ajouté une fonction d'intégration continue : toute propagation des modifications locales vers le serveur GitLab exécutera automatiquement une batterie de tests pour valider votre nouvelle version.
  Nous utilisons ici un serveur GitLab privé, dédié au cours : `https://stl.algo-prog.info`, et pas le site public `gitlab.com`.

Les sources d'ILP étudiées en cours sont distribuées via le serveur GitLab privé. Elles sont découpées en quatre niveaux, ILP1 à ILP4, de complexité croissante. Les premiers TME s'effectueront sur ILP1.
Après avoir configuré une fois pour toutes votre environnement à la PPTI (Sec. 2) ou sur votre ordinateur personnel (Sec. 3) et activé votre compte sur le site GitLab du cours (Sec. 4), le travail de TME fonctionnera de la manière suivante (Sec. 5) :

1. création sur le serveur GitLab d'une copie privée d'ILP (*git fork*) ;
2. importation du projet en local dans Eclipse (*git clone*) ;
3. travail en local sous l'IDE Eclipse [1] ;
4. synchronisation de vos changement locaux vers le serveur GitLab (*git push*) ;
5. si vous continuez le travail sur un autre ordinateur, importation (*git clone*) ou mise à jour (*git pull*) de votre projet local à partir du serveur GitLab ;
6. rendu final en synchronisant (*git push*) et en marquant (*git tag*) vos derniers changements sur le serveur GitLab.

2 Configuration de l'environnement à la PPTI

Quota. Pour vérifier qu'il vous reste suffisamment d'espace disponible sur votre compte, utilisez la commande `quota` dans un terminal. Si une étoile (`*`) apparaît, il ne vous reste plus de place ; vous devez absolument en libérer.

1. Vous êtes libres d'utiliser un autre IDE de votre choix, mais nous ne documenterons ici que l'utilisation d'Eclipse.

Java. Les machines de la PPTI possèdent plusieurs version de Java, dont Java 11 et 14. Toutes ces versions fonctionnent pour DLP.

Eclipse IDE. Plusieurs versions d'Eclipse cohabitent à la PPTI. Nous vous suggérons d'utiliser la plus récente en tapant, dans un terminal, la commande : « `eclipse-2020 &` ».
Si c'est la première fois que vous utilisez Eclipse, celui-ci vous demande le nom du répertoire *workspace*. Vous pouvez utiliser le nom par défaut `~/eclipse-workspace`, sachant que les projets gérés par git ne seront pas hébergés dans le *workspace*, mais dans un répertoire `~/git` séparé.

Configuration du proxy pour Eclipse. Si vous travaillez à la PPTI, cliquez dans Eclipse sur *Window* > *Preferences* puis *General* > *Network Connections*. Changez *Active Provider* pour qu'il indique *Manual*. Vous pouvez alors modifier les entrées HTTP et HTTPS pour ce *provider* en cliquant dessus puis sur *Edit...*. Pour les entrées HTTP et HTTPS, choisissez comme *Host* : `proxy.ufr-info-p6.jussieu.fr` et comme port 3128.

Configuration du proxy de navigation. À la PPTI, vous devez également configurer le proxy de votre navigateur. Choisissez, pour les protocoles HTTP et HTTPS, le serveur `proxy.ufr-info-p6.jussieu.fr` et le port 3128.

Configuration de proxy pour git. À la PPTI, afin d'éviter les erreurs d'authentification avec le serveur git, vous devez ouvrir un terminal et y taper les commandes suivantes :

```
git config --global http.sslVerify false
git config --global http.proxy http://proxy.ufr-info-p6.jussieu.fr:3128
git config --global https.proxy https://proxy.ufr-info-p6.jussieu.fr:3128
```

(note : les deux dernières commandes ne sont utiles que si vous utilisez git en ligne de commande, tandis que la première est aussi nécessaire pour travailler sur Eclipse).

Greffons. Nous vous conseillons d'utiliser les greffons Eclipse suivants, disponibles sur le *Marketplace* Eclipse :

— EGit – Git integration for Eclipse ;

— ANTLR 4 IDE

Allez dans le menu *Help*, option *Eclipse Marketplace...* ; entrez « egit » ou « antlr » dans la zone de recherche pour vérifier que ces greffons sont présents, et pour les installer si nécessaire.

3 Configuration de l'environnement sur une machine personnelle

Si vous travaillez sur votre ordinateur personnel, vous pouvez installer gratuitement les logiciels nécessaires :

Java JDK. Installez le dernier JDK Java *Ready for use* disponible sur `https://jdk.java.net` (JDK 18 au moment où ce document est écrit). Alternativement, sur une distribution Linux telle que Debian ou Ubuntu, vous pouvez installer la version disponible dans le gestionnaire de paquets (OpenJDK 11 actuellement).

Eclipse IDE. Installez Eclipse IDE, disponible sur `https://www.eclipse.org/downloads`.
Durant l'installation, sélectionnez l'option « Eclipse IDE for Java Developers ». Après avoir lancé Eclipse, installez également les greffons Egit et ANTLR 4 IDE disponibles sur le *Marketplace* (voir section ci-dessus).

ANTLR 4. Le fichier jar d'ANTLR 4 est inclus dans l'arbre de source fourni pour les TME (dans `Java/jars`), il ne devrait donc pas être nécessaire de l'installer séparément. En cas de soucis, ANTLR 4 est disponible sur `https://www.antlr.org/download.html`

JUnit 4. Les fichiers jar de JUnit 4 sont disponibles sur `https://junit.org/junit4/`, lien *Download and install*.

GitLab est installé sur un serveur distant, vous n'avez rien à installer.
Par ailleurs, vous n'avez pas besoin de configurer les proxy de git, d'Eclipse ou du navigateur ; ceci est uniquement nécessaire quand vous utilisez un ordinateur de la PPTI.

4 Compte sur le serveur GitLab privé du cours

Un compte personnel sur le serveur GitLab est créé automatiquement pour vous avant la première séance de TME (vous ne vous inscrivez pas vous-même). Vous recevrez deux emails envoyés par `gitlab@stl.algo-prog.info` : un email indiquant la création de votre compte (« *Account was created for you* ») et un email indiquant que vous avez été ajouté au groupe DLP de l'année courante. Ces emails sont envoyés à votre adresse `@etu.sorbonne-universite.fr`, qui est généralement redirigée vers l'adresse que vous avez renseignée lors de votre inscription pédagogique.

Pour commencer, vous devez cliquer sur le lien « *Click here to set your password* » de l'email de création de compte et choisir un mot de passe spécifique au serveur GitLab. Vous pourrez ensuite accéder aux sources ILP sur le serveur GitLab du cours `https://stl.algo-prog.info`. Notez que :

— votre *username* pour la connexion est votre **numéro d'étudiant** ;

— votre *email* est celui de l'université, en `@etu.sorbonne-universite.fr` ;

— vous devrez choisir un mot de passe (fort et distinct de vos autres mots de passe, pour plus de sécurité) lors de votre première connexion.

Si vous ne trouvez pas cet email, pensez à le chercher dans votre boîte spam ou sur le webmail étudiant. Notez en particulier que, si vous utilisez Gmail, l'email est souvent répertorié comme spam et les liens de connexion sont désactivés. Il est alors nécessaire d'indiquer explicitement à Gmail que l'email n'est pas dangereux pour rendre le lien cliquable. Vous pouvez également utiliser le lien *Forgot your password* de la page de login `https://stl.algo-prog.info` et entrer votre email `@etu.sorbonne-universite.fr`.

Vous êtes automatiquement membre du groupe `DLP-XXXX`, où `XXXX` indique l'année en cours (par exemple, pour 2022–2023, de sera `DLP-2022oct`). Ce groupe contient déjà un projet `ILP1`, en lecture seule, qui fournit les sources d'ILP1 vues en cours et nécessaires au TME. Comme indiqué dans la section suivante, vous en ferez une copie privée modifiable afin de réaliser les premiers TME.

Il peut être utile de réduire les notifications envoyées par GitLab par email. Pour cela, sur la page de GitLab, dans le menu *Settings*, choisissez l'option *Notifications*, ouvrez le menu *Global notification level* et cliquez sur *Custom*. Une liste apparaît, et vous pouvez désélectionner certaines notifications. Nous conseillons en particulier de désélectionner *Failed pipeline* pour éviter d'être submergé de notifications lors de l'intégration continue. Assurez-vous que votre fenêtre de navigateur occupe tout l'écran. Si la fenêtre est trop petite, les menus de GitLab sont remplacés par des icônes, et vous ne trouverez plus les noms des menus tels qu'indiqué dans cet énoncé.

5 Déroulement d'un TME

5.1 Installation du TME

Fork du squelette de TME sous GitLab. Le projet `ILP1` du groupe `DLP-XXXX` est en lecture seule. Pour réaliser le TME et compléter ce projet, vous devez d'abord en faire **une copie sur le serveur GitLab** (*fork*) qui sera **privée** à votre binôme et vous. Pour chaque binôme, **un seul élève** fera le *fork* du projet, et y ajoutera son binôme et son chargé de TME pour partager le projet :

— Ouvrez le menu *Menu* en haut à gauche, puis *Projects* > *Your projects*, sélectionnez le projet `DLP-XXXX/ILP1`, cliquez sur le bouton *Fork* puis cliquez sur votre nom. Ceci crée le projet personnel privé nommé ***username***`/ILP1` (où *username* est votre numéro d'étudiant). Vous travaillerez désormais dans ce projet, où vous avez les droits d'écriture.

— Ouvrez le menu *Menu* en haut à gauche, puis *Projects* > *Your projects*, et sélectionnez cette fois le projet privé que vous venez de créer. Il doit avoir pour nom ***username***`/ILP1` et porter la mention *Forked from* ***DLP-XXXX/ILP1***. Cliquez sur *Projet information* dans le menu de gauche, puis *Members*.
Invitez votre binôme en lui donnant pour rôle ***Maintainer***.
Invitez également votre chargé de TME en lui donnant pour rôle ***Maintainer***.

Si c'est votre binôme qui a créé le projet et vous a ajouté avec les bons droits, vous devriez le trouver dans l'onglet *Projects* > *Your projects* du *Menu* sous le nom ***binome***`/ILP1` où *binome* est le numéro d'étudiant de votre binôme.

Importation du projet privé dans Eclipse. Dans Eclipse, choisissez dans le menu : *File* > *Import...* > *Git* > *Projects from Git* puis *Next*. Dans l'écran suivant, choisissez l'option *Clone URI* puis *Next*.
Sur la page du projet du serveur GitLab, cliquez sur le bouton *Clone* et copiez l'URI du dépôt git du projet ; elle est indiquée sous le titre *Clone with HTTPS* et a la forme : `https://stl.algo-prog.info/`***forkusername***`/ILP1.git`
où *forkusername* est le numéro de l'étudiant qui a fait le *fork* (vous ou votre binôme). Collez cette URI dans le champ *URI* sur Eclipse.
Prenez garde à bien importer votre projet de binôme, ***forkusername***`/ILP1`, et pas `DLP-XXXX/ILP1` ; vous ne pourrez

pas travailler dans ce dernier, qui est en lecture seule !
À la PPTI, vous devez utiliser le protocole HTTPS, donc une adresse commençant par `https://`. Une adresse commençant par `git@` utilise le protocole SSH qui ne fonctionnera pas à la PPTI.

Dans *Authentication* vous devez renseigner vos identifiants GitLab : votre *username* GitLab (i.e., votre numéro d'étudiant) et votre mot de passe associé. Prenez garde à bien entrer vos identifiants de compte GitLab personnel, pas ceux de votre binôme, même si c'est lui qui a fait le *fork* sur GitLab. Le projet GitLab est commun au binôme, mais la copie locale du projet est personnelle. Tout membre du projet GitLab peut importer (cloner) une copie locale.
Le reste de la fenêtre se remplit automatiquement ; cliquez sur *Next*. Les écrans suivants, *Branch Selection* et *Local Destination* sont pré-remplis à des valeurs acceptables ; il suffit de cliquer *Next*. Choisissez *Import existing Eclipse projects* et *Next* puis *Finish*.

Un nouveau projet `ILP1` apparaît dans le *Package Explorer* à gauche. La mention `[ilp1 main]` associée au projet indique que le projet est bien géré par git. La copie locale des fichiers est hébergée dans le répertoire `~/git`, et non votre *workspace*.

Clones multiples. L'importation (clone) d'un projet peut être faite simultanément par plusieurs membres du projet (par l'élève qui a fait le *fork* comme par son binôme) et sur plusieurs ordinateurs simultanément (en salle PPTI, chez vous, sur un portable, etc.). Cela vous permet de facilement travailler en binôme à la PPTI et chez vous. La section suivante explique comment synchroniser les différentes versions locales de votre projet et celle hébergée sur le serveur GitLab.

Vous pouvez maintenant commencer le TME 1 et revenir à ce document pour les instructions de synchronisation périodique de votre travail sur le serveur GitLab, et pour les instructions de rendu en fin de séance.

5.2 Synchronisation avec le serveur GitLab

Après avoir effectué une importation dans Eclipse (ou *clone* en terminologie git) en début de TME, nous avons travaillé sur une copie locale du projet. Il est nécessaire de synchroniser périodiquement votre projet local avec le projet GitLab pour :

- vous synchroniser avec votre binôme ;
- éventuellement synchroniser des copies locales sur plusieurs ordinateurs ;
- garder une trace des modifications et pouvoir éventuellement revenir à une version précédente en cas d'erreur ;
- lancer les tests d'intégration continue sur le serveur pour valider votre TME ;
- faire votre rendu hebdomadaire de TME (l'enseignant a accès aux fichiers et aux résultats des tests sur GitLab, mais pas à ceux de votre compte PPTI ou de votre ordinateur).

Les opérations utiles sont donc la propagation d'une copie locale vers le serveur (*push*) et depuis le serveur vers une copie locale (*pull*). Il est possible d'utiliser git, soit depuis l'interface graphique d'Eclipse (greffon EGit), soit en ligne de commande.
Vous pouvez consulter l'état des fichiers sur le serveur GitLab en utilisant le site web `https://stl.algo-prog.info`. Vous y trouverez la dernière version des fichiers et l'historique des modifications. Vous pourrez en particulier vérifier que le projet a bien été synchronisé pour le rendu de TME, et voir exactement ce que votre chargé de TME verra (et notera).

Push. Pour propager des modifications depuis le projet local vers GitLab, faites un clic droit sur le projet, puis *Team > Commit...*. Un onglet *Git staging* apparaît en bas, ou dans une fenêtre (l'interface varie d'une version d'Eclipse à l'autre ; votre interface peut donc varier sensiblement de celle décrite ici). La zone *Unstaged changes* contient la liste des fichiers modifiés (ou ajoutés) localement mais non encore présents sur le serveur. Déplacez les fichiers que vous voulez synchroniser (fichiers modifiés ou nouveaux fichiers) vers la zone *Staged changes*. Entrez un bref message décrivant les modifications dans *Commit message*. Vérifiez la validité des zones *Author* et *Comitter*. En principe, elles devraient être identiques et contenir `username <email>` où *username* est votre numéro d'étudiant. Elles doivent refléter l'identité de la personne qui a fait les modifications (pas forcément de la personne qui a fait le *fork* sur GitLab en cas de travail en binôme). Si c'est bien le cas, alors cliquez sur *Commit and push...* sinon, avant de cliquer, vous pouvez renseigner ces zones manuellement.

Pull. Pour récupérer localement des modifications disponibles sur le serveur GitLab, faites un clic droit sur le projet, puis *Team > Pull*.

Conflits. Si des modifications ont été faites sur le serveur (par exemple par une propagation, *push*, de votre camarade) depuis votre dernier *pull*, vous ne pourrez pas propager vos modifications locales directement ; git refusera avec une erreur. En effet, cela provoquerait des conflits entre deux nouvelles versions d'un fichier. git vous force à résoudre les conflits localement, avant de propager vos fichiers corrigés vers le serveur :

— Faites d'abord un *pull*.

— Les fichiers marqués d'un diamant rouge dans le *Package Explorer* à gauche sont les fichiers avec un conflit. git s'est efforcé de fusionner les modifications locales avec celles présentes sur le serveur, mais il a pu faire des erreurs ; vous devez examiner chaque fichier et corriger à la main les problèmes causés par la fusion. Les zones non fusionnées sont identifiées par des balises `<<<<<<`, `-----` et `>>>>>>` dans votre source Java. git vous indique de cette manière les deux versions disponibles (version locale et dernière version disponible sur le serveur). Il s'agit souvent de choisir une des deux versions, en supprimant les lignes redondantes et les balises.

— Après avoir examiné et éventuellement corrigé un fichier en conflit, vous devez faire un clic droit sur le nom du fichier dans le *Package Explorer* puis *Team > Add to index* pour indiquer que le fichier est maintenant correct. Le diamant rouge disparaît. Ceci doit être fait pour chaque fichier ayant un conflit.

— Après suppression de tous les conflits, vous devez faire un *commit*, avec *Team > Commit*. Le message de *commit* est renseigné automatiquement : il indique les fichiers qui étaient en conflit (vous pouvez bien sûr modifier ce message).

— Vous pouvez enfin faire un *push*.

Bonnes pratiques. C'est une bonne idée d'anticiper les conflits en **commençant toute session de travail par un *pull*** (ou un *clone*), pour repartir avec les dernières versions des fichiers disponibles sur le serveur, et en **terminant toute session de travail par un *push***, pour que vos modifications locales soient envoyées sur le serveur et puissent être importées par votre binôme ou vous-même sur un autre ordinateur.

5.3 Intégration continue sur GitLab

L'intégration continue est une pratique de développement logiciel consistant à s'assurer que, à chaque instant, le projet hébergé sur le serveur est correct et passe tous les tests. Le serveur GitLab est configuré pour l'intégration continue : après chaque propagation de votre copie locale vers le serveur (*push*), le code sur le serveur est compilé et les tests JUnit fournis sont exécutés.

Après votre premier commit, vous pouvez consulter le résultat des tests sur le serveur GitLab en cliquant sur votre projet, puis sur *CI / CD* et *Pipelines*. Une icône *V* vert indique que tous les tests sont passés correctement, une croix rouge signale qu'au moins un test a échoué, et un croissant bleu ou un symbole pause que les tests sont en cours et qu'il faut patienter (l'icône correspondant au dernier commit est également disponible sur la page du projet, au-dessus de la liste des fichiers).

Dans la page *CI / CD > Pipelines*, cliquer sur l'icône de la colonne *Status* permet d'accéder à une page avec le détail des tests. Il y a généralement un jeu de test par TME. Cliquer sur le nom du jeu de test vous montre une console avec le log complet d'exécution. Il est également possible, en cliquant sur l'onglet *Tests*, de voir un résumé de chaque test, qui indique s'il y a eu une erreur de compilation ou une erreur d'exécution pour chaque méthode de chaque classe de test.

Ce mécanisme vient compléter l'exécution des tests unitaires que vous devriez lancer périodiquement sur votre copie locale depuis Eclipse (au moins avant chaque propagation vers le serveur). Par ailleurs, le chargé de TME a accès aux rapports de tests sur le serveur GitLab, ce qui lui permet d'évaluer votre rendu de TME.

L'intégration continue est gérée par le fichier `.gitlab-ci.yml` à la racine de votre projet, qui spécifie les classes de test. Vous serez amenés à enrichir ce fichier pour ajouter les tests demandés en TME.

5.4 Rendu de TME

Chaque semaine, il est obligatoire de rendre le TME à votre chargé de TME en fin de la séance. Si vous le souhaitez, vous pouvez aussi rendre **une seconde version améliorée avant le début du TME suivant**.

Le rendu se fait en propageant vos modifications vers le serveur GitLab, comme indiqué à la sous-section 5.2, et en y **associant un *tag***. Pour cela après avoir effectué la synchronisation (*push*) :

— Connectez-vous sur la page de votre projet sur `https://stl.algo-prog.info`.

— Assurez-vous que votre chargé de TME est membre de votre projet, avec le rôle *Maintainer*.

— Vérifiez que toutes les classes demandées sont bien présentes dans GitLab et bien synchronisées avec le projet local visible dans Eclipse.

— Vérifiez également que les tests unitaires du TME lancés par l'intégration continue sur le serveur GitLab se sont exécutés correctement (voir 5.3).

— Dans le menu de gauche, sélectionnez *Repository* > *Tags* et cliquez sur *New Tag*.

— Donnez un nom à votre *tag*, comme « rendu-initial-tme1 » ou « rendu-final-tme1 » (pour le TME 1) selon qu'il s'agit d'un rendu partiel en fin de séance ou bien d'un rendu de TME finalisé.

— Cliquez sur *Create tag*.

— En cas d'erreur, il est toujours possible d'amender ou de créer un nouveau *tag*.

Il est impératif de créer un *tag* pour chaque rendu et de réaliser au moins un rendu par TME. Ces rendus réguliers sont évalués dans le cadre du contrôle continu.

6 Épreuves sur machine

Vous travaillez en binôme pendant les TME, mais les examens sont individuels. Il est donc indispensable que chacun se familiarise avec l'environnement de travail.

Chaque examen se déroulera sur une session vierge. Vous aurez accès aux sources ILP1 à ILP4 (identiques aux sources disponibles lors des TME), mais vous n'aurez pas accès à votre compte, ni au serveur GitLab privé du cours. Vous travaillerez uniquement en local, sur Eclipse. En examen, l'accès sera limité à l'intranet de la PPTI, avec notamment un accès à la documentation Java et aux transparents de cours. Tous les documents papier seront autorisés. Vous pourrez vous munir d'une clé USB, étiquetée à votre nom. Attention, en mode examen, seuls les formats FAT et ext2 sont supportés pour les clés USB (pas NTFS).

7 Liens utiles : logiciels et documentations

— Serveur GitLab privé du cours : `https://stl.algo-prog.info`

— Documentation de la PPTI (dont Java) : `https://www-ppti.ufr-info-p6.jussieu.fr`

— Documentation de git : `https://git-scm.com/book/en/v2`

— Documentation du greffon EGit : `http://www.eclipse.org/egit/documentation`

— JDK Java : `https://jdk.java.net/`

— JUnit 4 : `https://junit.org/junit4/`

— Eclipse IDE : `https://www.eclipse.org/downloads/`

— ANTLR : `https://www.antlr.org/`