

Utilisation de NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un EDI moderne (éditeur en couleur, projets multilingage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plate-forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'EDI NetBeans s'appuie sur cette plate-forme et s'enrichit à l'aide de plug-ins. *[source : wikipédia]*

Remarque : les sources des codes utilisés dans ce TP viennent des exemples de la documentation NetBeans.

1 Prise en main de l'environnement

1.1 Un premier projet NetBeans : Hello World

Maven, Ant et Graddle sont trois outils d'automatisation de la compilation en Java. Apache Ant ("Another Neat Tool") est une bibliothèque Java utilisée pour automatiser les processus de construction des applications Java. En outre, Ant peut être utilisé pour construire des applications non Java. Il faisait initialement partie de la base de code d'Apache Tomcat et a été publié en tant que projet autonome en 2000. Par de nombreux aspects, Ant est très similaire à Make, et il est suffisamment simple pour que tout le monde puisse commencer à l'utiliser sans prérequis particulier. Les fichiers de construction Ant sont écrits en XML, et par convention, ils sont appelés `build.xml`. Les différentes phases d'un processus de construction sont appelées "cibles". Le principal avantage de Ant est sa flexibilité. Ant n'impose aucune convention de codage ou structure de projet. Par conséquent, cela signifie que Ant demande aux développeurs d'écrire toutes les commandes par eux-mêmes, ce qui conduit parfois à de gros fichiers de construction XML difficiles à maintenir.

Apache Maven est un outil de gestion des dépendances et d'automatisation de la construction, principalement utilisé pour les applications Java. Maven continue d'utiliser des fichiers XML, tout comme Ant, mais d'une manière différente. Alors qu'Ant donne de la flexibilité et exige que tout soit écrit à partir de zéro, Maven s'appuie sur des conventions et fournit des commandes prédéfinies ("goals"). Maven fournit un support intégré pour la gestion des dépendances. Le fichier de configuration de Maven, qui contient les instructions de construction et de gestion des dépendances, est appelé par convention `pom.xml`. En outre, Maven impose également une structure de projet stricte.

Gradle est un outil de gestion des dépendances et d'automatisation de la construction qui a été conçu à partir des concepts de Ant et Maven. Gradle n'utilise pas de fichiers XML, contrairement à Ant ou Maven. Les fichiers de configuration sont plus petits puisque le langage a été spécifiquement conçu pour résoudre les problèmes de domaines spécifiques. Le fichier de configuration de Gradle est par convention appelé `build.gradle` en Groovy, ou `build.gradle.kts` en Kotlin. Nous utiliserons Groovy dans ce TP et en DEVMOB pour créer des applications Android.

Maven Pour écrire le programme Hello World, nous allons faire un *New Project > Java with Maven > Java Application*, le nommer HelloWorld. Si ce n'est pas fait par défaut, vous créerez une classe nommée HelloWorld.java. Compléter ensuite le code de la classe comme proposé dans le Listing 1.

```
1 public static void main(String[] args){
2     System.out.println("Hello World");
3 }
```

Listing 1 – Programme Java HelloWorld

Le programme sera ensuite compilé (commande Build) puis exécuté en utilisant la commande Run dans la barre de menu (Menu textitRun) ou à l'aide d'un clic droit Run File sur le fichier de la classe principale.

Ensuite pour déployer le projet, faire Clean and Build Main Project. Assurez-vous au préalable que votre projet est déclaré comme projet principal (clic droit Set as main project dans le menu contextuel).

Dans l'onglet File, le navigateur de fichier présente la séparation entre le code source et les fichiers compilés. Dans le dossier target, il y a un jar immédiatement déployable, si vous avez indiqué de le créer à la compilation (voir ci-après). Lancer ce jar en ligne de commande dans un terminal java -jar NOMDUJAR.jar.

Remarque : Le fichier pom.xml de Maven doit avoir une forme similaire au texte de la Figure ?? pour créer un jar exécutable en précisant le Manifest. Un fichier d'exemple pom.xml est disponible dans le serveur pédagogique.

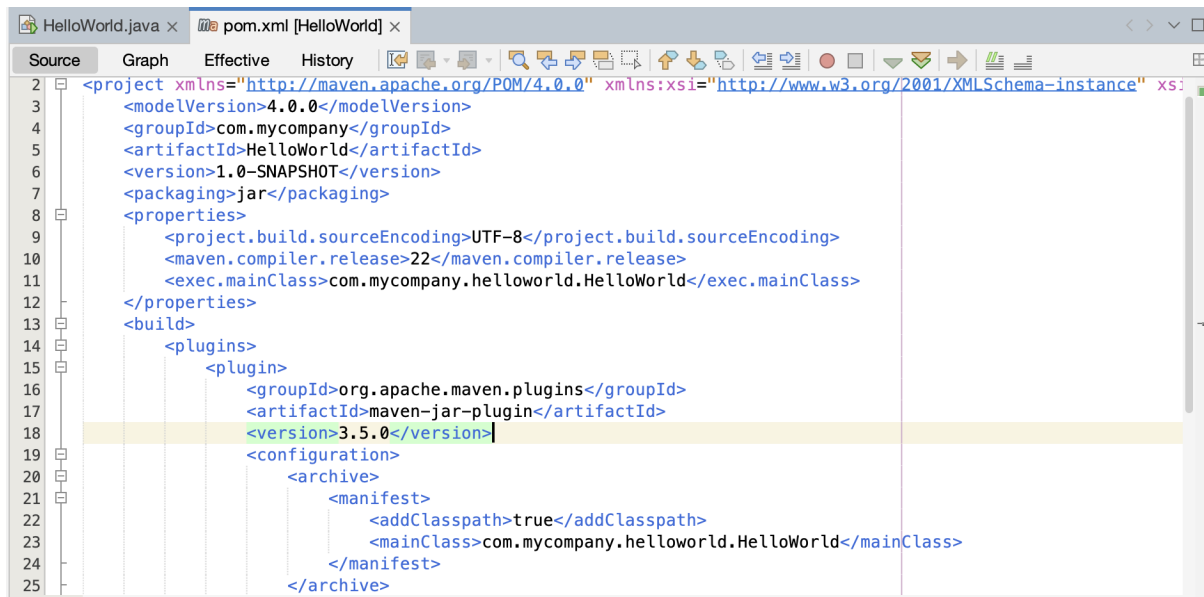


FIGURE 1 – Image d'un fichier pom.xml généré avec Netbeans 26 et un compilateur de Maven 22.

Le fichier jar créé se nomme HelloWorld-1.0-SNAPSHOT.jar, situé dans le dossier target de votre projet.

Ant Si vous vous faites le même projet "HelloWorld" avec Ant (*File>New Project> Java with Ant>JavaApplication*), les projets NetBeans créés avec Ant possède déjà un fichier `build.xml` que vous pouvez regarder. Il fait référence à un fichier `nbproject/build-impl.xml` qui est automatiquement généré par NetBeans en fonction des paramètres de votre projet et qui contient les cibles principales. Ne l'écrivez pas. Pour faire un jar, dans la vue Files, faire clic droit sur `build.xml` et faire **Run Target >jar**. Cela générera un jar dans le dossier `dist`.

Gradle Si vous utilisez Gradle (*File>New Project> Java with Gradle>JavaApplication*), à la création du projet Gradle génère automatiquement un projet Java avec une structure prédéfinie. L'ensemble des tâches est détaillé dans le fichier `build.gradle` et la création d'un jar en fait parti (lors du build). Il vous faudra compléter le fichier `build.gradle` pour créer un jar exécutable. Ajoutez la section `jar` comme dans l'exemple ci-dessous :

```
1 jar {
2     manifest {
3         attributes(
4             // a adapter au nom de la classe principale
5             'Main-Class': 'com.example.HelloWorld.gradle.App'
6         )
7     }
8 }
```

Listing 2 – Section jar dans build.gradle

1.2 Paramétrage pour le développement

1.2.1 Préférences générales

Le paramétrage de NetBeans s'effectue à l'aide de la fenêtre de préférences (**NetBeans>Settings**) (Mac) ou (**Tools>Options**) (Windows et Linux). En particulier le menu **Editor** permet de paramétrer le fonctionnement de l'éditeur, principalement à l'aide des onglets :

- **Formatting** : pour la mise en forme du code selon le langage choisi *Parcourez les catégories pour définir le formatage de votre code.*
- **Code Template** : liste des abréviations permettant de saisir rapidement du code. Regardez celles associées à java. *Dans l'éditeur de code tapez `sout` puis Tab. Vous pouvez aussi ajouter vos propres abréviations.*

Les préférences peuvent être exportées et importées, pour par exemple partager un paramétrage au sein d'un même projet.

KeyMap permet d'associer des raccourcis clavier aux actions les plus fréquentes. La plus fréquente étant la complétion automatique de code par **CTRL+espace** (ou un autre raccourci que vous aurez choisi).

Window>Action Item permet de voir la liste des éléments qui en commentaires vont provoquer un ajout dans la fenêtre **Action Item** de NetBeans. Faites un essai en ajoutant une ligne de commentaire dans votre code commençant par **TODO** ou **FIXME**. Vous pouvez aussi ajouter vos propres mots clés. La fenêtre **Action Item** permet de filtrer les "Actions" sur le fichier ou le projet actif. Elle donne aussi la liste des erreurs dans le code. Le filtrage peut être paramétré.

1.2.2 Menu Tool

Tool>Java Platform vous indiquera quel JDK est installé, où sont les classes et où trouver la javadoc. Vous pouvez ajouter le chemin d'accès à un zip contenant la javadoc si vous souhaitez travailler hors connexion.

Tool>Plugins vous donnera la liste des plugins installés et disponibles. Vous pouvez ajouter par exemple **SonarLint** qui va réaliser une analyse statique en temps réel du code Java et détecter les erreurs et mauvaises pratiques. Vous pouvez aussi installer **PlantUML** pour générer des diagrammes UML à partir de vos codes. Il vous faudra peut-être installer *GraphViz* pour que PlantUML fonctionne correctement.

Faites délibérément une classe **Point** avec 2 attributs *x* et *y* et un constructeur sans javadoc et ensuite utilisez le menu **Tool>Analyse Javadoc**. Vous verrez apparaître les erreurs de javadoc à corriger. Il existe une aide à la production de commentaires Javadoc. En se plaçant avant le code à commenter et en tapant `/**` puis "entrer", on obtient un squelette de Javadoc à compléter.

La javadoc sera ensuite générée par le menu **Run>Generate Javadoc** et sera placée dans le dossier `target>reports>apidocs`.

1.2.3 Paramétrage du projet

Le paramétrage des projets se fait à l'aide du menu **Properties** accessible par un clic droit sur le projet dans le navigateur de projet.

Library Vous permet d'ajouter les liens à des projets externes, des librairies ou de jar nécessaires à la compilation de votre projet. *Remarque : l'ajout est aussi possible par un clic droit sur le dossier Libraries dans le navigateur de projets.*

Build L'item **Build** vous permet de définir un ensemble d'options à la compilation, par exemple des options préprocesseur ou des options pour le compilateur.

Run L'item **Run** définit l'environnement d'exécution (Arguments en entrée, Répertoire de travail, Options de la machine virtuelle).

Licence Header Vous permet d'associer une licence spécifique insérée automatiquement en en-tête de vos fichiers de code.

1.3 Aide à la génération de code et Refactoring

Menu Source Dans le menu source, outre le formatage de code, il existe d'autres assistants utiles :

- **Toggle comment** : permet de transformer les lignes sélectionnées en commentaires et les commentaires en code ;
- **Move** ou **Duplicate Code** : pour déplacer ou copier des blocs de code ;
- **Organize/Fix imports** : analyse l'ensemble du programme et ajoute/corrige les **import** nécessaires au début du programme ;
- **Insert code**
 - **Override Methods...** : affiche une liste de toutes les méthodes héritées où l'on coche celles qui seront redéfinies, le squelette de ces méthodes est alors affiché ;
 - **Generate Getters and Setters...** : génère les accesseurs aux champs d'une classe ;
 - **Generate Constructor** : génère un constructeur à partir des champs de la classe choisie.

Générer les Getter and Setters pour la classe **Point** définie précédemment.

NetBeans propose une série de fonctions pour la transformation du code (en anglais *Refactoring*). Cela va permettre d'améliorer la structure du code sans modifier le comportement du programme. Ces fonctions se trouvent dans le menu **Refactor**. Les principales fonctions sont **Move**, **Rename** et **Copy** pour modifier des attributs, des noms de méthodes ou même des noms de Classes.

Un menu contextuel de Refactoring est aussi accessible directement dans le code. Utilisez-le pour modifier le nom de vos attributs.

1.3.1 Interfaces graphiques

NetBeans possède un outil d'aide à la création d'interfaces graphiques. Il est accessible en créant un nouveau fichier et en choisissant **Swing GUI Forms** (ou **AWT GUI Forms**), souvent dans "other" dans le menu contextuel. Choisissez par exemple de créer une Application en Swing (*Application Sample Form*). Ensuite les boutons **Source** et **Design** en haut de la fenêtre d'édition permettent de naviguer entre une vue Palette et le code source de l'interface graphique en construction.

1.4 Debug et profilage

Un 1er diagnostic Le menu **source > inspect** permet un diagnostic du code.

Debug Il existe un menu **Debug**. Le mode débog permet d'ajouter des points d'arrêts dans l'exécution du programme. Ensuite les icônes du mode débog permettent de faire tourner le programme pas à pas et de voir les valeurs des variables. Testez-le sur votre code de *projet d'objet* en ajoutant des points d'arrêts.

Profile Créer un nouveau projet en sélectionnant **Samples > Java with Ant> AnagramGame** et sélectionner le dans la fenêtre des projets (par défaut c'est le projet défini comme principal qui sera profilé). Vous pouvez aussi continuer à utiliser votre projet d'objet si vous préférez.

Vous pouvez faire plusieurs tâches de profilage différentes :

- Télémétrie : afficher les l'utilisation en temps réel de la CPU, de la mémoire, des appels au Garbage Collector et du nombre de Thread et classes chargés
- Méthodes : affichage des arborescences d'appels et des méthodes en cours d'utilisation, des temps d'exécution et éventuellement des numéros d'appel.
- Objets : affiche les objets avec leur nombre d'instances et la taille totale, avec éventuellement des arbres d'appel d'allocation.
- Threads : affichage des threads des processus, leurs temps et leurs états dans une chronologie.
- Verrous : affichage des verrous (et de leur propriétaire) et les threads bloqués avec leur nombre en attente et le temps d'attente total.
- Requêtes SQL : affiche les requêtes SQL invoquées à l'aide de l'API JDBC, y compris les comptages, les heures d'appel et les arborescences d'appels

Choisissez **Profile Project** et testez-les sur le projet Anagram. *Dans le menu CPU > Advanced vous pouvez choisir quelles méthodes profiler.*

1.5 Gestion des versions et travail en équipe

1.5.1 Local history

Pour chaque unité de compilation, NetBeans enregistre un nombre paramétrable de versions antérieures stockées à chaque fois par *Save*, c'est le *Local History*. Vous avez accès aux versions précédentes d'un fichier via **View>Editor>History**.

1.5.2 Utilisation de Git et GitHub

Pour gérer les versions successives d'un programme, NetBeans permet de se connecter à Git à travers le menu **Team**. Si nécessaire, pour revoir les commandes git (en ligne de commande) vous pouvez suivre le tutoriel suivant <https://try.github.io/levels/1/challenges/1>. Pour créer un projet sur GitHub commencez par vous créer un compte et ensuite suivez le tutoriel suivant : <https://netbeans.apache.org/tutorial/main/kb/docs/ide/git/>

1.5.3 Utilisation des tasks et Issues

Pour suivre un projet, vous allez utiliser la Planification et suivi avec Projects de GitHub : <https://docs.github.com/fr/issues/planning-and-tracking-with-projects>.

Vous allez créer un "User project" pour suivre les tâches de votre projet. Vous pourrez créer des "Issues" pour chaque tâche à réaliser et les assigner à des membres de votre équipe.

<https://docs.github.com/fr/issues/planning-and-tracking-with-projects/creating-projects/creating-a-project#creating-a-user-project>

Commencez par un projet vide "start from scratch" et faites une Table et créez des items en les assignant aux membres du projet. Naviguez entre les menus pour découvrir les possibilités de suivi.

Vous pouvez connecter Netbeans à GitHub Issues en téléchargeant le plugin *GitHub Issues Support* <https://plugins.netbeans.apache.org/catalogue/?id=37>. Une fois le plugin installé vous pourrez accéder aux issues de votre projet GitHub via le menu **Windows>Tasks**. Pour que cela soit possible il vous faudra au préalable ajouter un référentiel github

- Ouvrez une fenêtre de tâche (Windows > Tasks)
- Cliquez sur l'icône "Add a repository"
- Sélectionnez le connecteur GitHub Issues
- Saisissez le nom d'affichage, votre nom d'utilisateur, le jeton d'authentification et les informations relatives au *repository*. Votre Personnel Acces Token (classic) devra avoir à minima les droits sur les *repo*. Ajouter les droits sur les workflows, cela sera utile pour un TP ultérieur.
- Cliquez sur le bouton Se connecter (si vous ne parvenez pas à vous connecter à un référentiel, vérifiez les valeurs saisies)
- Cliquez sur le bouton OK

Une fois connecté vous pourrez voir les issues de votre projet GitHub dans la fenêtre des tâches. Vous pouvez filtrer les issues par état (ouvertes/fermées), par assignation, par étiquette, etc. Vous pouvez également ajouter des issues directement depuis NetBeans en cliquant sur l'icône "Add Issue" dans la fenêtre des tâches.

2 Mise en œuvre

Le but est de réaliser des traitements d'images simples sur des images binaires. D'abord vous devez charger des images à partir de fichiers au format PGM et les sauver. Pour simplifier, nous ne travaillerons qu'avec des images en niveau de gris de taille maximum fixée.

Le format PGM est simple, mais présente l'inconvénient d'avoir rapidement affaire à de gros fichiers. Un fichier PGM en format ASCII commence par une ligne contenant P2. La ligne suivante est une ligne de commentaire commençant par #. Sur la ligne suivante on trouvera les dimensions de l'image (largeur puis hauteur). La ligne suivante donne la plus grande valeur de niveaux de gris présente dans l'image (on écrira systématiquement 255). Suivent ensuite les valeurs des niveaux de gris de chaque pixel ligne par ligne. Seule contrainte supplémentaire, les lignes ne doivent pas contenir plus de 70 caractères.

Lecture/écriture Vous lirez et écrirez des images en PGM et donnerez leur histogramme sous forme d'image PGM.

Seuillage Vous coderez ensuite une fonction de seuillage et de différence entre deux images ainsi que des opérations d'agrandissement et de réduction de la taille des images.

GUI Vous ferez aussi une interface graphique pour accéder à ces différentes fonctions.

Le projet sera versionné sur github classroom. Merci de suivre le lien d'invitation disponible sur hippocampus.