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 IDE moderne (éditeur en couleur, projets multi-langage, 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'IDE NetBeans s'appuie sur cette plate forme.

L'IDE Netbeans s'enrichit à l'aide de plug-in. *[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

Pour écrire le programme `Hello World`, nous allons faire un *New Project >Java > Java Application*, le nommer `HelloWorld` et préciser les différents champs (l'utilisation d'un dossier à part pour stocker les bibliothèque et le nom de la classe principale).

Compléter ensuite le code de la classe principale créée par :

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

Le programme sera ensuite exécuté en utilisant la commande `Run` dans la barre de menu ou à l'aide d'un clic droit `Run File` sur le fichier de la classe principale.

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

Dans l'onglet `File`, le navigateur de fichier présente la séparation entre le code source et les fichiers compilés. Dans le dossiers `dist` qui vient d'être créé il y a un jar immédiatement déployable. Lancer ce jar en ligne de commande dans un terminal `java -jar HelloWorld.jar`.

1.2 Paramétrage pour le développement

1.2.1 Préférences générales

Le paramétrage de NetBean s'effectue à l'aide de la fenêtre de préférences `(NetBeans>Preferences)`. 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. *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`.

`Window>Action Item` permet de voir la liste des éléments qui en commentaires vont provoquer un ajout dans la fenêtre `Action Item` de NetBeans. Faite un essai en ajoutant une ligne de commentaire dans votre code commençant par `TODO` ou `FIXME`. Vous pouvez 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 `Spellcheck French` pour voir une correction orthographique en français. Le plugins `Find Bug integration` vous permet de lancer des outils de revue de code.

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 apparaitre 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 `dist`.

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.

Source Créer un projet switchTest avec une classe principale SwitchTest. Dans les `Properties>Sources` du projet, modifiez la source en JDK 6. Copiez ensuite dans la classe principale le code suivant :

```
public class SwitchTest {

public static void main(String[] args) {

    String color = "red";
    String colorRGB;
    if (color.equals("black")) {
        colorRGB = "000000";
    } else if (color.equals("red")) {
        colorRGB = "ff0000";
    } else if (color.equals("green")) {
        colorRGB = "008000";
    } else if (color.equals("blue")) {
        colorRGB = "0000ff";
```

```
        } else {
            colorRGB = "Invalid color";
        }
        System.out.println(colorRGB);
        }
    }
```

Après exécution, modifier la même propriété pour la repasser en JDK 7. Notez les suggestions de modifications du code et effectuez les.

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 permets d'associer une licence spécifique insérée automatiquement en entête de vos fichiers de code.

1.3 Une Application utilisant deux projets interdépendants

L'application vous créez contiendra deux projets :

— Un projet `Java Class Library` dans lequel vous allez créer une classe utilitaire : MyLib

— Un projet `Java Application` avec une classe principale qui implémente une méthode de la classe utilitaire du projet de `MyLib` : `MyApp`.

Le principe sera le suivant. Après avoir créé les projets, vous allez ajouter le projet `MyLib` au classPath du projet `MyApp`. Ensuite, vous coderez l'application. Le projet `Class Library` contiendra une classe utilitaire avec une méthode `acrostic`. La méthode `acrostic` prend un tableau de mots comme un paramètre, puis génère un acrostiche sur la base de ces mots. Le projet `MyApp` contiendra une classe principale qui appelle la méthode `acrostic` et transmet les mots qui sont entrés comme arguments lorsque l'application est exécutée. Il n'est évidement pas utile de faire deux applications pour cela, mais cela est à des fins d'illustrations.
Indiquer comme `Library` de `MyApp` le projet `MyLib`.
Créer une nouvelle classe dans `MyLib` avec `New > Java Class` et nommez la classe `LibClass` et le package qui la contient s'appellera `org.me.mylib`.
Dans la classe principale, complétez le code par

```
public static String acrostic(String[] args) {
        StringBuffer b = new StringBuffer();
        for (int i = 0; i < args.length; i++) {
            if (args[i].length() > i) {
                b.append(args[i].charAt(i));
            } else {
                b.append('?');
            }
        }
        return b.toString();
```

```
}
```

Remarque : Pour reformater le code utiliser clic droit puis `Format` *dans la fenêtre du code à reformater ou menu* `Source` *puis* `Format`.
Dans la méthode `Main` de `MyApp`, ajoutez les lignes :

```
String result = LibClass.acrostic(args);
System.out.println("Result = " + result);
```

Dans le champ `Arguments` du menu `Run` des propriétés de `MyApp` taper `However we all feel zealous` puis exécuter l'application.

1.4 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 où même des noms de Classes.

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

1.4.1 Interfaces graphiques

NetBeans possède un outil d'aide à la création d'interface graphiques. Il est accessible en créant un nouveau fichier et en choisissant `Swing GUI Forms` (ou `AWT GUI Forms`). Choissisez par exemple de créer une Application en Swing. 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.

1.5 Debug et profilage

Un 1er diagnostic Le menu `source > inspect` permet un diagnostic du code (après installation du plugin `Find Bug`).

Debug Il existe un menu `Debug`. Le mode débug permet d'ajouter des points d'arrêts dans l'exécution du programme. Ensuite les icônes du mode débug permettent de faire tourner le programme pas à pas et de voir les valeurs des variables. Tester le sur `MyApp` et `MyLib` en ajoutant des points d'arrêts.

Profile Créer un nouveau projet à partir du Project Wizard en sélectionnant `Samples > Java category > AnagramGame` et sélectionner le dans la fenêtre des projets (par défaut c'est le projet défini comme principal qui sera profilé).

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

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

1.6 Gestion des versions et travail en équipe

Pour chaque unité de compilation, NetBeans enregistre un nombre paramétrable de versions antérieures stockées à chaque fois par *Save*, c'est le *Local History*.

Pour gérer les versions successives d'un programme, NetBeans permet de se connecter à des serveurs SVN (Subversion) ou Git à travers le menu `Team`.

Vous pouvez par exemple vous connecter au serveur : `https://subversion.ec-nantes.fr:/svn/eleves/medev/tp2`avec vos login/psswd de centrale.

Pour utiliser svn avec NetBean, suivez le tutoriel suivant : `https://netbeans.org/kb/docs/ide/subversion.html`

Pour vous familiariser avec les commande 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 tutorial suivant : `https://netbeans.org/kb/docs/ide/git.html`

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 des 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 (largueur 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 le SVN de centrale. Attention à bien créer un dossier par groupe dans `medev/tp2` sous la forme Nom1_Nom2.