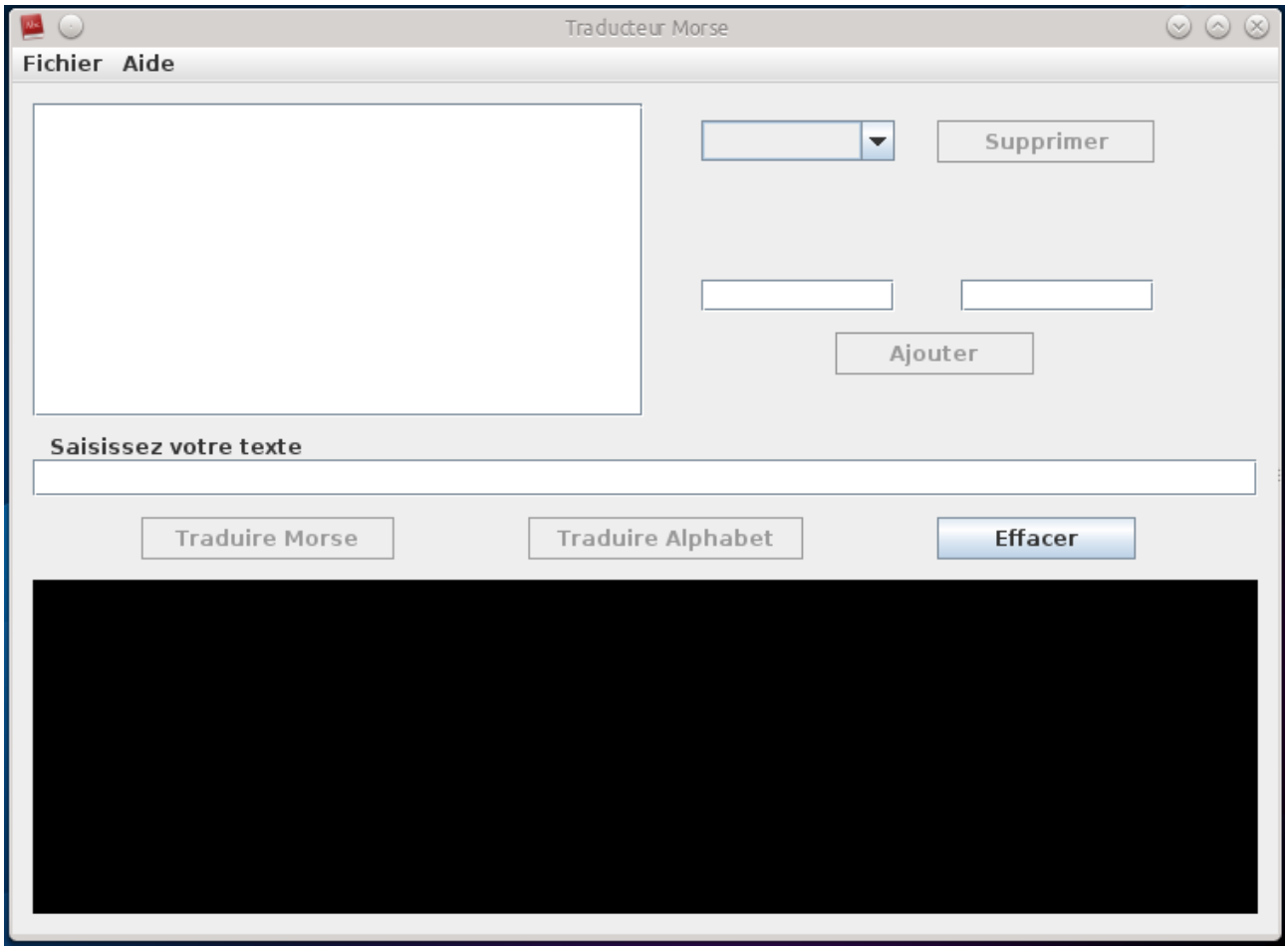
PROJET MORSE EN JAVA – LPSIL Génie Logiciel

Equipe Projet Maîtrise d'œuvre :
Kévin MASCHTALER – **Pascal CHEVALLOT**

Maîtrise d'Ouvrage : **monsieur Ahmed ZIDNA** – Professeur Algorithme / JAVA IUT de Metz- ahmed.zidna@univ-lorraine.fr

Le présent document, livré au format OpenDocument .ODT et PDF, récapitule les grandes réunions du binôme ; bien entendu, entre chaque session de travail en commun, chacun des membres de l'équipe travaillait sur des fonctionnalités du logiciel et effectuer un « commit » sur GitHub : la démarche collaborative du projet avec Git / GitHub est décrite ci-dessous.

Voici un aperçu de l'interface graphique utilisateur du Traducteur Morse, ainsi que les étapes vous permettant de le manipuler. Dans le menu « Aide », la section « A propos... » reprend les étapes principales pour utiliser ce traducteur.



1 – Lancer l'application en cliquant sur le fichier « **morse.jar** » présent à la racine du dossier MORSE_Maschtaler_Chevallot ;

2 – Depuis le menu « Fichier », Il faut tout d'abord charger le dictionnaire « dict.ini »

également présent à la racine du dossier MORSE_Maschtaler_Chevallot : dans la zone de notification, un message indique que le dictionnaire est chargé ;

3 – Depuis le menu « Fichier », vous pouvez choisir d'afficher le contenu du dictionnaire ;

4 – Une fois chargé, le dictionnaire permet de traduire du texte en caractères alphanumériques vers des chaînes de « - » et de « . » représentant le code Morse et vice-versa : vous pouvez, au choix, saisir du texte libre dans la zone prévue à cet effet et appuyer sur les boutons idoines : le texte traduit apparaît dans la zone de texte supérieur. Le bouton « Effacer » permet d'effacer ce texte.

5 - Vous pouvez également, depuis le menu « Fichier », chercher sur votre système de fichier un fichier texte .txt à traduire : la traduction est alors automatique, elle s'affiche dans la zone de texte supérieure et le fichier de même nom portant l'extension .morse est créé au même endroit que le fichier .txt original.

Pour vous assurer du fonctionnement correct de l'application, vous pouvez alors charger ce fichier .morse : la traduction en caractères alphanumériques apparaît alors dans la zone de texte supérieur et le fichier .txt est créé au même niveau que le fichier .morse

6 – La zone supérieure droite de l'interface vous permet de manipuler le dictionnaire, notamment de supprimer un caractère : une fois supprimé, vous pouvez tester en le saisissant dans la zone de saisie libre : la traduction ne se fait plus.

7 – Vous pouvez également ajouter un caractère alphanumérique à gauche et son équivalence morse à droite puis cliquer sur « Ajouter » : si par exemple, vous ajoutez le caractère préalablement supprimé, la traduction redevient fonctionnelle.

La classe **Morse.java** est la classe principale de l'application ; elle contient toutes les fonctions spécifiques au traducteur.

La classe **Tools.java** est une classe outil permettant de lire et d'écrire un fichier.

La classe **Main.java** est la classe qui gère l'interface graphique utilisateur de ce Traducteur Morse.

Le package « appmorse » embarque également un certain nombre d'icônes qui agrémentent l'interface graphique utilisateur.

1) - Réunion Lancement Projet 17/10/2013

Rappel de la commande présente dans le cahier des charges :

Le but de ce projet est de traduire un fichier texte en un fichier Morse et inversement. Il est vivement recommandé de décomposer le projet en plusieurs parties :
- les fonctions de base,
- les fonctions de haut niveau
- et enfin l'application.

Le projet doit entre autres :
1. Construire une liste de conversions à partir d'un fichier qui contient le code Morse.
Une conversion est une structure qui contient la lettre et son code Morse.

2. Effectuer des opérations de base sur cette liste de conversions,

3. Effectuer des opérations de haut niveau telles que : l'insertion d'un code, la suppression d'un code, la recherche d'un code dans une liste,....etc.

4. Traduire un fichier texte en un fichier Morse et inversement, traduire un fichier Morse en un fichier texte en utilisant la liste de conversions.
Extension : utiliser une interface graphique pour gérer la traduction.

Plan de travail :

A l'initialisation du projet, l'équipe de maîtrise d'œuvre prévoit le développement de :
- 1 classe 'Morse' avec :
    - table de conversions « lettre ↔ morse » = solution à définir : tableau ? Java.util.ArrayList ? Tree ? Map ?
    - Fonctions de base : ajouter, retirer une ligne dans le dictionnaire de conversions ; trouver les correspondances entre les deux langues.

- 1 application console 'convertisseurMorse' où l'utilisateur saisira le texte à traduire.
- si possible, 1 application graphique 'convertisseurGraphiqueMorse' de type 'Google Translate'.

- Gestion du projet : l'équipe de maîtrise d'œuvre utilisera le logiciel libre GIT afin de maîtriser, de façon décentralisée, l'évolution collaborative des versions du logiciel, tout au long du développement. Elle s'appuiera en outre sur le service d'hébergement de GITHUB (https://github.com).
- La branche principale du projet se trouve à l'adresse :
https://github.com/Kmaschta/lp_morse
- Le fork du projet se trouve à l'adresse :
https://github.com/pchevallot/lp_morse

2) - Réunion de travail sur le code – 15/11/2013

- le choix de ***HashMap*** (tableau dynamique associatif) est confirmé (ArrayList n'est pas retenu : le fait de redimensionner demande beaucoup de ressources) : il nous apparaît en effet particulièrement bien adapté à notre problématique de dictionnaire. C'est en outre un algorithme très performant qui ***associe une clé à une valeur***. HashMap utilise une table de hachage pour implémenter l'interface Map. HashMap assure des requêtes en temps constant : cela peut s'avérer performant dans le cas où le texte à traduire est volumineux.

- réalisation de la fonction initDictionary et du dictionnaire Alp̂habet – Morse dans un fichier 'dict.ini' : cette fonction permet d'initialiser la classe Morse grâce au fichier 'dict.ini' qui contient la lettre ou le caractère, ainsi que son code morse associé. Utilisation de la fonction 'enregistre' pour charger les conversions ('put').

- réalisation de la fonction readFile : grâce à la classe Scanner, on crée un flux de lecture de fichier pour charger dans une chaîne le fichier à traduire.

3) - Réunion de travail sur le code – 22/11/2013

Réalisation de :

- de la fonction alphaToMorseChar
- de la fonction morseToAlphaChar
- de la fonction alphaToMorse
- de la fonction morseToAlpha
- de la fonction getDictionnary en utilisant l'interface Map.Entry : affiche le dictionnaire
- de la fonction addToDictionnary : ajout d'un élément au dictionnaire
- de la fonction removeFromDictionnary : suppression d'un élément au dictionnaire
- de la fonction alphaToMorseFile
- de la fonction morseToAlphaFile

4) – Réunion de travail sur le code – 10/12/2013

Corrections et ajouts de fonctions dans l'interface graphique IHM :
- Zone de saisie libre pour traduction.
- Connexion des boutons « Traduire » aux méthodes de la classe Morse.

5) – Réunion de travail sur le code – 11/12/2013

Ajouts de fonctions dans l'IHM :
- Chargement du dictionnaire avec affichage du message de bon chargement.
- Ouvrir fichier *.txt ou *.morse pour traduction automatique avec enregistrement du fichier traduit au même niveau que le fichier à traduire.

6) – Réunion de travail sur le code – 12/12/2013

- Corrections pour la détection automatique des fichiers *.txt et *.morse.
- Ajout d'une zone avec un ascenseur JscrollPane pour l'affichage du dictionnaire dans l'interface.
- Affichage de la réussite / de l'échec de la traduction, ainsi que le chemin où est enregistré le fichier traduit.
- Finalisation de l'interface graphique permettant la manipulation du dictionnaire : bouton Supprimer qui permet de supprimer une entrée du dictionnaire : la comboBox est mise-à-jour.
Bouton « Ajouter » un couple « caractère alphanumérique » - « chaîne de caractères Morse ».