

Institut Supérieur d’Informatique de

Modélisation et de leurs Applications

Campus des Cézeaux

24 avenue des Landais

BP 10125

63173 AUBIÈRE Cedex

Compte rendu de TP

Filière Génie Logiciel et Systèmes Informatiques

Auto-complétion

TP réalisé par Nicolas Prugne et Antoine Colmard

TP demandé dans le cadre du cours de compilation de M. Deleplanque

Table des matières

Table des matières

[1 Introduction 4](#_Toc382563242)

[1.1 L’auto-complétion 4](#_Toc382563243)

[1.2 Objectifs du TP 4](#_Toc382563244)

[2 Génie logiciel 4](#_Toc382563245)

[2.1 Présentation du package UML développé 4](#_Toc382563246)

[2.2 La classe Mot 5](#_Toc382563247)

[2.3 La classe AutoCompletionDatabase 6](#_Toc382563248)

[3 Algorithmique 7](#_Toc382563249)

[4 Résultats 7](#_Toc382563250)

[5 Discussion 7](#_Toc382563251)

Table des figures et illustrations

[Figure 1 - Package Auto-Completion 5](#_Toc382563252)

[Figure 2 - La classe Mot 6](#_Toc382563253)

[Figure 3 - La classe AutoCompletionDatabase 6](#_Toc382563254)

# Introduction

## L’auto-complétion

L’auto-complétion ou complément automatique est une fonctionnalité informatique qui permet à son utilisateur de saisir plus rapidement une chaine de caractères. Au fur et à mesure que l’utilisateur saisie une chaine, un algorithme va analyser les premiers caractères tapés, et proposer à l’utilisateur un ensemble de mots qui débutent de la même manière que la chaine en cours de saisie. L’utilisateur n’a plus qu’à sélectionner la chaine correspondante au mot qu’il souhaite taper. Ainsi, il limite la quantité d’informations à saisir directement au clavier ce qui lui procure un gain de temps. Pour fonctionner, l’auto-complétion a besoin d’établir un dictionnaire de mots proposables à l’utilisateur. Pour ce faire, elle peut se baser sur ce que l’utilisateur a déjà saisi, ou utiliser une liste de mots préétablis qu’elle lit dans un fichier ou dans une base de données par exemple.

## Objectifs du TP

L’objectif de ce TP est donc de coder un programme capable de gérer une auto-complétion simple en console. Ainsi, il doit pouvoir lire et charger en mémoire une liste de mots à partir d’un fichier texte passé en paramètre. Dans ce fichier, chaque mot est associé à un nombre d’occurrences. Dans un cas concret, ce nombre pourrait être calculé en fonction du nombre d’apparition d’un mot dans un texte (analysé au préalable). Cependant dans ce TP, la liste a été construite de manière empirique et non à partir de l’analyse réelle d’un texte. Après le chargement de la liste de mots, le logiciel doit proposer à l’utilisateur de saisir une sous chaine de caractères correspondant au début d’un mot. Le programme doit ensuite analyser cette sous chaine et fournir une liste de mots correspondant aux caractères qu’elle contient. Cette liste doit être affichée en fonction du nombre d’occurrences de chaque mot. Plus un mot a un nombre d’occurrences important, plus il doit apparaitre au début de la liste proposée.

Divers classes et structures de données sont nécessaires à l’implémentation d’un tel programme. La partie suivante a pour but de présenter l’analyse et la conception des classes créées afin de répondre aux spécifications du TP.

# Génie logiciel

Avant d’être implémenté, le logiciel présenté ci-dessus a nécessité une phase de réflexion et conception afin de concevoir un modèle répondant à ses spécifications. Cette conception s’appuie le paradigme de la programmation orientée objet, c’est pourquoi les composants logiciels produits au cours de ce TP s’apparentent à des classes et non à de simples structures de données. La partie suivante a pour rôle de présenter ces divers composants logiciels.

## Présentation du package UML développé

L’auto-complétion nécessite donc la gestion d’un dictionnaire de mots en mémoire. Pour ce faire, deux classes ont été développées et regroupées dans un package UML nommé Auto-Completion (Figure 1).



Figure 1 - Package Auto-Completion

Les structures de chacune des classes de ce package seront détaillées à la suite du compte rendu, cependant il est possible d’établir une analyse générale de l’architecture conçue. Tout d’abord, le package montre que le programme se base sur deux classes pour gérer son dictionnaire de mots. La première appelée AutoCompletionDatabase permet de référencer tous les mots connus par le programme. Elle fournit également une API basique afin de rechercher un des mots du dictionnaire qui débute de la même manière qu’une séquence de caractères passée en paramètre. Cette classe constitue le cœur du programme. Cependant, le package représente le fait qu’elle utilise une classe annexe afin de stocker les mots qu’elle répertorie. Cette classe baptisée Mot a pour rôle de stocker une chaine de caractères représentant ainsi un mot. Et dans le même temps, elle associe à cette chaine un entier non signé afin de compter son nombre d’occurrences.

La classe AutoCompletionDatabase est donc constituée d’un ensemble de Mot. Celle-ci présente divers fonctions et la partie suivante a pour rôle d’expliciter clairement son rôle.

## La classe Mot

La classe Mot a donc pour rôle de stocker les chaînes de caractères du dictionnaire que gère le programme tout en leur associant un entier afin de compter leur nombre d’utilisations (Figure 2).



Figure 2 - La classe Mot

Parmi ses méthodes, la classe Mot compte plusieurs constructeurs dont l’un d’entre eux est un constructeur par recopie. En effet, la classe respecte la forme de Coplien et possède donc également un opérateur d’affectation et un destructeur.

Deux opérateurs d’affectation lui ont été ajoutés. Ils permettent d’effectuer des traitements algorithmiques sur la classe, comme des tris par exemple. De plus, deux accesseurs sont disponibles afin de pouvoir consulter le contenu de la chaine stockée, ainsi que son nombre d’occurrences. Enfin, une dernière méthode permet d’incrémenter le nombre d’occurrences du mot stocké.

## La classe AutoCompletionDatabase

La classe AutoCompletionDatabase est donc au cœur du mécanisme d’auto-complétion implémenté au cours de ce TP. Pour ce faire elle stocke une liste d’objets de type Mot ainsi, qu’une chaine de caractères correspondant au nom d’un fichier. Cet attribut lui permet de lire et d’écrire dans un fichier texte les mots qu’elle contient (Figure 3).



Figure 3 - La classe AutoCompletionDatabase

La classe AutoCompletionDatabase fournit deux constructeurs. Le premier permet de construire un objet à partir d’un fichier texte dont le nom est passé en paramètre. Cela permet d’initialiser la base de mots à partir d’un fichier existant. Le second constructeur est un constructeur par défaut (sans paramètre). En effet, il est également possible de partir d’une base vide et d’ajouter au fur et à mesure de l’exécution du programme, des mots dans cette base. Cet ajout peut être fait via la méthode insertMot.

La classe possède également deux méthodes destinées à sauvegarder son contenu dans un fichier texte. Enfin, la méthode affichageMotCompleted permet de réaliser, à proprement parler, le traitement nécessaire à l’auto-complétion. Cette méthode se destine à afficher dans la console une liste de mots débutants de la même manière que le paramètre qu’on lui passe. Pour ce faire, la classe utilise la liste de mots qu’elle contient ainsi qu’un algorithme de recherche détaillé dans la partie suivante.

# Algorithmique

Afin de proposer une fonctionnalité d’auto-complétion performante, le programme se base sur un algorithme de recherche simple mais efficace.

# Résultats

# Discussion