# Rapport 1 - Projet BRAINF-CK SI3

MIAOU - Miaou Is An Open Umbrella

3 octobre 2016

## 1 Introduction

Nous vous présenterons ici notre compte rendu de la réalisation du niveau 1 ($L_1$) du projet Brainf*ck. Ce premier niveau constitue une étape cruciale et non négligeable, puisqu'elle pose ce qui servira de base à l'intégralité du projet. Nous avons donc pris soin de bien mettre en place ce socle afin de nous assurer une certaine sérénité dans la suite de l'élaboration du projet.

## 2 Choix d'implémentations

En découvrant les premières subdivisions du projet, notre premier reflexe a été de définir la structure du code que nous proposerons. L'organisation a été ainsi décidée :

La mémoire est gérée dans une classe nommée `Memory`. Cette classe possède comme attribut un tableau de byte de 30 000 cases (construit par le constructeur par défaut, on pourra par la suite (pour des raisons de tests par exemple) définir la taille de ce tableau avec un autre constructeur et ainsi réduire ou augmenter l'espace mémoire), un entier définissant l'indice du tableau sur lequel on travaille, ainsi que des méthodes pour agir sur cette mémoire à l'index enregistré. Nous avons ainsi décidé de modifier la mémoire à la manière d'un processeur, afin de s'approcher au plus possible de l'approche bas niveau du langage Brainf*ck. Une opération d'incrémentation de la mémoire, par exemple, se composera donc d'une lecture de la valeur de la case mémoire sur laquelle nous travaillons, suivie d'une incrémentation de cette valeur et, enfin, une réécriture de la case mémoire avec la valeur incrémentée.

La mémoire est attribut de la classe `Machine`, qui sert de machine virtuelle. L'utilisateur n'agissant pas directement sur la mémoire, la machine virtuelle sert donc d'interface entre lui et celle-ci. L'interaction se fait donc par le biais de commandes et d'opérations que la machine virtuelle saura exécuter pour agir sur la mémoire elle-même.

Les instructions sont des classes. Nous avons préféré cette possibilité à celle des foncteurs, par exemple, car elle permet notamment de stocker aisément (en tant qu'attributs de classe) les valeurs correspondant aux différentes possibilités d'appel de l'instruction (à savoir syntaxe complète, syntaxe allégée et couleur). Toutes les instructions sont donc dans le package `brainfuck.instructions` et héritent toutes d'une même classe mère `Instruction` définissant les attributs cités précédemment et la méthode `accept`, contenant les opérations que doit exécuter la machine virtuelle, lorsque l'appel de cette instruction est demandé. Cette même méthode est donc sur-définie dans chaque classe fille.

Le programme analyse les arguments passés à la méthode `main`, qui correspondent à ceux qui sont passés à l'exécutable `bfck` (script shell exécutant le programme java). Cette étape d'analyse s'opère avec la classe `ArgParser` qui récupère, dans notre cas, le nom du fichier passé en argument de l'option '-p' et le stocke. Ce nom de fichier est ensuite passé en paramètre du constructeur de la classe `ReadFile`. Celle-ci permet de lire le fichier ligne par ligne et, si besoin, de lire chacune de ces lignes caractère par caractère. On peut ainsi lire chacune des instructions que contient le programme Brainf*ck .

Chacune de ces instructions est ensuite lue par la classe `Interpreter`, qui saura reconnaitre chacune des commandes lue par le lecteur de fichier et demandera à la machine virtuelle de l'exécuter.

## 3 Let's Keep Calm and Take a Step Back

Avec ce premier niveau, nous avons pu entamer ce qui constitue la base du projet. En effet, si les objectifs proposés paraissaient simples au premier abord, ils sont en réalité bien plus complexes car nécessitent la mise en place d'une bonne partie des fonctionnalités de basedu programme final. Il a donc été nécessaire de réfléchir aux différents choix d'implémentations présentés afin de trouver la solution qui nous paraissait, à ce stade du developpement, optimale.

Toutefois, nous avons choisi un autre ordre d'implémentation des fonctionnalités que celui proposé. En effet, nous avons d'abord voulu implémenter la machine virtuelle et les différentes instructions possibles avant d'intégrer le lecteur de fichiers et l'analyseur d'arguments. Cette décision a été prise pour plusieurs raisons. La principale fut que notre première approche du problème ne nécessitait pas ces deux derniers points. En revanche, la mise en place des instructions était essentielle à son implémentation. Notre priorité pour ce début de projet ayant réellement été de mettre en place cette structure de code à laquelle nous avions réfléchi, plus encore que de répondre à une ou des fonctionnalités demandées, nous avons plutôt priorisé la mise en place de cette structure de base. De plus, cela nous a paru être une idée plus maintenable et plus « facile » à mettre en place car nous pouvions ainsi tester directement les différentes méthodes et instructions en ne faisant que modifier la méthode `main`. Et en effet, il a été beaucoup plus simple de travailler directement avec la machine virtuelle dans un premier temps, sans passer par le lecteur de fichiers et l'intepréteur.