MGL7460 : Laboratoire #2
Développement d'une application en ligne de commandes avec `gli` et son DSL

22 septembre 2016

Le but de ce laboratoire est de vous familiariser avec l'utilisation de Ruby pour le développement d'un *gem*, notamment, vous familiariser avec l'organisation typique d'un *gem* Ruby — fichiers `Rakefile`, `*.gemspec`, `Gemfile`, répertoires `bin`, `lib`, `test`, etc. Un objectif additionnel est de vous familiariser avec l'utilisation d'un DSL Ruby, plus spécifiquement, avec le DSL du *gem* `gli`, lequel permet de décrire des applications en ligne de commandes avec leurs options, arguments et commandes.

Pour obtenir le code

a. Connectez vous au serveur `malt.labunix.uqam.ca`.

b. Une fois dans votre compte sur `malt`, exécutez les commandes suivantes — ici, le caractères «$» indique l'invite (le *prompt*) du *shell* Unix, donc **vous ne devez pas taper ce caractère** :

```
$ git clone http://www.labunix.uqam.ca/~tremblay/git/mini-sed.git

$ cd mini-sed

$ bundle install --path vendor/bundle

$ rake
```

Remarque : L'installation des *gems* lors de l'exécution de la commande «`bundle install...`» peut prendre plusieurs minutes!

Ce qui vous est fourni

- `bin/mini-sed` : Le programme principal, qui utilise `gli` et son DSL pour définir une application en ligne de commandes qui **émule** (une version simplifiée de) `sed`. **Ce fichier est un de ceux que vous devez compléter/modifier.**
- `lib` : Répertoire contenant divers fichiers utilisés pour la mise en oeuvre des fonctionnalités de l'application (modèle, et non interface personne–machine). **Le fichier principal, que vous devez compléter/modifier, est** `lib/mini-sed/mini-sed.rb`**.**
- `test` : Répertoire contenant divers tests **unitaires** — décrits avec `MiniTest`.
- `test_acceptation` : Répertoire contenant divers tests **d'acceptation** — décrits eux aussi avec `MiniTest`.
- `Rakefile` : Fichier qui permet d'automatiser l'exécution des tests. Les principales cibles pour ce labo — avec la commande «`rake`» — sont les suivantes :
  - `exercice_a`
  - `exercice_b`
  - `exercice_c`
  - `exercice_d`

  Voir plus bas pour la description de chacun des exercices.

Suggestion : Comme pour le laboratoire précédent, je vous suggère d'avoir (au moins) deux fenêtres, toujours ouvertes, et d'alterner entre ces deux fenêtre (édition, tests, etc.) :

a. Une fenêtre où vous faites l'édition du fichier à modifier (fichier `bin/mini-sed` ou fichier `lib/mini-sed/mini-sed.rb`) ;

b. Une fenêtre où vous lancez l'exécution de tests avec la commande `rake`.

Donc, évitez de lancer/fermer votre éditeur de texte de façon répétitive.

Ce que vous devez faire

a. Complétez la mise en oeuvre de la méthode `MiniSed.delete`.

   Pour ce faire, vous devez **compléter le «*squelette*» de méthode déjà définie** dans le fichier `lib/mini-sed/mini-sed.rb`.

   Cible pour les **tests unitaires** : `rake exercice_a`

   Indice : Style fonctionnel avec `Enumerable#reject`.

b. Complétez la mise en oeuvre de la méthode `MiniSed.print`, toujours dans le fichier `lib/mini-sed/mini-sed.rb`.

   Cible pour les **tests unitaires** (défaut du `Rakefile`) : `rake exercice_b`

   Indice : Ici, il faut utiliser le style impératif avec `each`... car le résultat produit peut être un **sur-ensemble** — i.e., il peut y avoir plus de lignes en sortie qu'en entrée!

   Suggestion : Pour simplifier la lancement des tests pour cet exercice, modifiez la tâche `:default` dans le `Rakefile`, en remplaçant `exercice_a` par `exercice_b`.

c. Complétez la description de la commande `delete`, partiellement définie dans le fichier `bin/mini-sed`.

   Ici, l'objectif est que les **tests d'acceptation** associés à cette commande (fichier `test_acceptation/delete_test.rb`) réussissent.

   Cible pour les tests d'acceptation (défaut du `Rakefile`) : `rake exercice_c`

d. Donnez une mise en oeuvre pour une méthode `MiniSed.insert` (`lib/mini-sed/mini-sed.rb`) et la description de cette commande (fichier `bin/mini-sed`).

   Le comportement de cette commande est le suivant :

/patron/`i\`*chaine*	Insère une nouvelle ligne contenant *chaine* **devant** la ligne courant si elle matche *patron*

   Cible pour les tests, **unitaires et d'acceptation** : `rake exercice_d`