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

Extraits du fichier `lib/mini-sed/mini-sed.rb`

```ruby
def self.delete( motif, lignes )
  lignes
    .reject { |ligne| /#{motif}/ =~ ligne }
end

def self.print( motif, lignes )
  # Doit utiliser each: plusieurs repetitions possibles d'une ligne!
  res = []
  lignes.each do |ligne|
    res << ligne if /#{motif}/ =~ ligne
    res << ligne
  end

  res
end

def self.insert( motif, chaine_a_ajouter, lignes )
  # Doit utiliser each car plus de lignes en sortie qu'en entree.
  res = []
  lignes.each do |ligne|
    res << (chaine_a_ajouter + "\n") if /#{motif}/ =~ ligne
    res << ligne
  end

  res
end
```

Extraits du fichier `bin/mini-sed` après *refactoring*!

```
desc 'Imprime les lignes contenant un motif'
arg_name 'motif [fichier...]'
command :print do |c|
  c.action do |global_options,options,args|
    motif, *fichiers = args
    fichiers << STDIN if fichiers.empty?

    fichiers.each do |fichier|
      MiniSed.traiter_fichier( fichier, global_options[:in_place] ) do |flux|
        MiniSed.print( motif, flux.readlines )
      end
    end
  end
end

desc 'Insertion d\'une ligne devant une qui matche'
arg_name 'motif chaine_a_inserer [fichier...]'
command :insert do |c|
  c.action do |global_options,options,args|
    motif, chaine_a_inserer, *fichiers = args
    fichiers << STDIN if fichiers.empty?

    fichiers.each do |fichier|
      MiniSed.traiter_fichier( fichier, global_options[:in_place] ) do |flux|
        MiniSed.insert( motif, chaine_a_inserer, flux.readlines )
      end
    end
  end
end

desc 'Supprime les lignes contenant un motif'
arg_name 'motif [fichier...]'
command :delete do |c|
  c.action do |global_options,options,args|
    motif, *fichiers = args
    fichiers << STDIN if fichiers.empty?

    fichiers.each do |fichier|
      MiniSed.traiter_fichier( fichier, global_options[:in_place] ) do |flux|
        MiniSed.delete( motif, flux.readlines )
      end
    end
  end
end
```

Extraits du fichier `bin/mini-sed` après un autre *refactoring*! pour rendre le code plus *DRY*

```
def commande_sans_option commande, nb_arguments: 1
  command commande do |c|
    c.action do |global_options,options,args|
      les_args, fichiers = args[0...nb_arguments], args[nb_arguments..-1]
      fichiers << STDIN if fichiers.empty?

      fichiers.each do |fichier|
        MiniSed.traiter_fichier( fichier, global_options[:in_place] ) do |flux|
          MiniSed.send commande, *les_args, flux.readlines
        end
      end
    end
  end
end

desc 'Imprime les lignes contenant un motif'
arg_name 'motif [fichier...]'
commande_sans_option :print, nb_arguments: 1

desc 'Insertion d\'une ligne devant une qui matche'
arg_name 'motif chaine_a_inserer [fichier...]'
commande_sans_option :insert, nb_arguments: 2

desc 'Supprime les lignes contenant un motif'
arg_name 'motif [fichier...]'
commande_sans_option :delete, nb_arguments: 1
```