INF5171: Laboratoire #1
Utilisation de `ruby` sur `japet` et mise en oeuvre d'une classe `Ensemble`

14 septembre 2017
13h30–15h30
PK-S1565

> *Learn to use* `Enumerable`*. You will not be a rubyist until you do.*
>
> *«Ruby QuickRef», R. Davis (*`http://www.zenspider.com/Languages/Ruby/QuickRef.html`*)*

Le but de ce laboratoire est de vous familiariser avec l'écriture et l'exécution de programmes Ruby — le langage que nous utiliserons dans la première partie du cours — ainsi qu'avec l'exécution de tests unitaires `MiniTest`. Le but est aussi de vous familiariser avec l'utilisation des méthodes du module `Enumerable` de Ruby — la partie de Ruby qui favorise et supporte l'utilisation du style **fonctionnel** de programmation.

Finalement, bien que ce premier labo ne traite pas de programmation parallèle, le but est aussi de vous familiariser avec l'environnement Linux de la machine **parallèle** que nous utiliserons pour le cours, `japet.labunix.uqam.ca`, une machine multiprocesseurs à soixante-quatre (64) coeurs/processeurs

---

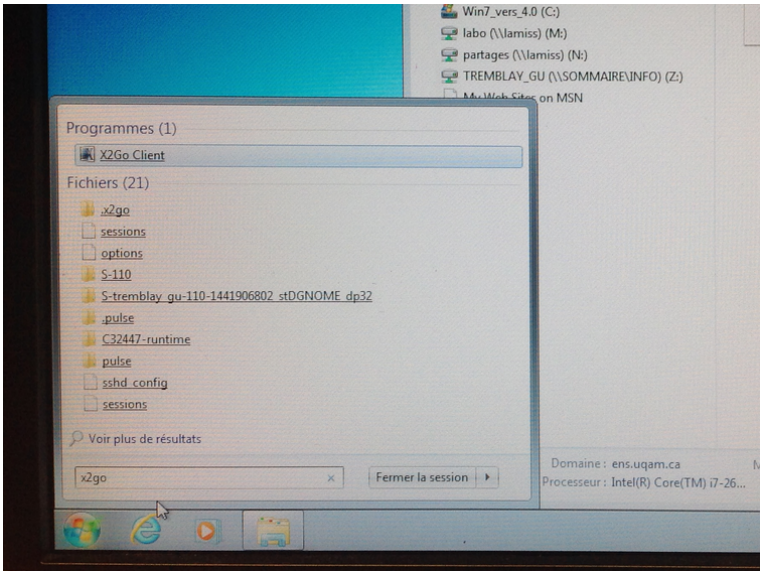
1 Comment se connecter à `japet`

a. Connectez-vous tout d'abord à un poste Windows.

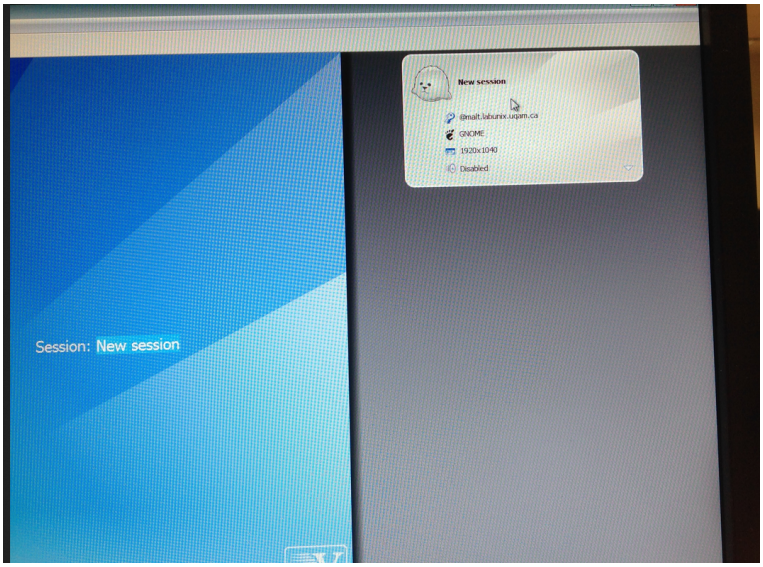
   Remarque importante : La **première fois** où vous allez vous logger sur un poste, cela pourrait prendre plusieurs (!) minutes avant que vous soyez connecté — la nouvelle configuration des postes pour votre identifiant doit tout d'abord être chargée et c'est un peu long (mais uniquement la première fois) ☹ Donc, soyez patient!

   Pour ce faire, vous devez utiliser votre nom d'usager (NIP) et mot de passe standards UQAM.

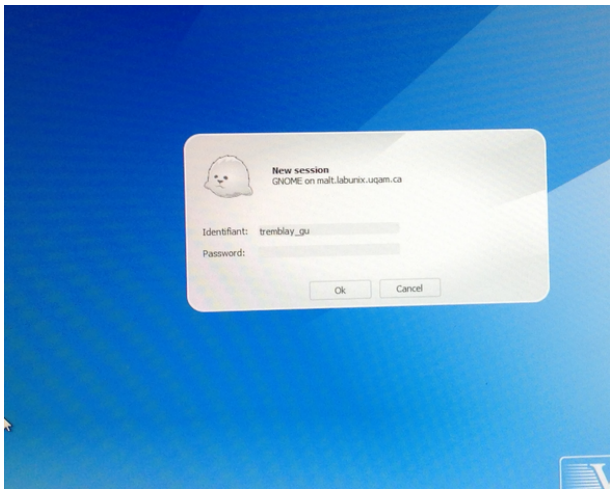
b. Une fois la session Windows ouverte, lancez l'application `X2Go Client` — voir en bas à gauche :



c. Dans l'application `x2go`, lancez une session sur `malt.labunix.uqam.ca` — voir en haut à droite dans la figure ci-bas.



d. Connectez-vous au serveur `malt.labunix.uqam.ca` en utilisant votre nom usager et mot de passe habituels — les mêmes que pour la machine Windows :



e. Une fois sur `malt`, vous pouvez alors vous connecter à `japet.labunix.uqam.ca`, et ce via la ligne de commande. Pour ce faire, ouvrez un terminal (clic droit avec la souris puis `Open in Terminal`) et, dans ce terminal, lancez la commande suivante :

```
$ ssh japet.labunix.uqam.ca
```

Quelques remarques et une suggestion :

- Lorsque vous êtes sur `japet`, votre répertoire de fichiers est **exactement le même que sur** `malt` ou sur n'importe quelle autre machine Linux du réseau `labunix`.
- Si vous ouvrez une fenêtre sur `japet` — peu importe avec quelle application — la fenêtre s'ouvrira sur votre poste de travail.
- **Important :** Pour fermer la connexion à `malt` créée avec `x2go`, il faut fermer **en cliquant sur le petit** **X** **de fermeture dans le coin droit en haut!**
- Si vous préférez, vous pouvez aussi travailler à partir de votre ordinateur portable, via une connexion `ssh` standard.
- Je vous suggère d'avoir au moins deux (2) fenêtres, toujours ouvertes, et d'alterner d'une fenêtre à l'autre (édition, tests) :
  a. Une première fenêtre où vous faites l'édition du fichier `ensemble-*.rb` ;
  b. Une deuxième fenêtre, qui doit absolument être ouverte sur `japet`, où vous lancez l'exécution des tests avec une commande `make` appropriée.
- Pour une introduction rapide à Unix (ou un bref rappel), consultez la page suivante :
  `http://www.labunix.uqam.ca/~tremblay/INF5171/Labos/intro-unix.txt`
  Tel qu'indiqué dans ce document, un éditeur graphique simple à utiliser est `gedit`.

2 Obtenir le code à compléter

Obtenez une copie des fichiers pour le labo en exécutant la commande suivante sur `japet` (ligne de commande) :

```
$ git clone http://www.labunix.uqam.ca/~tremblay/git/Ensemble.git
```

3 Ce que vous devez faire

Pour ce laboratoire, vous devez **compléter** diverses méthodes de la classe `Ensemble`, classe définie avec ses méthodes dans les fichiers `ensemble-imperatif.rb` et `ensemble-enumerable.rb` :

- Quelques opérations de base :
  a. `un_element`
  b. `max`
  c. `union`
  d. `intersection`
  e. `<=`
  f. `to_a`
- Diverses opérations liées à la sélection d'éléments :
  e. `select_lambda`
  f. `select`
  g. `partitionner`
  h. `trier`

La documentation pour cette classe (format YARD, semblable à JavaDoc) est disponible à l'adresse suivante :
`http://www.labunix.uqam.ca/~tremblay/INF5171/Labos/Ensemble-doc/EnsembleImperatif.html`

C'est sur cette page, ainsi que dans le fichier `ensemble-imperatif.rb` lui-même, que vous trouverez la spécification et description des opérations que vous devez compléter, et ce à l'aide de commentaires et de petits exemples. Évidemment, c'est aussi **en regardant les cas de tests appropriés** — dans les fichiers `ensemble_*_spec.rb` — que vous pourrez mieux comprendre le comportement attendu.

Informations additionnelles

- Vous devrez essayer (**si le temps le permet!?**) de réaliser deux (2) mises en oeuvre différentes des ensembles :

  a. `ensemble-imperatif.rb` : Mise en oeuvre **impérative** «classique» — donc avec du code procédural impératif, des boucles explicites, etc.

  b. `ensemble-enumerable.rb` : Mise en oeuvre dans un style fonctionnel, où la classe `Ensemble` inclut le module `Enumerable` et où les méthodes utilisent, le plus possible, les méthodes **fonctionnelles** du module `Enumerable`.

  Je vous suggère toutefois de débuter avec la mise en oeuvre **impérative**, style avec lequel vous êtes certainement plus familiers.

- Divers fichiers vous sont fournis :

  - `ensemble.rb` : Le fichier de haut niveau, qui sélectionne la mise en oeuvre désirée.
  - `ensemble-imperatif.rb` : Une mise en oeuvre, partielle, de la classe `Ensemble` avec du code impératif et procédural.
  - `ensemble-enumerable.rb` : Une mise en oeuvre, partielle, de la classe `Ensemble` avec du code utilisant le module `Enumerable` et le style fonctionnel.
  - `ensemble_base_spec.rb` et `ensemble_select_spec.rb` : Deux programmes de tests, définis avec `MiniTest` (dans le style RSpec, donc avec des «*expectations*»), respectivement pour les opérations de base et pour les opérations de sélection.
  - `makefile` : Un fichier pour automatiser le lancement des tests :
    * `$ make` : Lance les tests identifiés par la variable `TEST` — première ligne du fichier.
      Par défaut : `TEST=un_element`, donc la première méthode à compléter.
      Par défaut : c'est la mise en oeuvre impérative (procédurale) qui est testée.
      Voir plus bas pour tester la mise en oeuvre avec `Enumerable`.
      Pour tester une seule méthode, autre que `un_element`, modifiez la variable `TEST` dans le fichier `makefile`.
    * `$ make bases` : Lance les tests pour les toutes opérations de base.
    * `$ make selects` : Lance les tests pour les toutes opérations de sélection.
    * `$ make tests` : Lance tous les tests, par défaut pour la mise en oeuvre impérative.
    * `$ make tests_enumerable` : Lance tous les tests, mais pour la mise en oeuvre avec `Enumerable`.