Installation des outils pour 3I020

Pour commencer, créez un répertoire spécifique pour l'UE 3I020.

Etape 1 : configuration du shell

Votre shell Unix doit être configuré pour l'UE 3I020, avec au moins les définitions suivantes:

(fichier `~/.profile`)

```
# ... etc ...

# configuration du chemin
export PATH=$PATH:/Infos/lmd/2018/licence/ue/3I024-2019fev/bin

# configuration du proxy http/https
export http_proxy=proxy:3128
export https_proxy=$http_proxy  # <-- attention celui-ci n'existe pas par défaut

# ... etc ...
```

Remarque : pour la première utilisation il faudra directement saisir ces définitions dans le terminal (ou alors se reloger).

Etape 2 : java

Nous avons besoin de du JDK java 1.8 (ou ultérieur). Pour vérifier la version, utilisez le terminal.

Par exemple :

```
$ java -version
java version "1.8.0_171"
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)
```

Etape 3 : installation de leinigen

Leiningen est un gestionnaire de projet pour Clojure, qui permet de facilement récupérer des bibliothèque tierces Clojure ou Java (écosystème clojars/maven).

Le script `lein` (pour Unix/Linux ou OSx) peut être downloadé à l'adresse suivante: https://leiningen.org (il existe également un installateur pour windows, cf. https://djpowell.github.io/leiningen-win-installer)

Le programme `lein` est déjà installé sur les machines de la PTTI.

Pour la première utilisation, saisissez la commande suivante :

(répondez par `Y` à la question)

```
$ lein upgrade
The script at ./lein will be upgraded to the latest stable version.
Do you want to continue [Y/n]? y

Upgrading...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   137  100   137    0     0    564      0 --:--:-- --:--:-- --:--:--   566
100 12538  100 12538    0     0  29244      0 --:--:-- --:--:-- --:--:-- 29244
Leiningen is already up-to-date.
Leiningen 2.8.3 on Java 1.8.0_171 Java HotSpot(TM) 64-Bit Server VM
```

Etape 4 : récupération du squelette de TME

Les squelettes de TME seront distribués sous forme de projets clojure/leiningen sur un serveur git.

Les projets disponibles sont visibles sur la page web: https://gitlab.com/progdecl

Pour cloner le premier TME, la commande est la suivante:

(à saisir dans votre répertoire pour l'UE 3I020)

```
$ git clone https://gitlab.com/progdecl/theme01-intro-exos
Clonage dans 'theme01-intro-exos'...
remote: Counting objects: 12, done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 12 (delta 0), reused 0 (delta 0)
Dépaquetage des objets: 100% (12/12), fait.
Vérification de la connectivité... fait.
```

Dans le répertoire `theme01-intro-exos/` vous trouverez:

- un fichier `README.md` explicatif
- un fichier de construction `project.clj` pour lein
- un répertoire source `src/theme01_clojure/` avec:
- un fichier source `ex01_bases.clj` énoncé de l'exercice 1
- ... etc ...

Etape 5 : démarrage de nightlight

L'éditeur que nous allons utiliser (par défaut) s'appelle nightlight, et à l'avantage d'avoir un support spécifique Clojure (il est développé en Clojurescript) et d'être très simple à utiliser (mais limité en fonctionnalités).

Après s'est déplacé dans le répertoire du thème 1, lancez la commande suivante:

```
$ cd theme01-intro-exos
$ lein nightlight
Started Nightlight on http://localhost:4000
```

Ouvrez ensuite dans votre navigateur un onglet à l'adresse indiquée: http://localhost:4000

Vous pouvez activer l'*instalREPL* qui effectue des auto-évaluations systématiques.

Etape 6 : Documentation en ligne

Pendant vos développements en Clojure, nous vous conseillons d'ouvrir votre navigateur sur la page suivante et de la garder à proximité:

https://jafingerhut.github.io/cheatsheet/grimoire/cheatsheet-tiptip-cdocs-summary.html

Etape 7 : Tests

Les tests représentent une partie importante de tout développement logiciel. Pour l'UE nous avons choisi `midje` une boîte à outils pour le TDD (test-driven development) en Clojure (il existe également un support natif mais moins complet que `midje`).

Dans un terminal, vous pouvez lancer les tests avec la commande :

```
$ lein midje

FAIL "pour l'accès aux informations des vecteurs" at (ex04_collections.clj:32)
    Expected: true
      Actual: false

FAIL "pour les mises-à-jour des vecteurs" at (ex04_collections.clj:75)
    Expected: true
      Actual: false

... etc ...
```

Il existe également un mode `autotest` très pratique, qui permet de relancer automatiquement les tests après chaque modification de fichier.

```
$ lein midje :autotest

================================================================================
Loading (theme01-clojure.ex02-fact-fib theme01-clojure.ex04-collections /
  theme01-clojure.ex05-recur theme01-clojure.ex06-mastermind /
  theme01-clojure.ex01-bases theme01-clojure.ex03-phrases)

FAIL "pour l'accès aux informations des vecteurs" at (ex04_collections.clj:32)
    Expected: true
      Actual: false

... etc ...

FAILURE: 11 checks failed.  (But 60 succeeded.)
[Completed at 10:01:44]
```

Le mode `autotest` ne fonctionne que s'il n'y a pas d'erreur de compilation. Pour sortir de ce mode, il suffit de taper `Ctrl-c` dans le terminal.

Remarque: Pour pouvoir démarrer le thème $n + 1$ vous devez avoir 100% des tests qui passent pour le thème n.