TD n°2 *Développement Dirigé par les Tests*

Dans ce TD, vous allez développer, avec `playframework`, un convertisseur en ligne pour traduire des entiers entre leur représentation en décimal et leur représentation en chiffres romains.

Le véritable but de cette session est de s'initier à la méthode agile appelée *Test-Driven Development* ou TDD, c'est à dire le développement dirigé par les tests. On utilisera également une autre méthode agile appelée *Pair Programming* ou l'on travaille en binôme. Mais, pour commencer, le responsable de TD animera ce qu'on appelle une session de *Randori*.

Exercice 1. Github

Pour vous aider à démarrer cette application, un squelette déjà fonctionnel est disponible sur github.

- Vous devez vous créer un compte sur github su vous n'en avez pas déjà un
- Loggez vous sur votre compte github
- Visitez le projet `tdd-chiffres-romains` à l'url :
  `https://github.com/denys-duchier/tdd-chiffres-romains`
- Cliquez `Fork` pour cloner ce projet sur votre compte github

Vous pouvez maintenant obtenir ce code sur votre compte à l'IUT. Placez-vous par exemple dans votre home :

```
cd
```

puis clonez votre copie du projet :

```
git clone https://....
```

où l'URL est celui donné dans la page web de votre copie du projet (la boite intitulée "`HTTPS clone URL`" dans la marge droite de la page). Entrez dans le projet et observez :

```
cd tdd-chiffres-romains
git remote -v
git branch -av
```

Exercice 2. Mise en place des binômes

Puisque vous allez travailler en *pair programming*, il va falloir mettre en place la possibilité de faire des merges entre vos dépôts respectifs.

- Choisissez un binôme
- Je désignerai par A et B les deux membres du binôme

A travaille / B regarde

A ajoute son nom au fichier `README`, puis fait :

```
git commit -a -m "ajout de A au README"
git push
```

A regarde / B travaille

B veut pouvoir récupérer les mises à jours de A. Pour cela il doit indiquer à git où se trouve le dépôt de A. Cela se fait en ajoutant un *remote* :

```
git remote add binome https://...
git remote -v
```

où l'url est celui du dépôt de A. Il faut maintenant télécharger l'historique de A :

```
git fetch binome
```

Observez :

```
git branch -av
```

Vous avez maintenant des *remote tracking branches* pour la/les branches du dépôt de A. Mais votre branche `master` n'a pas encore été modifiée : il reste à faire le merge :

```
git merge binome/master
git branch -av
git log
```

Vous ajoutez votre nom au `README`, puis :

```
git commit -a -m "ajout de B au README"
git push
```

A travaille / B regarde

A veut pouvoir récupérer les mises à jours de B. Donc à son tour :

```
git remote add binome https://...
git remote -v
```

où l'url est celui du dépôt de B.

```
git fetch binome
```

Observez :

```
git branch -av
```

Puis faites le merge :

```
git merge binome/master
git branch -av
git log
```

Exercice 3. Vérification du fonctionnement

Vérifiez que l'application se lance :

```
play run
```

et qu'elle est accessible sur `http://localhost:9000`.

Vérifiez également que le framework de test est en place :

```
play test
```

Exercice 4. Randori

Le responsable de TD va animer une session de Randori, c'est à dire où 2 étudiants sont au poste video-projeté : l'un observe et conseille, l'autre programme et décrit à haute voix ce qu'il fait. Après 1 ou 2 itération, l'étudiant programmeur retourne à sa place, l'observateur le remplace au clavier et un autre étudiant prend la place de l'observateur.

Exercice 5. Pair Programming

Après un amorçage suffisant en Randori, les binômes continuerons par eux-même : l'un joue le rôle d'observateur, l'autre de programmeur ; puis, après 1 ou 2 itérations, ils échangent leurs rôles, mais gardent leurs places respectives : les commits de l'un sont publiés par `push` puis récupérés par l'autre par `pull`.