CPOA - TP

Code initial pour le TP2

Rappel du cours : 📱📱💻 http://bit.ly/jmb-cpoa

Informations générales

NOM

BRUEL

Prénom

Jean-Michel

Groupe #

- [x] Enseignants
- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] Innopolis

Pré-requis

Il vous faut :

- [x] Un compte GitHub
- [ ] Un terminal de type Git Bash (si vous utilisez Window$)

Essayez la commande suivante dans votre terminal pour vérifier votre environnement `git` :

```
git config --global -l
```

Tâche initiale

- [x] Clickez sur le lien Github Classroom fourni par votre enseignant (en fait c'est déjà fait si vous lisez ces lignes).
- [ ] Clonez sur votre machine le projet Github généré pour vous par Github Classroom.

- [ ] Modifiez le `README` pour modifier Nom, Prénom et Groupe.
- [ ] Commit & push:

commit/push

```
fix #0 Initial task done
```

Dans la suite de ce document, à chaque fois que vous trouverez un énoncé commençant par `fix #···` vous devez vérifier que vos scripts/fichiers modifiés sont bien dans votre dépôt local en vue de committer et de pusher les modifications sur votre dépôt distant en utilisant comme message de commit cet énoncé.

- Si vous voulez vérifier que vous êtes prêt pour le `fix #0`, utilisez la commande : `make check`.
- Si vous voulez avoir la liste des ToDos de ce TP/TP, exécutez `make todos`.

Les exercices de ce TD sont tirés de l'excellent livre "Tête la première : Design Pattern". Bert Bates, Eric Freeman, Elisabeth Freeman, Kathy Sierra. Editions O'Reilly. 2005.

1. Les tests Cucumber

Vous allez vous attarder cette semaine sur les tests Cucumber.

1.1. Reprise des codes du TD2

TODO:

- ☐ Reprenez les codes java du TD2 sur la fabrique de chocolat
- ☐ Vérifiez à l'aide du `pom.xml` de ce dépôt que vos tests actuels passent et que votre environnement de travail est opérationnel.

commit/push

```
fix #1.1 I am ready!
```

1.2. Tutoriel Cucumber

- Si ce n'est déjà fait, installez les plugins infinitest et Cucumber

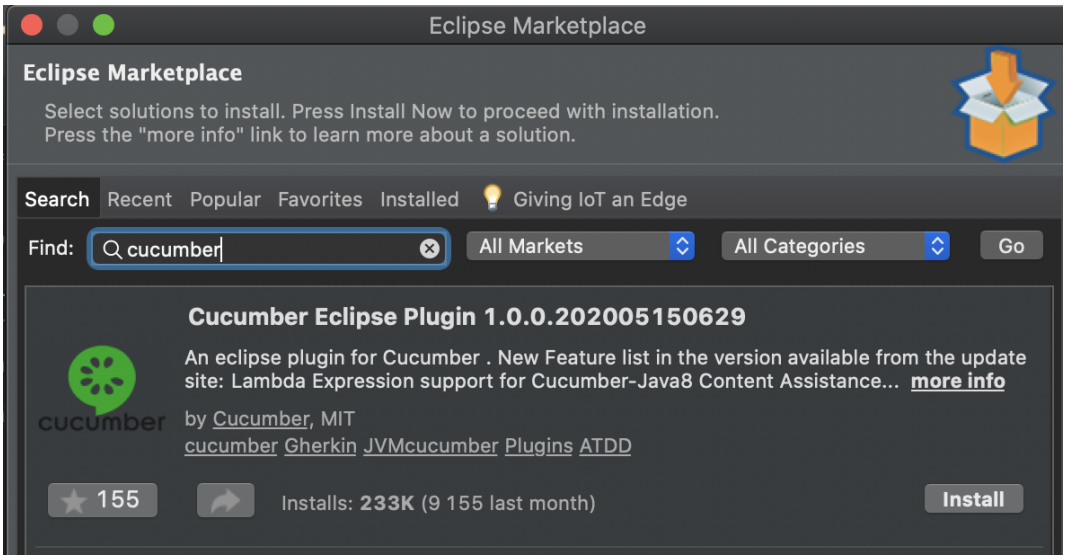

Figure 1. Exemple du plugin Cucumber sous eclipse

TODO:

- ☐ Suivez le tutoriel suivant en l'appliquant à votre code : https://cucumber.io/docs/guides/10-minute-tutorial/. Vous devez, à l'issue de ce tuto, avoir :
- ☐ Un `pom.xml` ou un `build.gradle` pour lancer les tests
- ☐ Ajouté un fichier de *feature* dans `src/test/resources`, par exemple :

Safety.feature exemple de feature Cucumber

```
#Author: JMB
Feature: Safe Chocolate Factory

  As a controller, I want to garanty that I am the only one
controlling my physical Boiler so that the Boiler cannot get
contradictory orders.

  Scenario: Trying to create 2 controllers
  Given a controller A and a controller B
  When  A is ordered to fill and boil
  Then  B cannot boil
```

- ☐ Utilisé le lanceur de tests dans `src/test/java` :

```
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(plugin = {"pretty"})
public class RunCucumberTest {

}
```

- ☐ Implémenté le code de test des *steps* (méthodes java) (dans `src/test/java`)
- ☐ Quand c'est fini :

commit/push

```
fix #1.2 Cucumber is working!
```

Appendix A: Pour Aller plus loin...

QUESTION

1. Essayez maintenant de configurer d'autres tests, notamment en utilisant des exemples pour générer plus de scénarios catastrophes qui ne peuvent se produire.
2. Si vous avez utilisé `mvn`, tentez l'expérience de `gradle` et vice-versa.

commit/push

```
fix #Bonus: Here is additional material...
```

Contributeurs

- Jean-Michel Bruel
- Louis Chanouha

À propos...