# Pratique d’assurance qualité

# Préparation

* Installez Node.js
  + <https://nodejs.org/>
* Installez Mocha (un cadriciel de développement en TDD pour javascript)
  + <https://mochajs.org/#installation>
* Naviguez au dossier de template et exécutez : *mocha tests.js*

# Ce qu’on pratique

## Le processus de développement piloté par les tests (TDD)

* **Rouge** : Comprendre les requis de l’utilisateur et écrire un test vérifiant ce à quoi on s’attend comme résultat. Ce test ne devrait pas passer initialement puisque le code n’a pas encore été écrit.
* **Vert** : Écrivez seulement le code nécessaire pour que le test passe, en vous assurant que tous les tests précédents passent encore (Il ne faut pas créer de régressions)
* **Refactoriser** le code qui a été écrit. Nettoyez le code afin de le rendre plus facile à comprendre et maintenir.

## Programmation en paire

* Un programmeur « conduit » tandis que le second « navigue ». Le premier opère le clavier tandis que le second observe, apprend, pose des questions et fait des suggestions.
* Il faut s’assurer de **parler beaucoup**.
* Après une heure de travail, on échange de rôle.

## Développement incrémental

* On se concentre sur un seul petit « incrément » à la fois.
* Chaque incrément suivant doit garder le produit utilisable.

## Git

* add, commit, push, checkout, merge, etc.
* Il faut s’assurer de faire de petits commits concernant de petits « incréments » isolés.

# Exercice

* Considérer chaque item des requis fonctionnels comme une itération (incrément). Dans chaque itération, écrivez vos tests et ensuite le code pour que les tests passent.
* Faites un commit de votre code après chaque itération (chaque fois qu’un test passe).
* Rappelez vous des règles d’or du code de qualité :
  + Écrire avec un bon style cohérent. Utilisez des noms clairs et expressifs.
  + Écrire de petites fonctions et classes n’ayant qu’une seule responsabilité.
  + Ne répétez pas de code.
  + Trouvez les solutions les plus simples possibles
  + Etc.
* Rappelez vous de refactoriser (si nécessaire) votre code après chaque test.

## Travail à réaliser

1. Créer une simple calculatrice par *string* avec une méthode *int add(numberString)*
   1. Pour un *string* vide, la méthode retourne 0.
      1. Add(“”) retourne 0.
   2. Pour un seul nombre, la méthode retourne la valeur du nombre
      1. Add(“2”) retourne 2. Add(“14”) retourn 14. Etc.
2. Permettre à la méthode Add de prendre un string ayant jusqu’à 2 nombres (séparés par ‘,’) et retourner leur somme.
   1. Add(“2,3”) retourne 5. Add(“12,5”) retourne 17. Etc.
3. Permettre à la méthode de traiter un nombre arbitraire de valeur.
   1. Add(“2,3,5”) retourne 10. Add(“11,10,2,3,5,1,3”) retourne 35. Etc.
4. Permettre à la méthode de trainer des sauts de lignes au lieu de virgules entre les chiffres.
   1. Le string suivant est valide : “1\n2,3” retourne 6
   2. Celui-ci est invalide : “1,\n”
5. Permettre de supporter different séparateurs
   1. Pour changer le séparateur, le string doit commencer par une ligne ayant le format suivant: “//[séparateur]\n[nombres]”
   2. Par exemple: “//;\n1;2” devrait retourner 3 en utilisant le séparateur ‘;’.
   3. La première ligne est optionelle, les scenarios précédents doivent encore être supportés.