Testing, testing, testing avec

Développements Pilotés par les Tests (TDD)

OBJECTIFS

- Rédaction de tests qui échouent (ROUGE)
- Écrire du code qui passe ces tests (VERT)
- Ecrire du code plus propre qui ne casse pas ces tests (Refactoring)
- Continuer la pratique de Git/GitHub

En utilisant SEG3503_PLAYGROUND

- Créer le repertoire **/lab04**
- Unzip **tic.zip** et **fizzbuzz.zip**
- Assurez-vous que vous pouvez
  - ✓ Compiler le code
  - ✓ Exécuter des tests

IGNOREZ MAINTENANT CES PROJETS :-)

- Creer tic_java OU fizzbuzz_java
- Ecrire un test qui échoue
- Faire échouer (commit ce changement)
- Faire passer (commit ce changement)
- Améliorez-le si possible (et commit ce changement)

TEST DRIVE Pour Une Heure

- Une fois que vous avez votre premier test d'échec, démarrez l'horloge
- Rouge - Vert - Refactoriser pendant une heure
- Si vous rencontrez un défi technique arrêtez le chrono
- Si vous êtes bloqué par un problème Java (ou JUnit), arrêtez l'horloge
- Chaque progression doit être capturée en tant que commit (cela inclut l'ajout d'un nouveau test d'échec ROUGE)

SOUMISSION

- Tous les travaux doivent être dans **seg3503_playground/lab04**
- Le message de commit Git doit inclure les résultats des tests JUnit
- Créer **README.md** pour résumer votre travail
    - Identifiez au moins 5 groupes de commits (par personne) pour mettre en évidence des exemples de
        - Un test d'echec
        - Un test de réussite
        - Code refactorisé (où le test réussit toujours)
    - Pour clarifier, si vous travaillez en équipe, vous avez besoin de 10 groupes de commit au total
        - 5 par coéquipier
        - Identifiez-vous clairement sinon vous serez pénalisé
- Partagez votre répo avec l'enseignant et les AEs
    - Les soumissions à BrightSpace doivent clairement référencer votre référentiel GitHub