Tests et Legacy

But

● Apprendre à couvrir du legacy avec tests
● Découvrir les merveilles du remaniement
    ● Simplification
    ● Expression du métier dans le code
● Savoir “écouter” les tests

Instructions

code :https://github.com/marwensaid/legacy-code-tests

Couvrez complètement avec tests
On évite comme la peste de modifier du code non testé.
Si on doit toucher le code pour le rendre testable on le fait avec des remaniements automatisés de l’IDE!

- Écrivez un test à la fois
- Commencez avec les données les plus simples (aussi les branches les moins
- profondes)

C’est terminé quand 100% du code est couvert. Sauf évidemment les instructions non testables unitairement : UserSession.getInstance().getLoggedUser() et TripDAO.findTripsByUser(user).

Remaniez la classe TripService

Et d’autres classes comme vous sentez le besoin.

Astuces de remaniement
● Commencer dans les branches les plus profondes – il y a moins de dépendances et c’est plus facile à appréhender.
● Rapprocher la déclaration des variables de leur utilisation – minimise le nombre de dépendances
● Lorsqu’un bout de code ne parle un un autre objet et uniquement à cet objet, déplacez ce code dans l’autre objet.
● Extraire des méthodes et variables descriptives
● Faire validations en premier

Astuce : Décrivez à haute voix en langage naturel (sans mot technique) les règles métier. Ecrivez ces phrases dans l’IDE. Transformez ceci en code le plus fidèlement possible!

Traitez l’odeur de test “partial mocking”

Maintenant le code contient probablement deux méthodes protected. Nous ne testons pas la classe TripService dans sa totalité. Ceci est une odeur de test qui pointe sur des problèmes de design du code de production.
Quels sont ces problèmes?
Réglez ces problèmes dans le code de production.
Est-ce que les problèmes de test persistent?