

METTRE EN PLACE DES TESTS SUR DU CODE LEGACY



SOIRÉE TECHNIQUE - 11/10/2018

QUI SUIS-JE ?

Patrick GIRY

Software gardener



patrick.giry@arolla.fr

@PatrickGIRY

<https://github.com/PatrickGIRY>

20 ans d'expérience dans
le développement logiciel



Test Driven Development
Behavior Driven Development
Domain Driven Design
Functional programming

CO-ORGANISATEUR

Meetup

Dojo développement Paris



C
Language

FORTH



ProLog

Coding Dojo

KataCatalogue

- [KataBankOCR](#)
- [KataFizzBuzz](#)
- [FooBarQix](#)
- [KataPotter](#)
- [KataRomanNumerals](#)
- [KataRomanCalculator](#)
- [KataNumbersInWords](#)
- [KataArgs](#)
- [KataAnagram](#)
- [KataDepthFirstSearch](#)
- [KataNumberToLCD](#)

<https://github.com/dojo-developpement-paris/>

CO-ORGANISATEUR



SoCraTes France 2018 WHAT IT IS ABOUT? INFORMATION HISTORY FAMILY SPONSORS REGISTRATION WIKI


@SoCraTes_FR

*October 25-28th, 2018.
International Software Craftmanship and Testing
Conference.*

SOCRATES FRANCE 2018

TELL ME MORE





METTRE EN PLACE DES TESTS SUR DU CODE LEGACY

POURQUOI AJOUTER DES TESTS À UN CODE LEGACY?

- Pour pouvoir modifier le code existant (pour une évolution ou une correction) de manière sécurisée.
- Pour bien analyser le code et ses dépendances.
- Pour documenter le comportement attendu du code.

EXIGENCES MÉTIERS

Contexte

Un réseau social pour les voyages

- Vous devez être connecté pour voir le contenu.
- Vous devez être un ami pour voir les voyages de quelqu'un d'autre.


```
package trip;

import exception.UserNotLoggedInException;
import user.User;
import user.UserSession;

import java.util.ArrayList;
import java.util.List;

public class TripService {

    public List<Trip> getTripsByUser(User user) throws UserNotLoggedInException {
        List<Trip> tripList = new ArrayList<Trip>();
        User loggedInUser = UserSession.getInstance().getLoggedInUser();
        boolean isFriend = false;
        if (loggedInUser != null) {
            for (User friend : user.getFriends()) {
                if (friend.getId() == loggedInUser.getId()) {
                    isFriend = true;
                    break;
                }
            }
            if (isFriend) {
                tripList = tripDAO.findTripsByUser(user);
            }
            return tripList;
        } else {
            throw new UserNotLoggedInException();
        }
    }
}
```



RÈGLES POUR LE CODE LEGACY

**Vous ne pouvez pas changer
le code production si il n'est
pas couvert par les tests ?**

**Seul le refactoring automatique
(via IDE) est permis si c'est
nécessaire pour écrire les tests.**

ASTUCES POUR TRAVAILLER AVEC DU CODE LEGACY

Commencer les tests de la branche la moins profonde vers la branche la plus profondes.



```
package trip;

import exception.UserNotLoggedInException;
import user.User;
import user.UserSession;

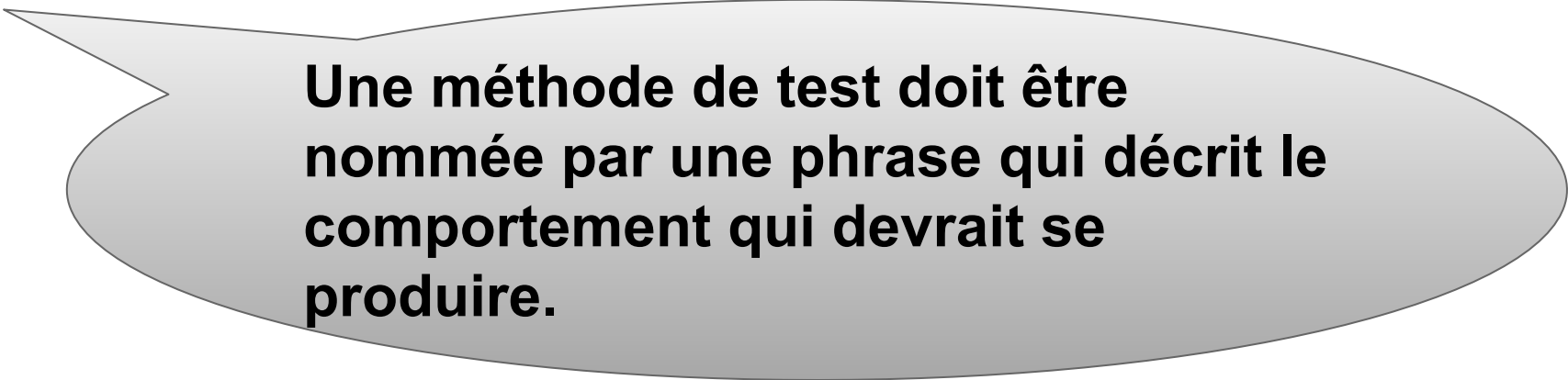
import java.util.ArrayList;
import java.util.List;

public class TripService {

    public List<Trip> getTripsByUser(User user) throws UserNotLoggedInException {
        List<Trip> tripList = new ArrayList<Trip>();
        User loggedInUser = UserSession.getInstance().getLoggedInUser();
        boolean isFriend = false;
        if (loggedInUser != null) {
            for (User friend : user.getFriends()) {
                if (friend.getId() == loggedInUser.getId()) {
                    isFriend = true;
                    break;
                }
            }
            if (isFriend) {
                tripList = tripDAO.findTripsByUser(user);
            }
            return tripList;
        } else {
            throw new UserNotLoggedInException();
        }
    }
}
```



COMMENT NOMMER UNE MÉTHODE DE TEST?



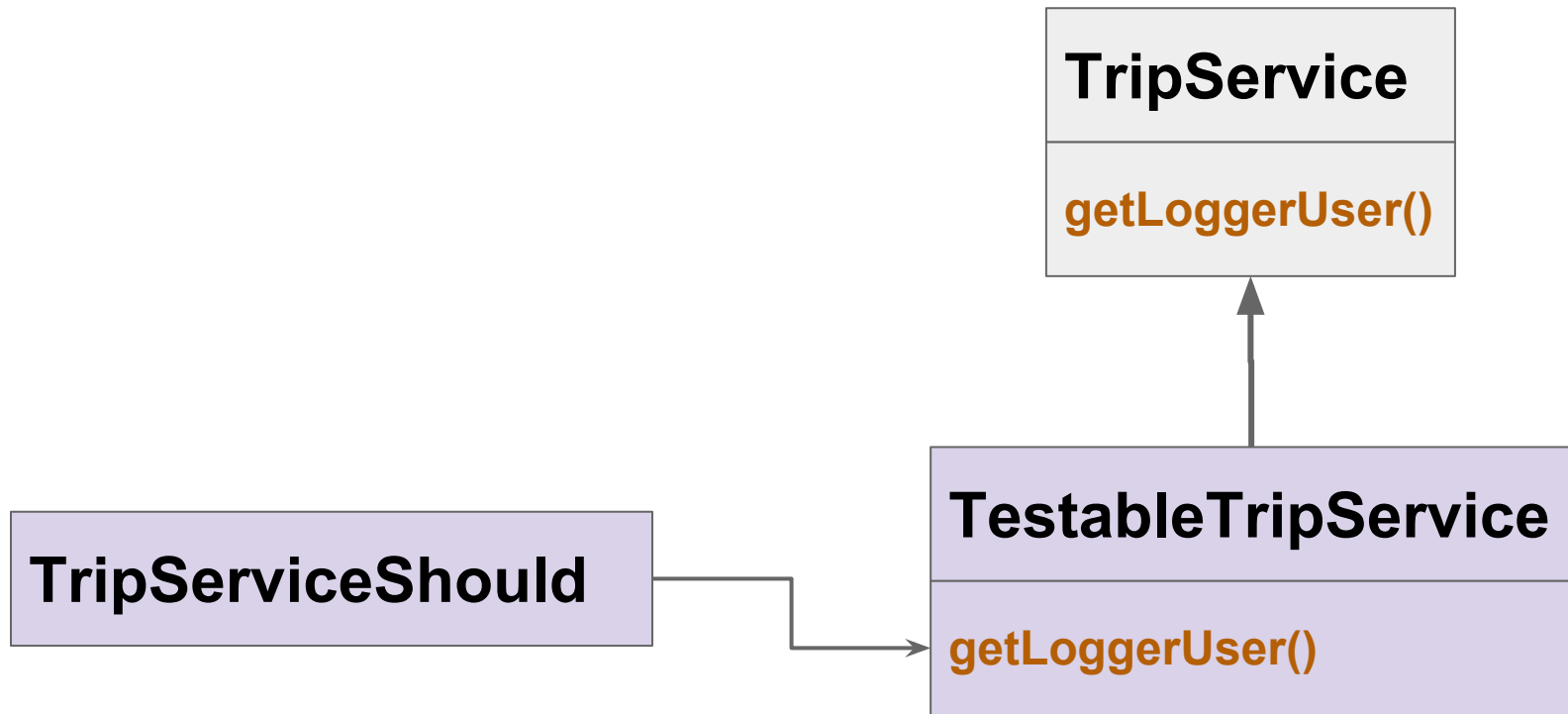
Une méthode de test doit être nommée par une phrase qui décrit le comportement qui devrait se produire.

TEST DE LA BRANCHE LA PLUS COURTE



**Should throw an exception when
user not logged in**

MAÎTRISER L'UTILISATEUR CONNECTÉ POUR LES TESTS



SUPPRIMER LES VALEURS "MAGIQUES"

- Ajouter le concept d'utilisateur "invité".
- Ajouter le concept d'utilisateur non utilisé.

AUTRES TESTS




**Should not return any trips when
users are not friends**



**Should return friend trips when
users are friends**

LES COMPORTEMENTS ATTENDUS SONT TESTÉS




**On peut faire du refactoring sur le
code production.**

ASTUCES POUR TRAVAILLER AVEC DU CODE LEGACY



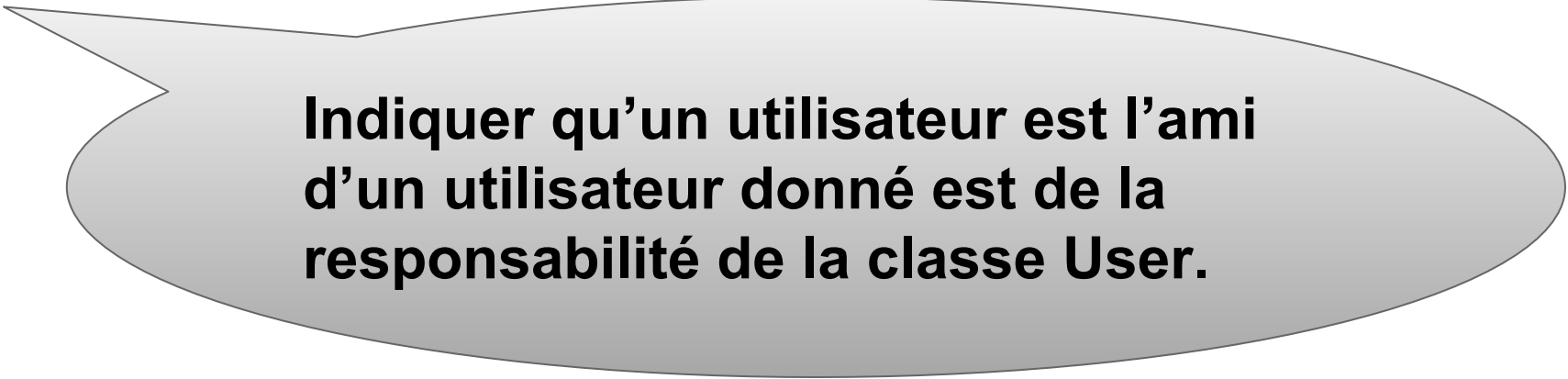
**Commencer le refactoring
par la branche la plus
profonde vers la branche
la moins profonde.**

DÉCLARER LES VARIABLES AU PLUS PROCHE DE LEURS USAGES



**La variable isFriend n'est utilisé que
dans le bloc du if.**

PRINCIPE DE RESPONSABILITÉ UNIQUE

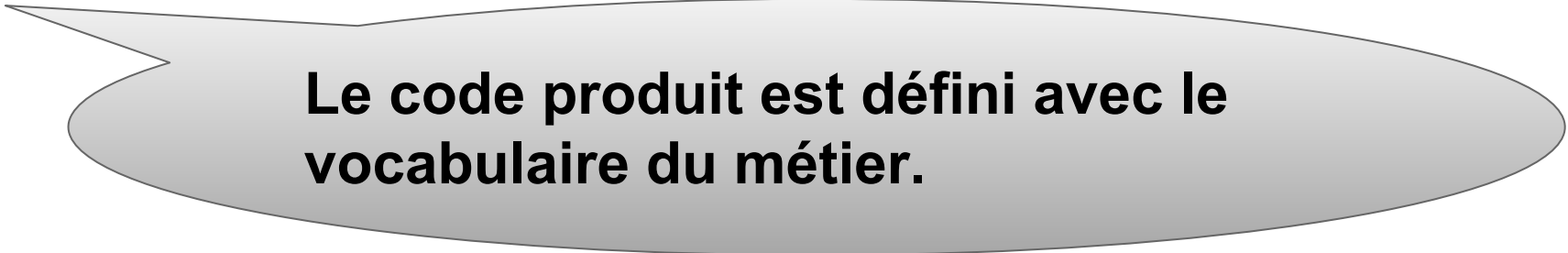


**Indiquer qu'un utilisateur est l'ami
d'un utilisateur donné est de la
responsabilité de la classe User.**

LIVING DOCUMENTATION + UBIQUITOUS LANGUAGE



**Les tests spécifient et documentent
les comportements attendus.**



**Le code produit est défini avec le
vocabulaire du métier.**

TRAVAIL D'ARTISAN

- ❑ Ecrire du code lisible et maintenable
 - Le code doit exprimer des règles métiers
- ❑ S'efforcer de faire simple
- ❑ Bien connaître ses outils (par exemple les frameworks, les raccourcis, etc ...)
- ❑ Travailler en petits incréments sûres
 - Valider souvent
- ❑ Être courageux, ne pas hésiter à faire des changements



<https://github.com/PatrickGIRY/tripServiceKataInJava10>

