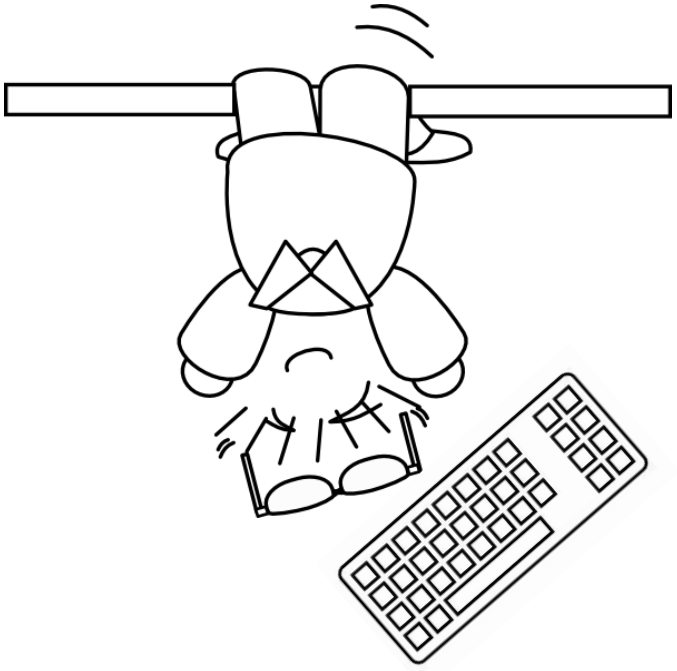
# Coder à l'envers

## pour

# penser à l'endroit

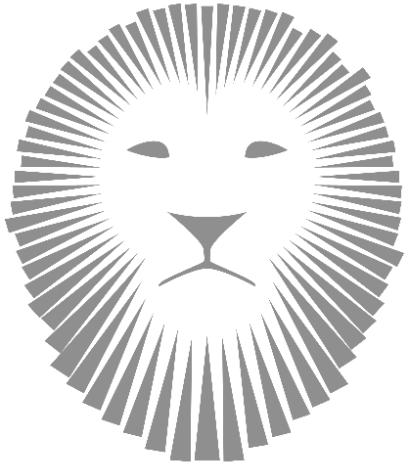
1ère partie



## Une initiation au

# TDD

(Test Driven Development)



# Khris

lyontechhub.slack.com

# Le TDD ...

# ... c’est quoi ?

*C’est une pratique …*

*… de développement*

# *C'est une pratique ...*

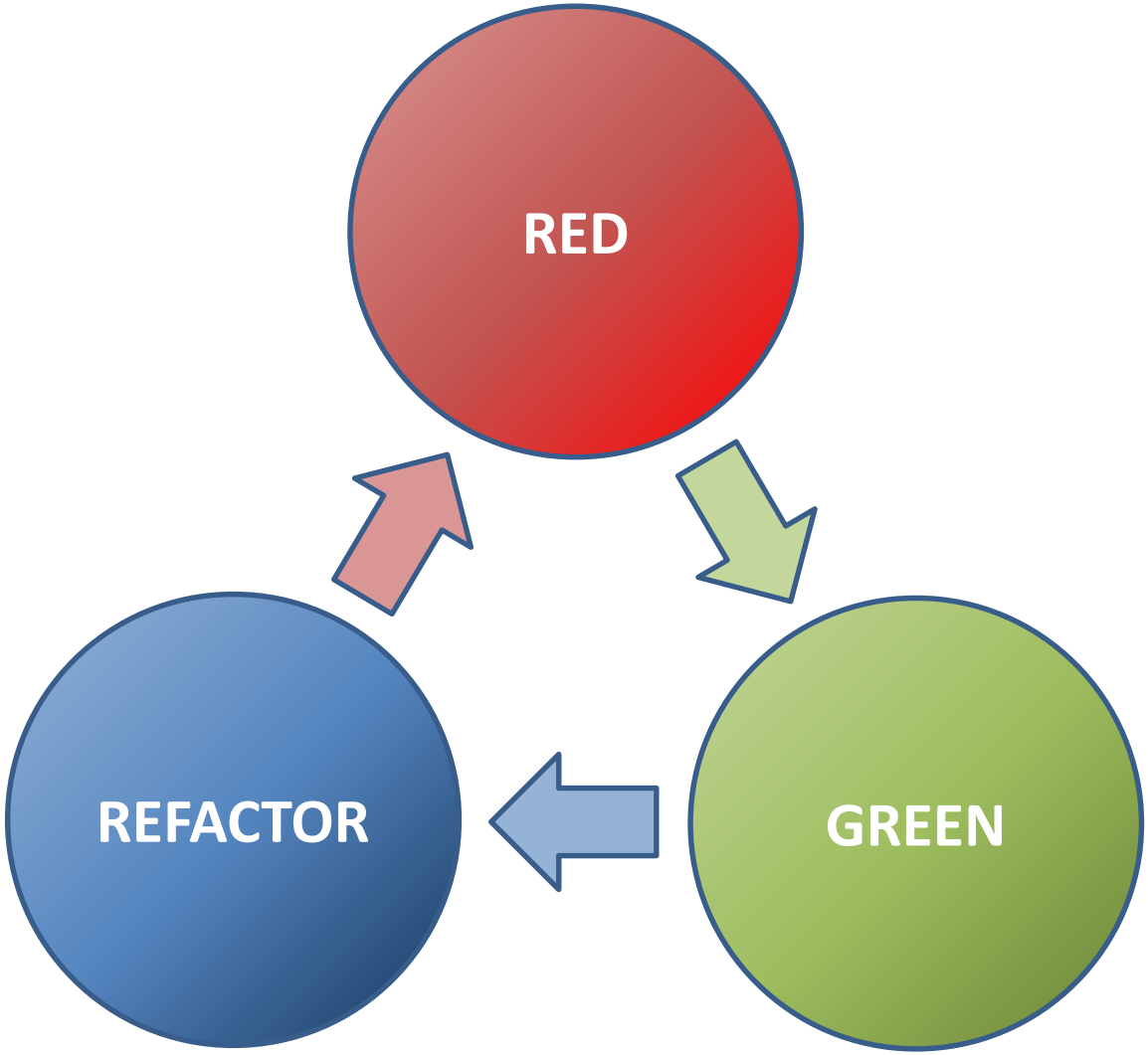
# *... de développement*

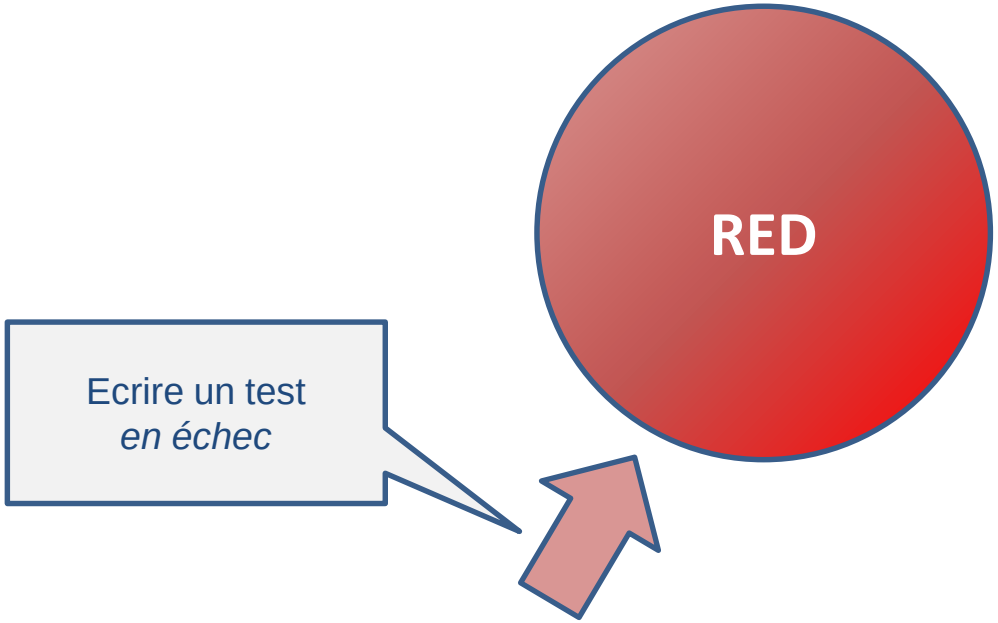
**Et non une pratique de test**

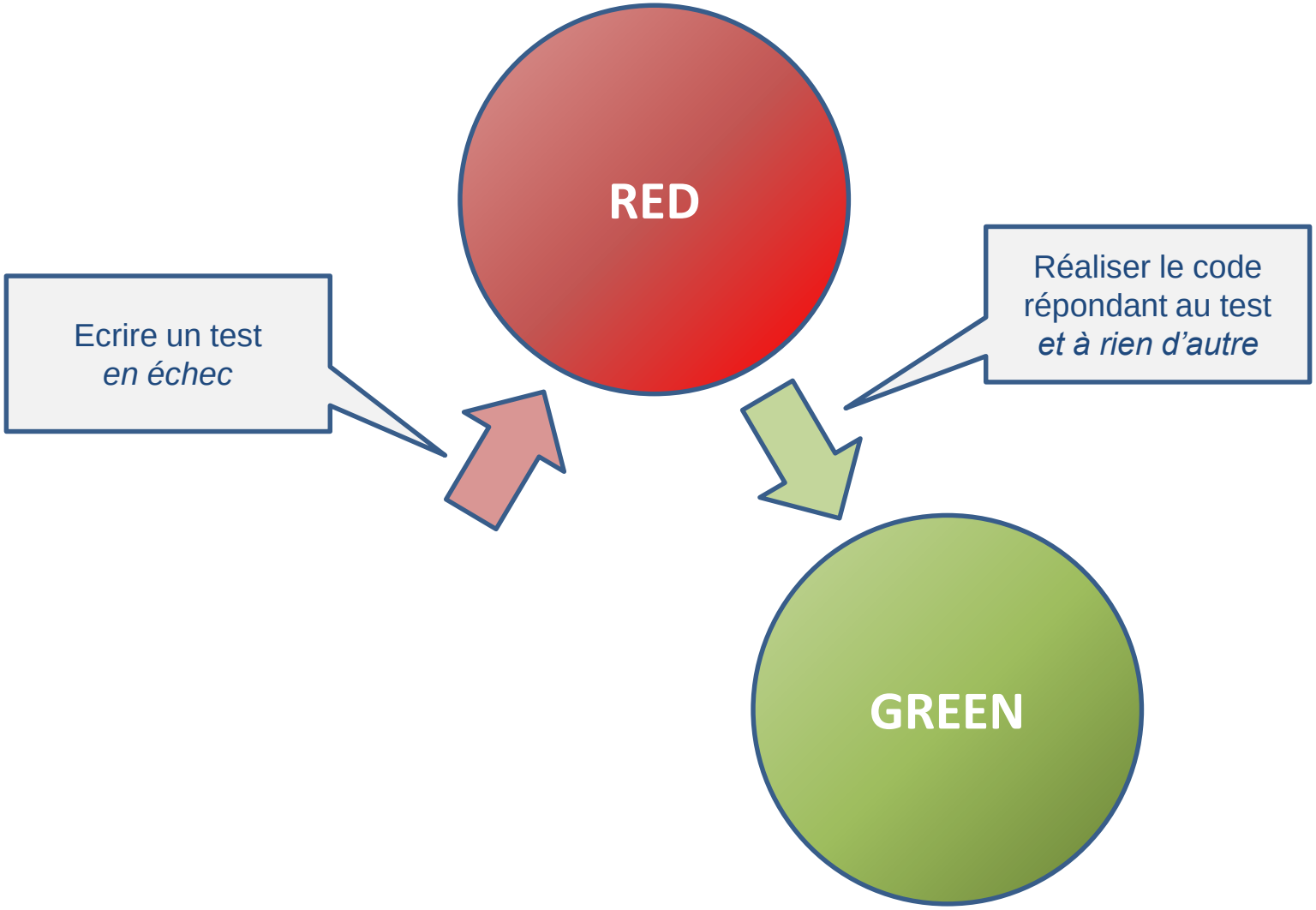
**Son objectif est de produire du code de production**

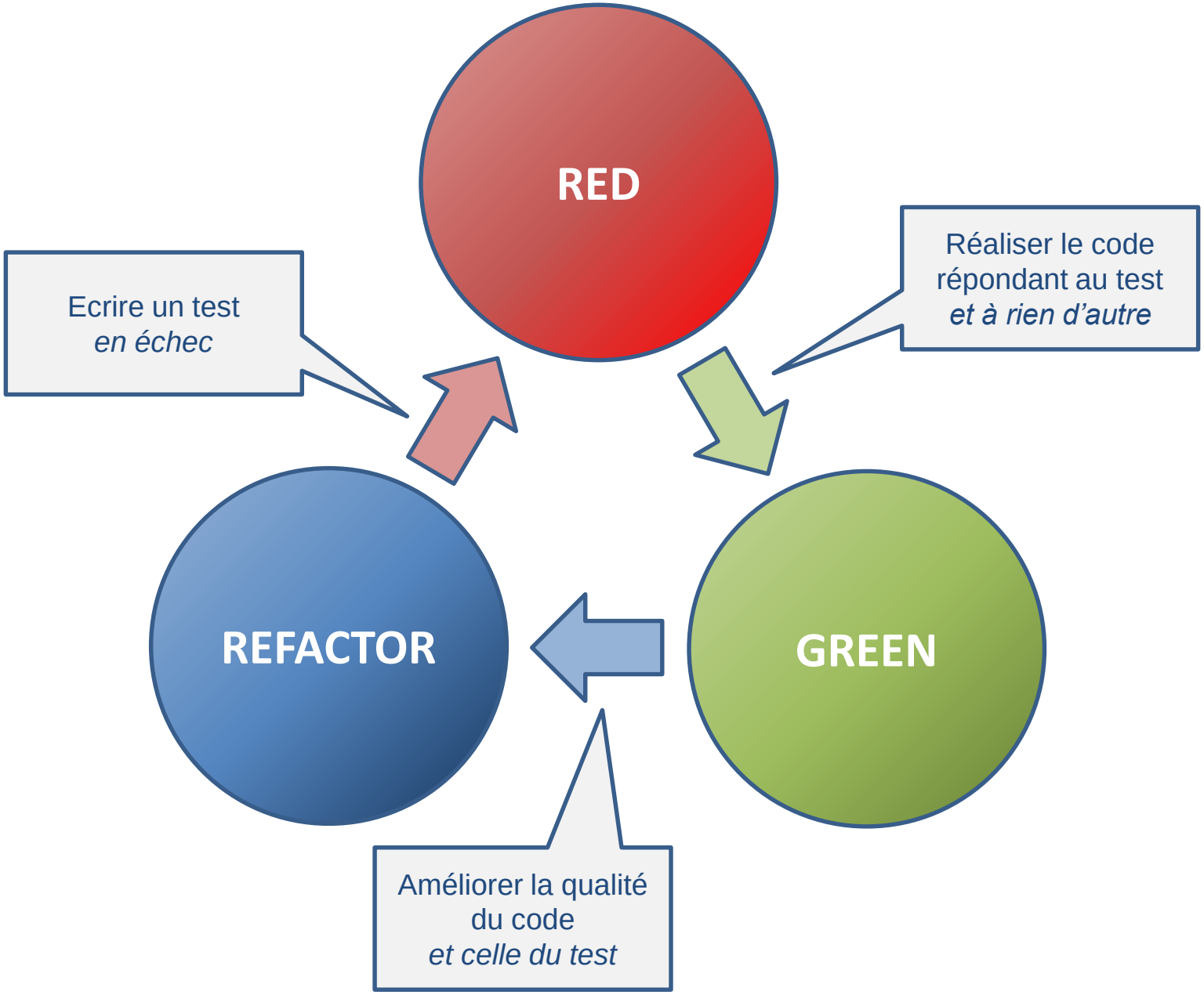
**La couverture de tests est un heureux effet secondaire**

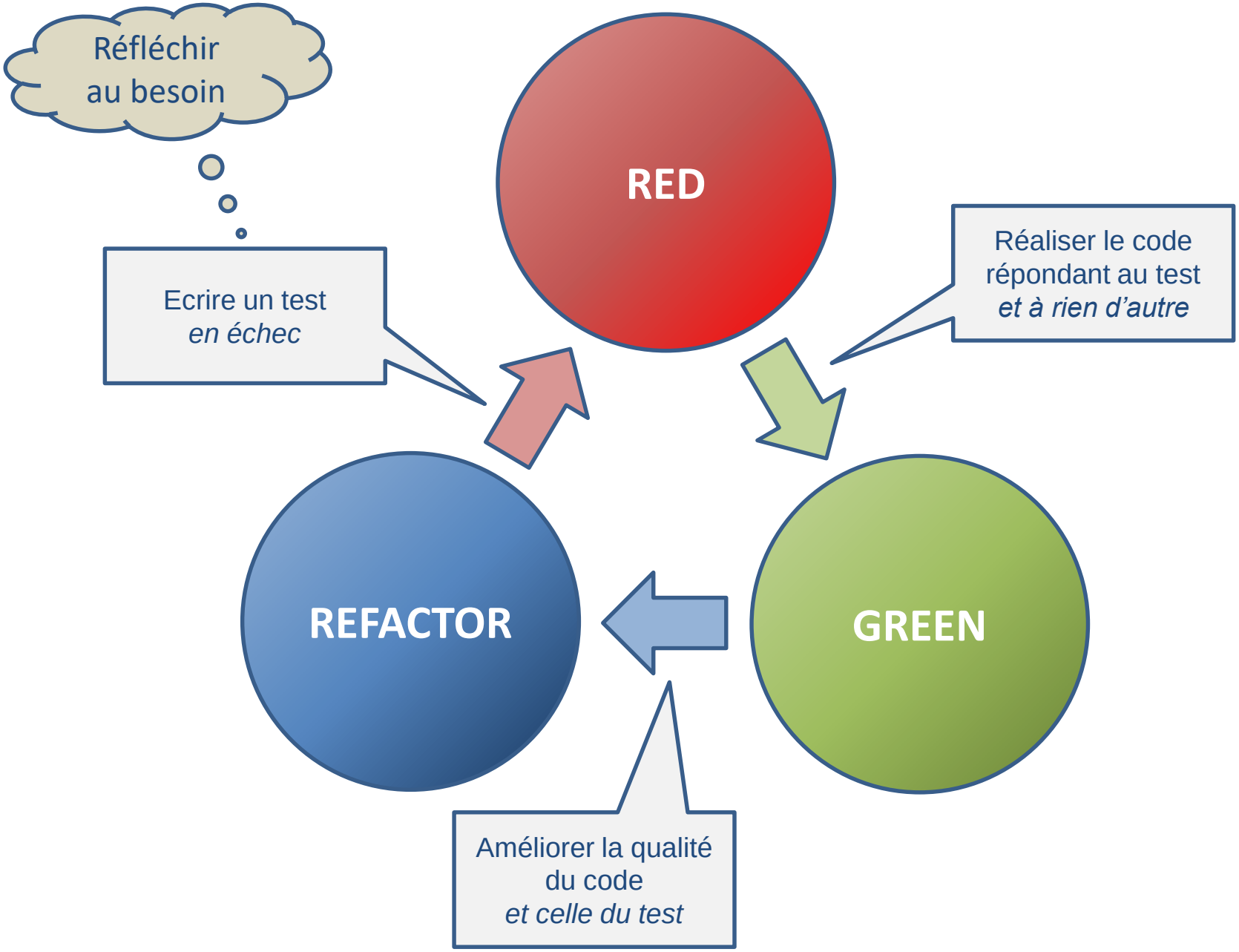
**C'est une pratique XP parmi d'autres (complémentaires)**

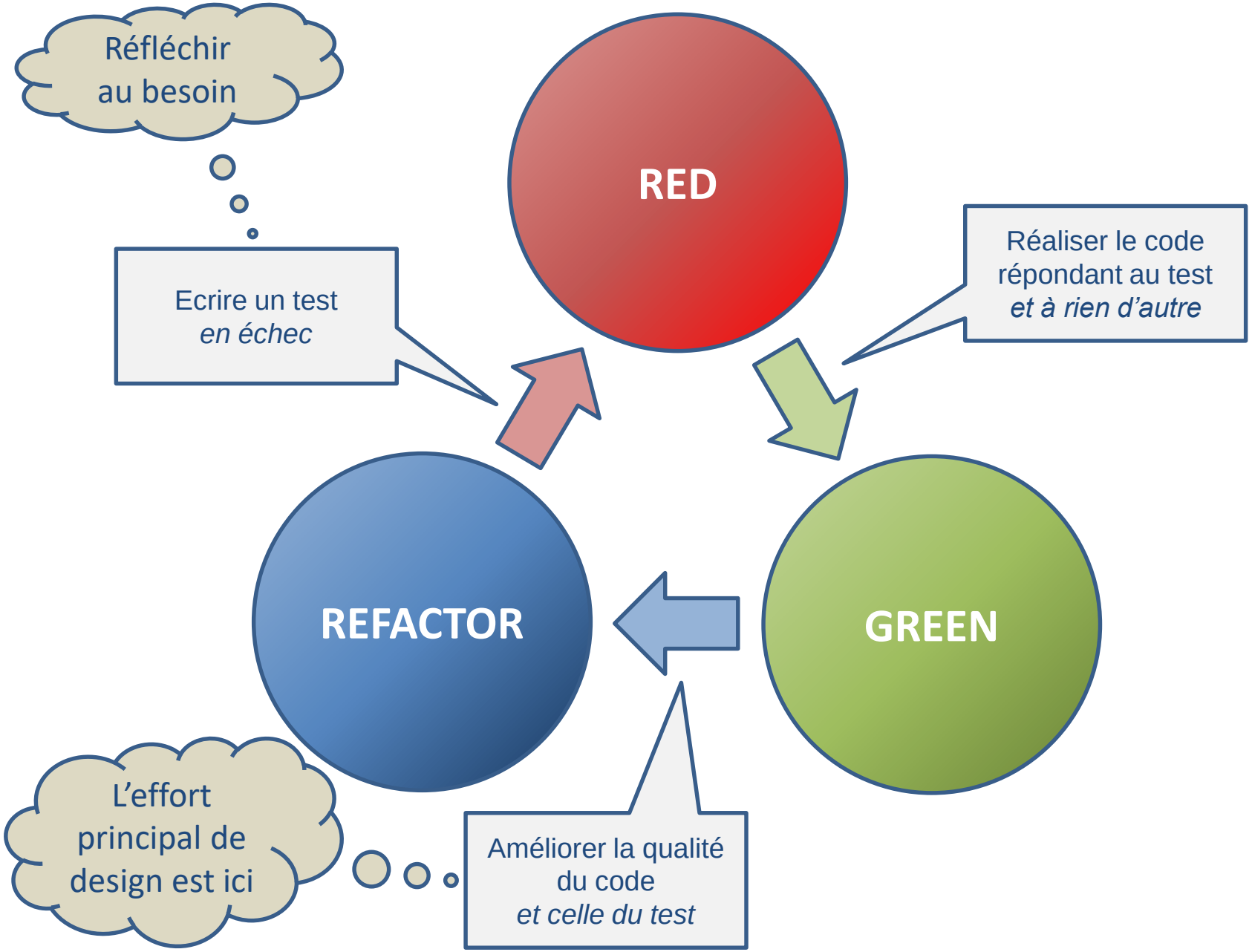

RED
GREEN
REFACTOR


RED
Ecrire un test
*en échec*


RED
Ecrire un test
*en échec*
Réaliser le code
répondant au test
*et à rien d'autre*
GREEN

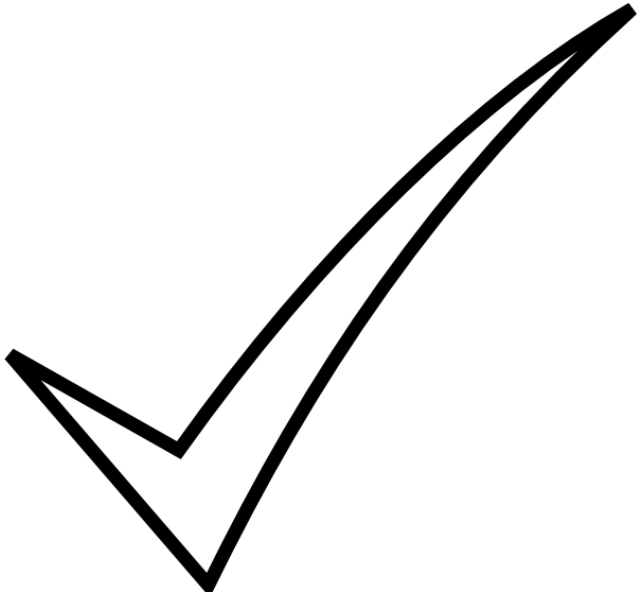

RED
Réaliser le code répondant au test *et à rien d'autre*
GREEN
Améliorer la qualité du code *et celle du test*
REFACTOR
Ecrire un test *en échec*


Réfléchir
au besoin
Ecrire un test
*en échec*
RED
Réaliser le code
répondant au test
*et à rien d'autre*
GREEN
Améliorer la qualité
du code
*et celle du test*
REFACTOR

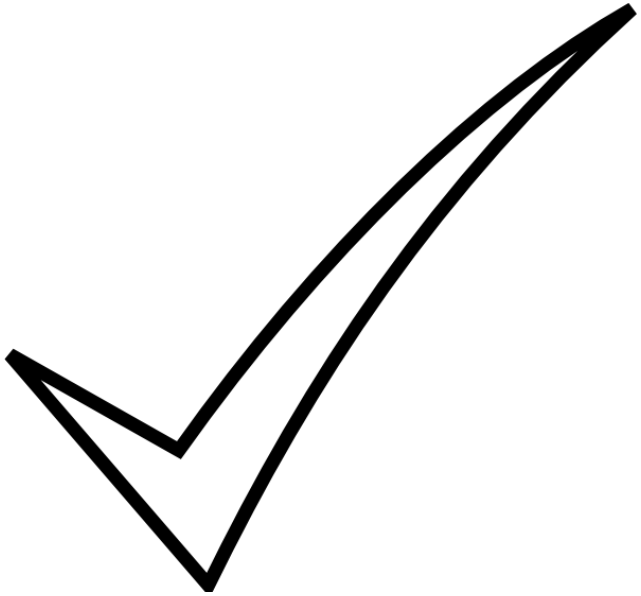

Réfléchir
au besoin
RED
Ecrire un test
*en échec*
Réaliser le code
répondant au test
*et à rien d'autre*
REFACTOR
GREEN
L'effort
principal de
design est ici
Améliorer la qualité
du code
*et celle du test*

**Oui mais ...**

**... dans quel but ?**



***Produit de qualité***



# *Produit de qualité*

**Satisfaction des utilisateurs**

**Robustesse du produit**

**Tolérance au changement**

**Instauration de la confiance**

**Cycle d’amélioration continue**

**Oui mais ...**

**... en quoi le TDD aide à tout cela ?**

# *Biais cognitifs*

# *Biais cognitifs*

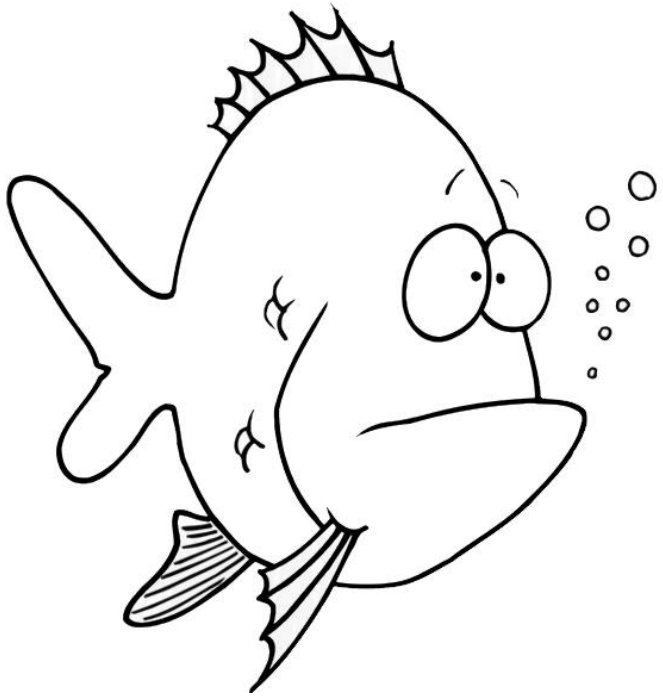
**Quatre grandes catégories de biais :**

- **Trop d'informations**
- **Limites de la mémoire**
- **Manque de sens**
- **Action dans l'urgence**

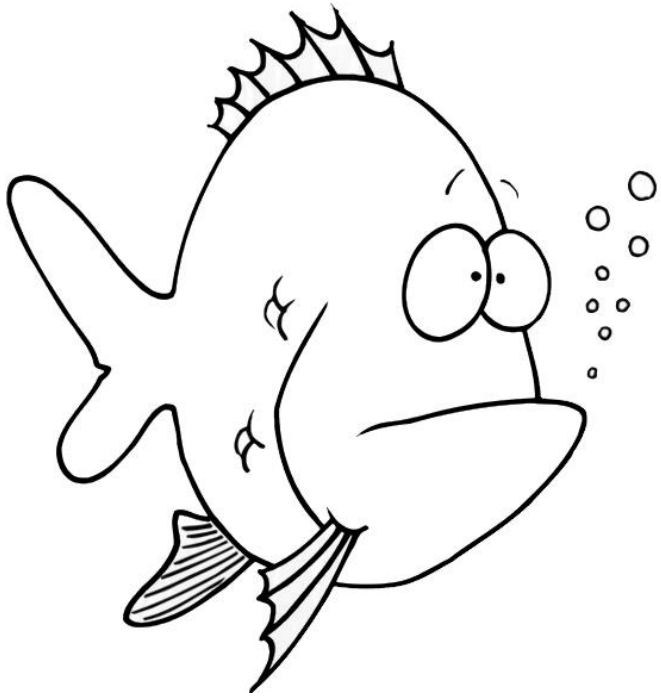
# *Biais cognitifs*

**Quatre grandes catégories de biais :**

| | |
|---|---|
| **- Trop d'informations** | **biais de confirmation** |
| **- Limites de la mémoire** | **effet test** |
| **- Manque de sens** | **biais de conformisme** |
| **- Action dans l'urgence** | **effet moins-c'est-mieux** |



# *Capacité d'attention*

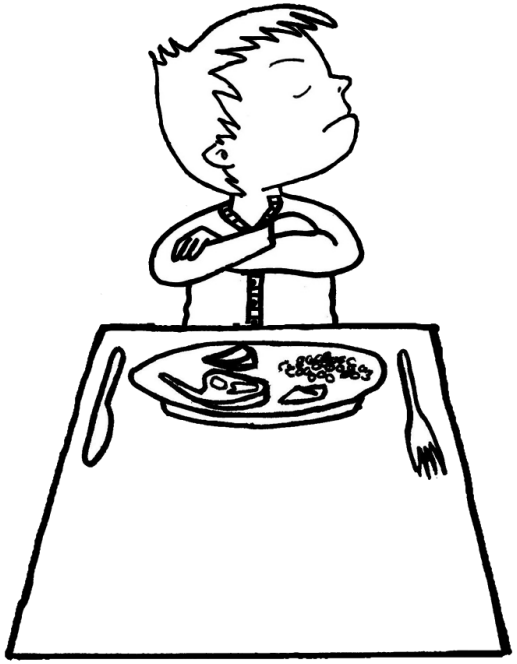


# *Capacité d'attention*

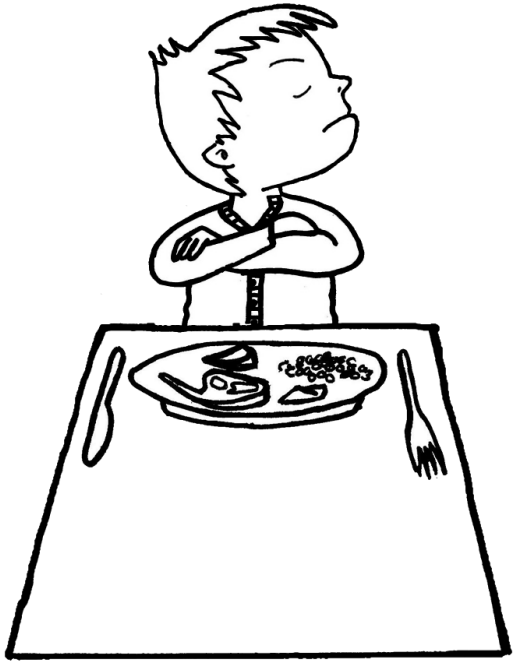
**TDD**

**Cadre sur lequel se concentrer**

**Point de repère dans le temps et l'avancement**



# *Conditionnement opérant*



# *Conditionnement opérant*

**TDD**

**Le « vert » valorise un aboutissement**

**Le « rouge » souligne une incomplétude**

**La confiance augmente au fil des cycles**


# *Apprentissage*

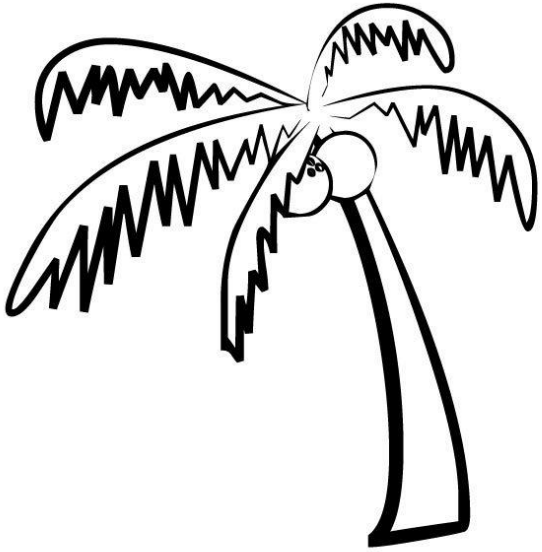

# *Apprentissage*

**Différentes effets combinés :**

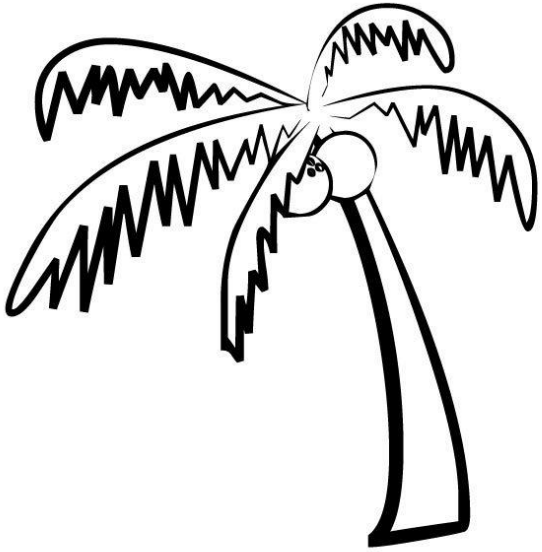
**TDD**

- **Essai-erreur**
- **Répétition**
- **Emergence de la solution**

**Amélioration continue**

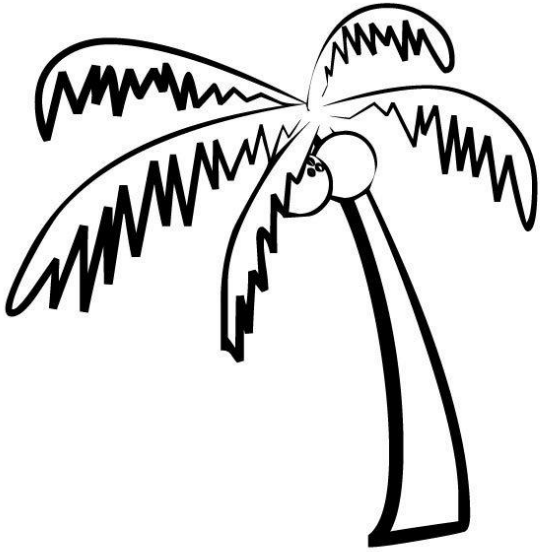


# *Procrastination*



# Procrastination

TODO...



# *Procrastination*

**TDD**

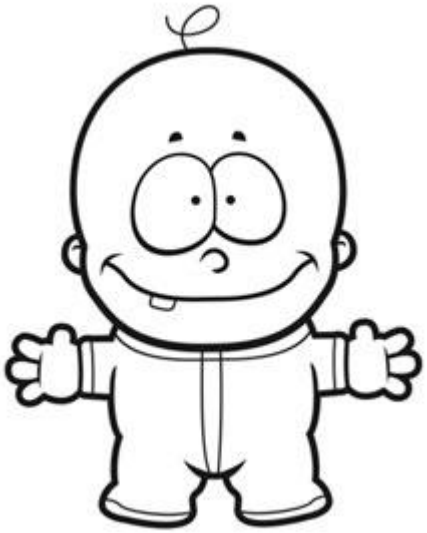
**Indicateur d'avancement**

**Progression itérative**

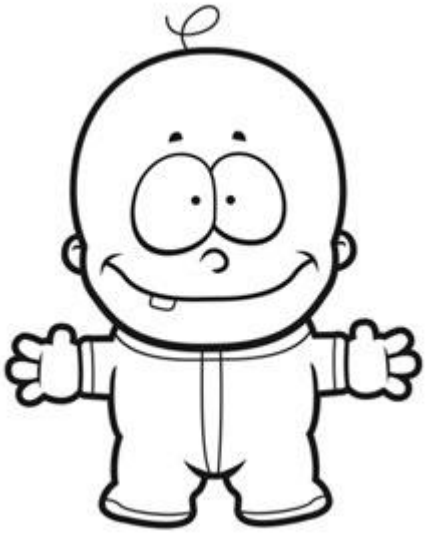
**Les tests ne sont plus oubliés ou remis à « demain »**

**Oui mais dans ce cas ...**

**... pourquoi est-il si peu répandu ?**



# *Un métier émergent*



# *Un métier émergent*

**Une grande complexité globale des variables en jeu**

**Très peu de recul sur les différentes pratiques**

**Les programmes scolaires évoluent … à leur rythme**

*Facile de loin ...*

*... loin d'être facile*

# *Facile de loin ...*

# *... loin d'être facile*

**Bousculer ses habitudes de travail**

**Se plier à un cycle contraint**

**Comprendre avant de coder**

**C'est une pratique, pas une formule magique, ni un dogme**

## *Un effet « grossissant »*

# *Un effet « grossissant »*

**Se heurter au code existant**

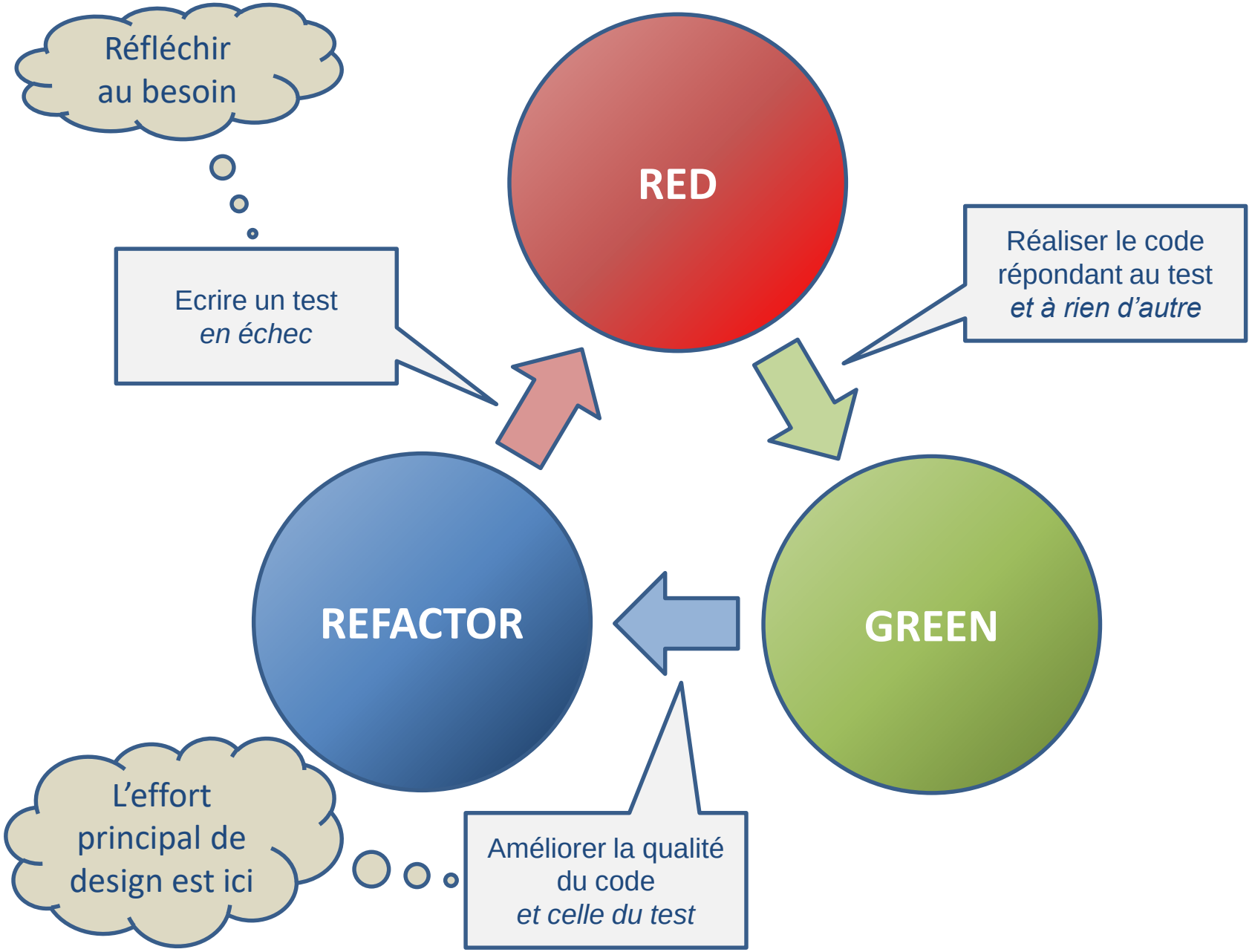
**Découvrir des problèmes – les accepter et s'améliorer**

- **design**
- **expression des besoins**
- **organisation**
- **…**

**Ne pas blâmer la pratique pour ce qu'elle révèle**

**Revenons à nos moutons ...**

**... le TDD c'est donc :**


Réfléchir
au besoin
RED
Ecrire un test
*en échec*
Réaliser le code
répondant au test
*et à rien d'autre*
REFACTOR
GREEN
L'effort
principal de
design est ici
Améliorer la qualité
du code
*et celle du test*

**Oui mais en parlant de design …**

**… qu’est ce qu’un bon design ?**

# *3 catégories de design*

# *3 catégories de design*

**Architecture**

**Macro design**

**Micro design**

# 3 catégories de design

**Architecture** = infrastructure, persistance ...

**Macro design** = packages, dépendances ...

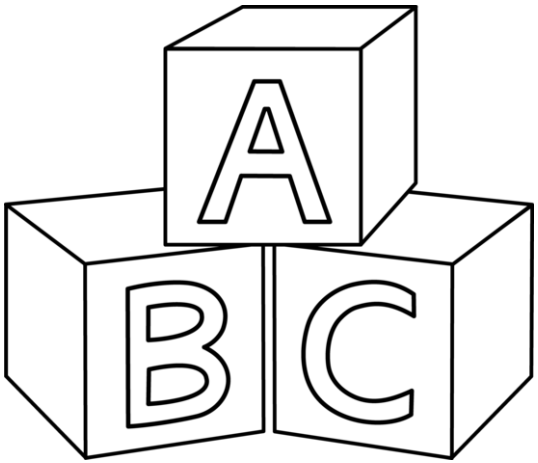
**Micro design** = code métier, règles de gestion ...

# *3 catégories de design*

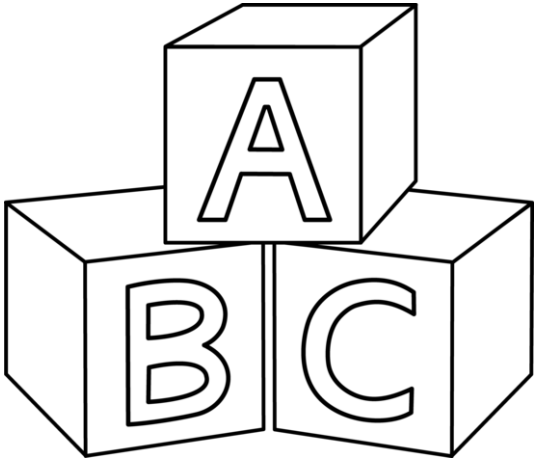
**Architecture** = infrastructure, persistance ...

**Macro design** = packages, dépendances ...

TDD → **Micro design** = code métier, règles de gestion ...




# *4 règles élémentaires de conception*



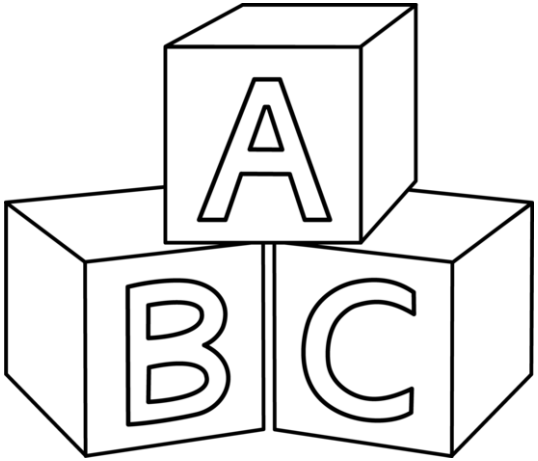

# *4 règles élémentaires de conception*

**Tous les tests sont au vert**

**Le code révèle l'intention**

**Pas de duplication**

**Pas d'élément superflu**




# *4 règles élémentaires de conception*

**Tous les tests sont au vert**

**Le code révèle l'intention**

**Pas de duplication**

**Pas d'élément superflu**

**Oui mais ...**

**... ça ne suffit pas !**

# *Et la route est longue*

# *Et la route est longue*

**Design Patterns**

**Principes SOLID**

**Clean Code**

**Domain Driven Design**

**...**

# Donc ...

# ... comment démarrer ?

# *Quelques guidelines pour le cycle TDD*

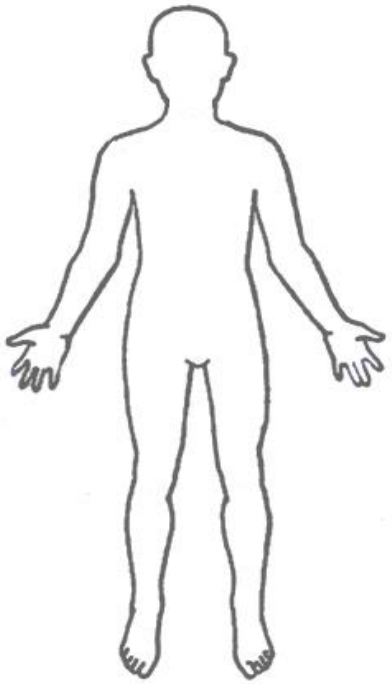
# *Quelques guidelines pour le cycle TDD*

**Ajouter un (et un seul) nouveau test seulement « au vert »**

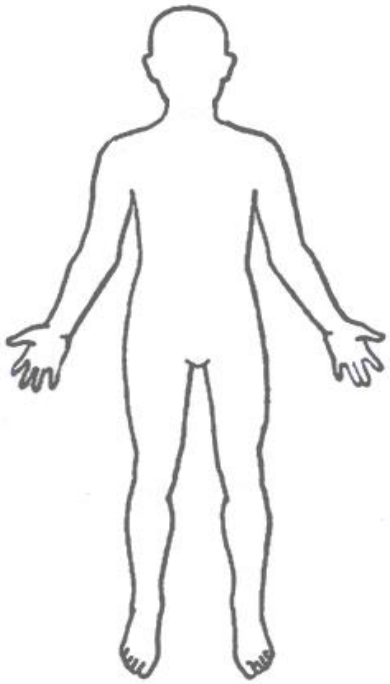
**Voir échouer le test avant de coder sa solution**

**Coder dans l'optique de revenir au plus vite « au vert »**

**Effectuer un refactoring du code ou du test, pas des deux à la fois**



# *Anatomie d'un test*

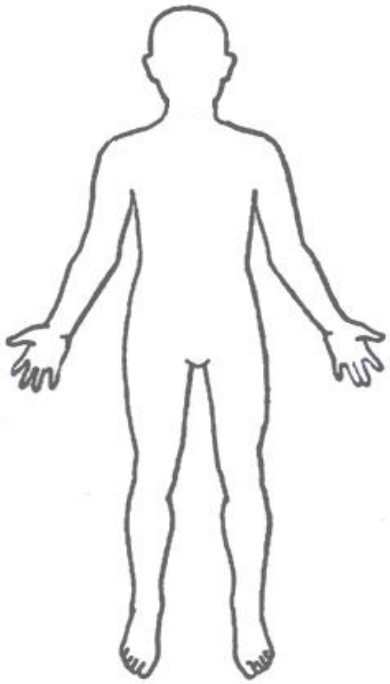


# *Anatomie d'un test*

**GIVEN**

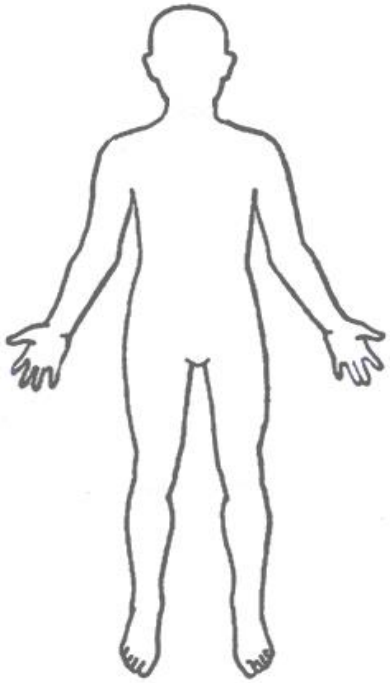
**WHEN**

**THEN**



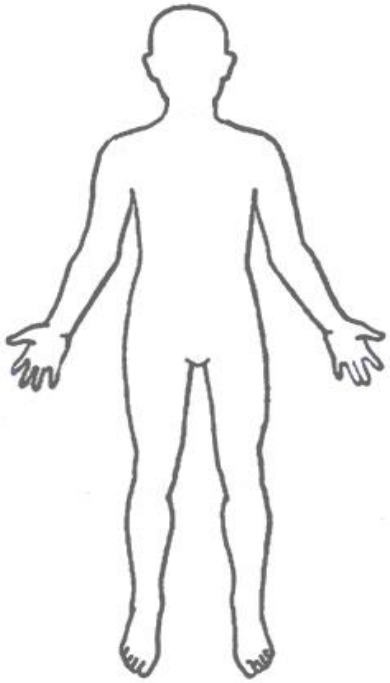
# *Anatomie d'un test*

| | |
|---|---|
| GIVEN | **Contexte** |
| WHEN | **Événement** |
| THEN | **Attendu** |



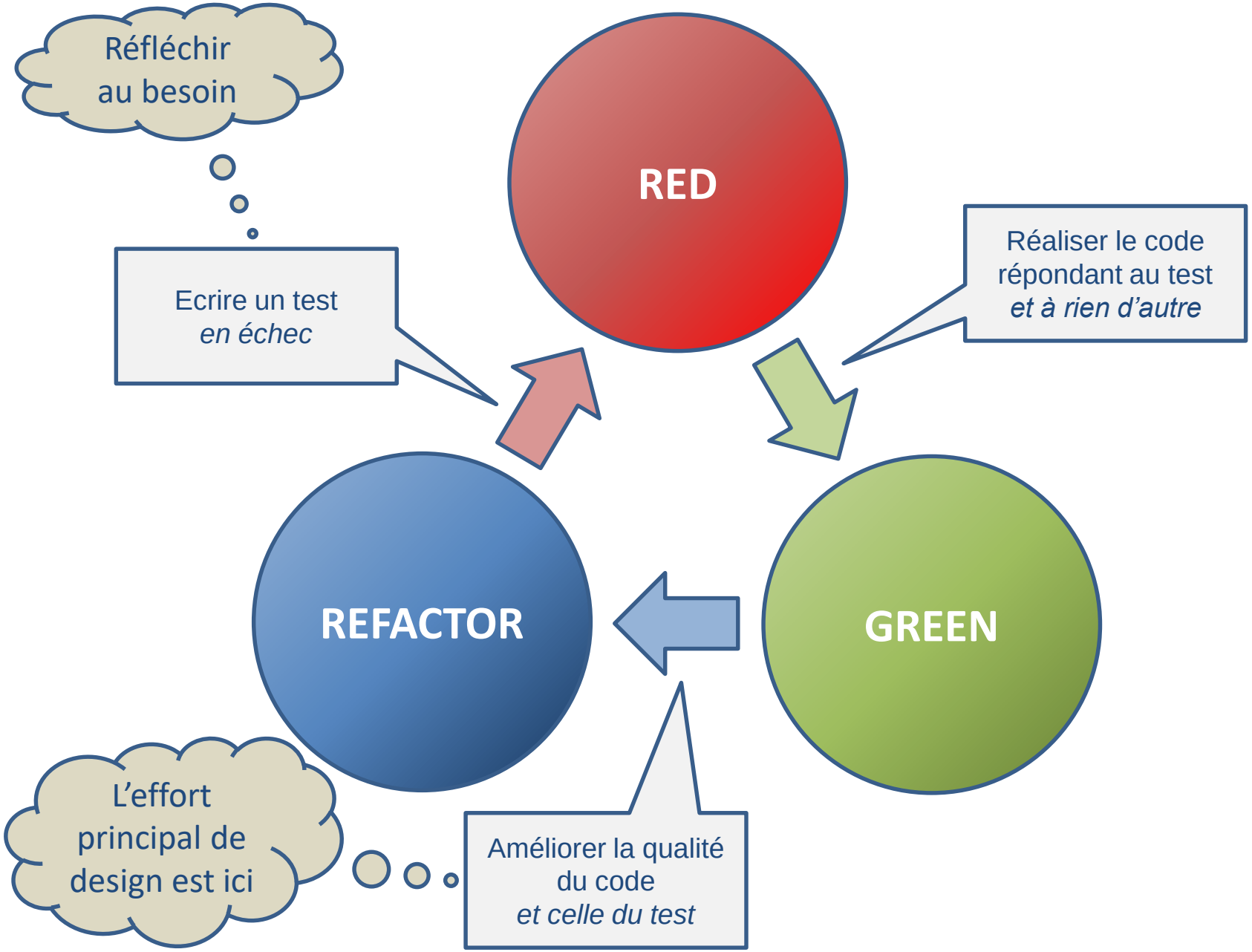
# *Anatomie d'un test*

| | | |
|---|---|---|
| GIVEN | **Contexte** | **= états / données** |
| WHEN | **Événement** | **= ce qui est testé** |
| THEN | **Attendu** | **= réponse au besoin** |



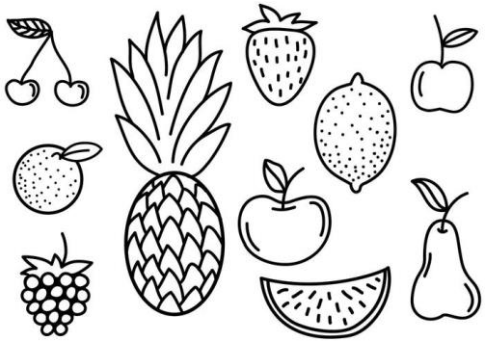
# *Anatomie d'un test*

| | | | |
|---|---|---|---|
| 3 | GIVEN | Contexte | = états / données |
| 2 | WHEN | Événement | = ce qui est testé |
| 1 | THEN | Attendu | = réponse au besoin |


Réfléchir
au besoin
RED
Ecrire un test
*en échec*
Réaliser le code
répondant au test
*et à rien d'autre*
REFACTOR
GREEN
L'effort
principal de
design est ici
Améliorer la qualité
du code
*et celle du test*

**Au final, la clef c’est …**

**… la pratique !**



# *Fruit Shop*

**Faire payer le bon montant quand le client passe en caisse.**

**Règles de gestion (prix en centimes) :**

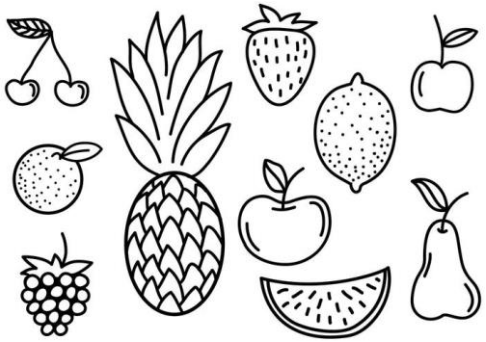
- **Une pomme coûte 100**
- **Une banane coûte 150**
- **Une cerise coûte 75**

```
@Test
public void neRienPayerPourUnPanierVide() {

}
```

```java
@Test
public void neRienPayerPourUnPanierVide() {

    // THEN
    assertThat(prixAPayer).isEqualTo(0);
}
```

```java
@Test
public void neRienPayerPourUnPanierVide() {

    // WHEN
    int prixAPayer = panier.calculerMontantTotal();

    // THEN
    assertThat(prixAPayer).isEqualTo(0);
}
```

```java
@Test
public void neRienPayerPourUnPanierVide() {

    // GIVEN
    Panier panier = new Panier();

    // WHEN
    int prixAPayer = panier.calculerMontantTotal();

    // THEN
    assertThat(prixAPayer).isEqualTo(0);
}
```

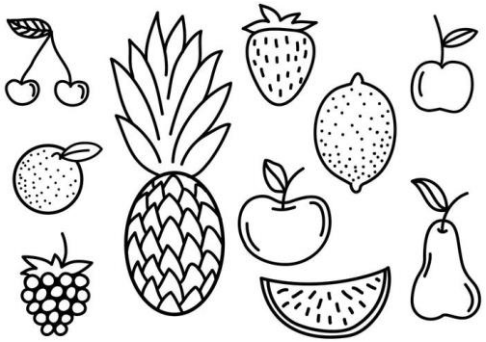


# *Fruit Shop*

**Faire payer le bon montant quand le client passe en caisse.**

**Règles de gestion (prix en centimes) :**

- **Une pomme coûte 100**
- **Une banane coûte 150**
- **Une cerise coûte 75**

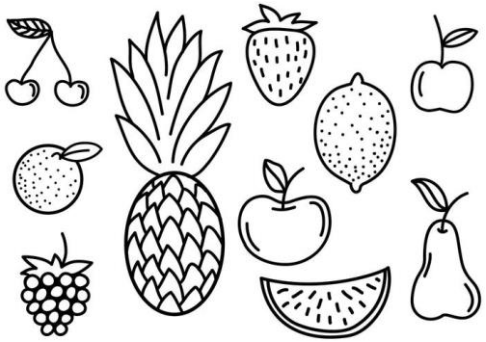


# *Fruit Shop*

**Faire payer le bon montant quand le client passe en caisse.**

**Règles de gestion (prix en centimes) :**

- **Une pomme coûte 100**
- **Une pomme offerte pour deux pommes achetées**
- **Une banane coûte 150**
- **Une cerise coûte 75**

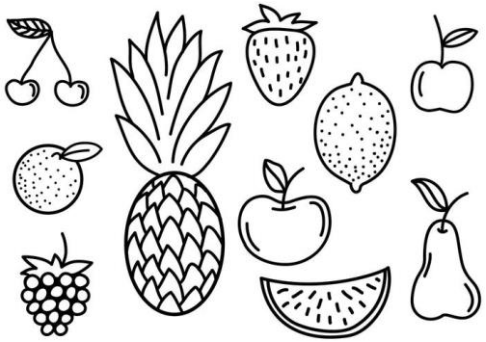


# *Fruit Shop*

**Faire payer le bon montant quand le client passe en caisse.**

**Règles de gestion (prix en centimes) :**

- **Une pomme coûte 100**
- **Une pomme offerte pour deux pommes achetées**
- **Une banane coûte 150**
- **Une cerise coûte 75**
- **Il faut pouvoir vendre des « apples » en Angleterre**

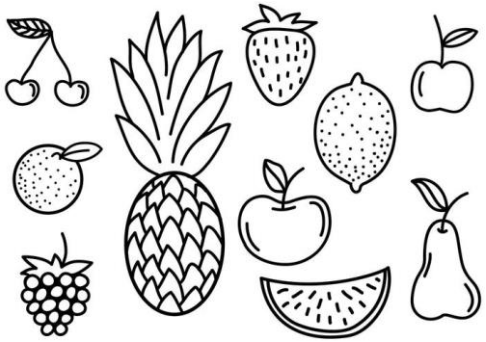


# *Fruit Shop*

**Faire payer le bon montant quand le client passe en caisse.**

**Règles de gestion (prix en centimes) :**

- **Une pomme coûte 100**
- **Une pomme offerte pour deux pommes achetées**
- **Une banane coûte 150**
- **La deuxième banane est à moitié prix**
- **Une cerise coûte 75**

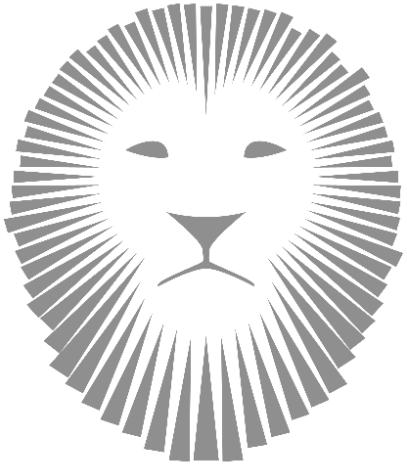


# *Fruit Shop*

**Faire payer le bon montant quand le client passe en caisse.**

**Règles de gestion (prix en centimes) :**

- **Une pomme coûte 100**
- **Une pomme offerte pour deux pommes achetées**
- **Une banane coûte 150**
- **La deuxième banane est à moitié prix**
- **Une cerise coûte 75**
- **Un client fidèle a droit à 10% de remise**



# Merci à tous !

lyontechhub.slack.com