Le property based testing pour répondre aux questions métier

- C'est quoi une propriété ?
- Des propriétés ? dans mon code ?
- Comment je les teste ?
- Le fonctionnel dans tout ça ?
- La petite démonstration

Property Test Based

"Les tests basés sur les propriétés sont conçus pour tester les aspects d'une propriété qui devraient toujours être vrais : les invariants"

Exemple : Distributivité de la multiplication

$$\forall a, b, c \in \mathbb{Z} \quad a * (b + c) = a * b + a * c$$

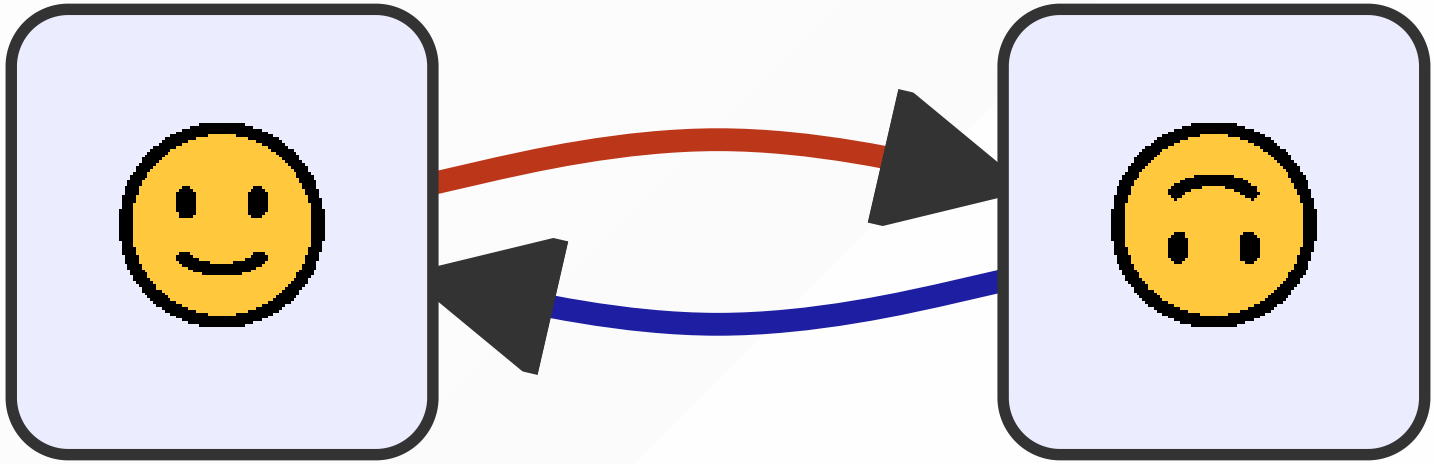
Des propriétés dans mon application ?

Dans tous les systèmes il y a des propriétés mais elles ne sont pas toujours faciles à trouver.

Quelques Patterns

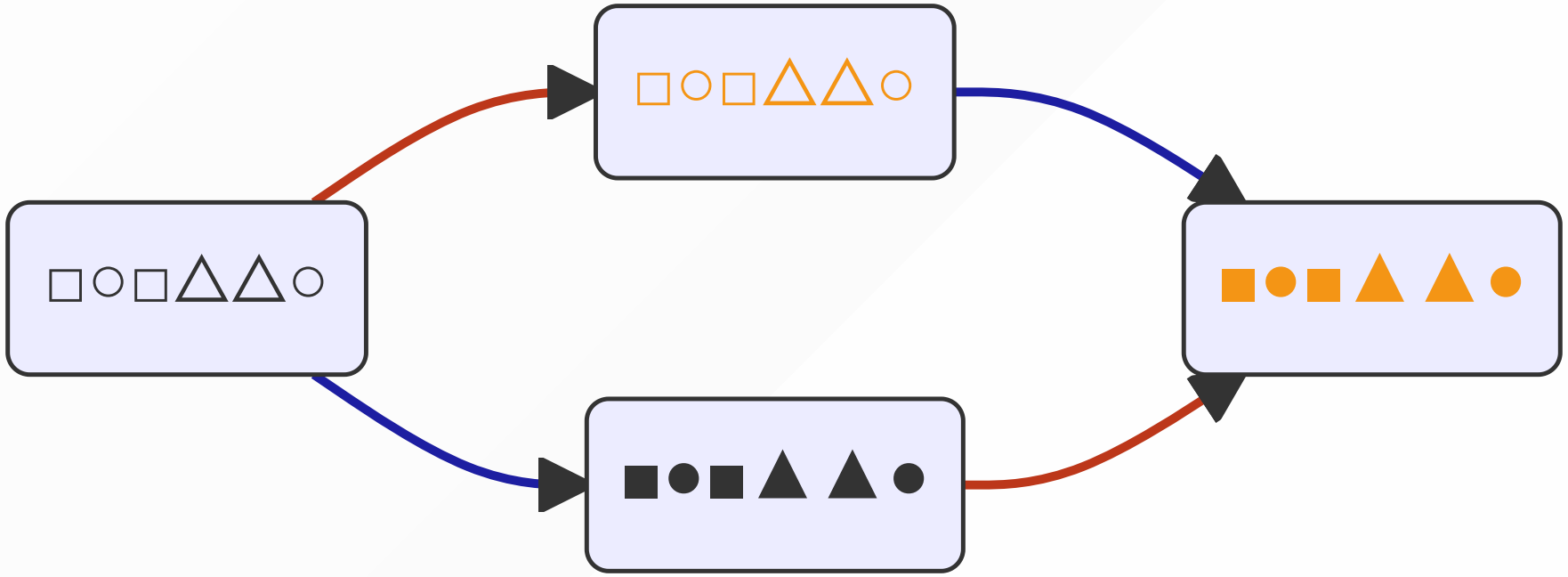
- Aller-retour
- Commutativité
- Invariance
- Idempotence
- Oracle

Aller-retour



```
@Property
boolean roundTripping(@ForAll int i) {
    return Math.log(Math.exp(i)) == i ;
}
```

Commutativité

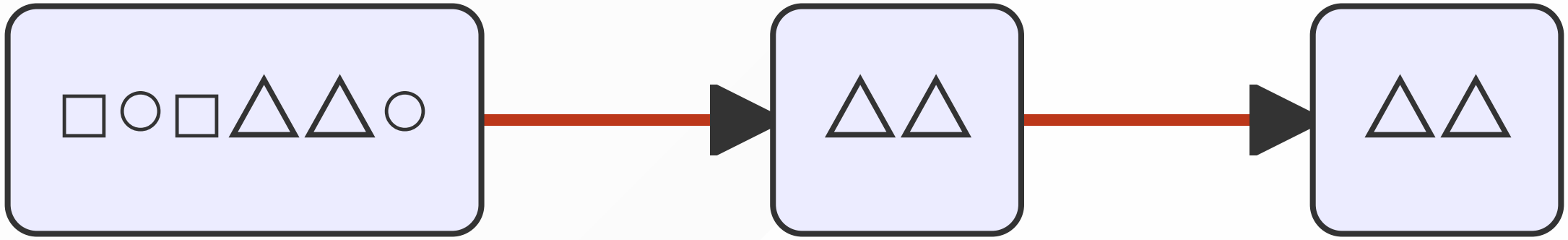


```
@Property
boolean commutativity(@ForAll int a, @ForAll int b) {
    return a + b == b + a ;
}
```

Invariance

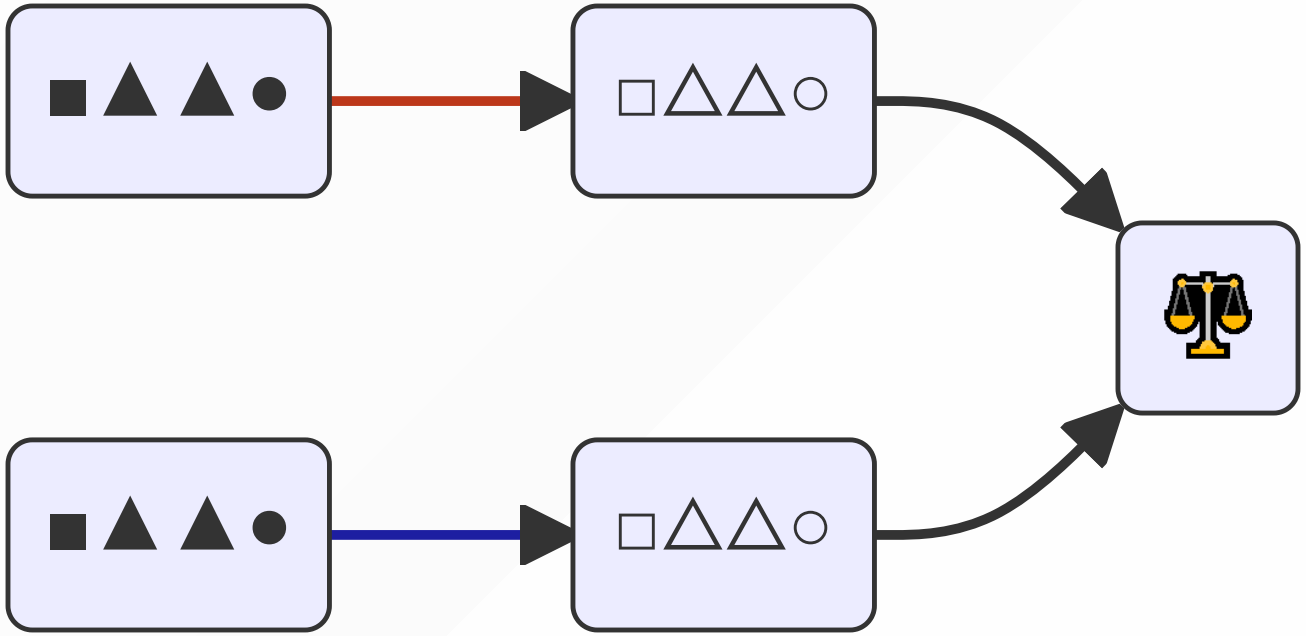
```
@Property
boolean invariant(@ForAll List<String> list) {
    int invariant = list.size();
    Collections.sort(list);
    return list.size() == invariant;
}
```

Idempotence



```
@Property
boolean idempotence(@ForAll String s) {
    return s.trim().equals(s.trim().trim());
}
```

Oracle




```
@Property
boolean oracle(@ForAll @StringLength(min = 3,max = 45) String s) {
    return myVeryFastCustomTrimFunction(s).equals(s.trim());
}
```

Tester la distributivité

```
// tests "classiques"
@Test void distributivityWith0() {assert 0 * (0+ 0) == 0*0+0*0;}
@Test void distributivityWith1() {assert 1 * (1+ 1) == 1*1+1*1;}
@Test void distributivityWith010() {assert  0 * (1+ 0) == 0*1+0*0;}
@Test void distributivityWith110() {assert   1 * (1+ 0) == 1*1+0*0;}
@Test void distributivityWithClassicNumber() {assert 42 * (42+ 42) == 42*42+42*42;}
@Test void distributivityWithBigNumber() {assert 100000 * (100000 + 100000) == 100000*100000+100000*100000;}
```

```
// Property Based Test
// @ForAll annotation qui va choisir aléatoirement un entier
@Property
boolean distributivity(@ForAll int a, @ForAll int b, @ForAll int c) {
    return a * (b+c) == a*b + a*c ;
}
```

Test de propriété fait à la main

```
private static Stream<Arguments> randomIntTierce() {
    Random random = new Random();
    return Stream.of(
        Arguments.of(random.nextInt(), random.nextInt(), random.nextInt()),
        Arguments.of(random.nextInt(), random.nextInt(), random.nextInt()),
        Arguments.of(random.nextInt(), random.nextInt(), null) // failing data
    );
}

@ParameterizedTest
@MethodSource("randomIntTierce")
void distributivity(Integer a, Integer b, Integer c) {
    assert a * (b+c) == a*b + a*c;
}
```

Pourquoi un Framework ?

- Ne pas réinventer la roue
- Générateur de données
  - Beaucoup de type primitifs et de cas limites
  - Configuration : itération, répartition des données
- Réduction - *Shrinking*
  - Capacité à trouver un cas simple de mise en échec d'un test
  - "ÜÀÀÖSSÀÄÖÖAÜÖAÄÜÄÜÖÜSSÖSSÖÄÀÖÖÀÄAÜÄÖAÖÄÀSSÜÀÜAÄAÀÀA"
  - => "Ä"

On en parle du métier ?

- Les interactions avec le métier selon la littérature :
  - *"Voici la nouvelle fonctionalité à implémenter, je vais vous présenter les cas d'usages avec nos personas, puis nous allons définir l'architecture de manière collégiale, rédiger les stories..."*
  - *"Voici les étapes et les informations pour repoduire ce bug ainsi que le comportement attendu"*
  - *"Voici la documentation exhaustive et à jour de tout le domaine fonctionnel de l'application"*

Mais la réalité c'est aussi ça

- Le Product Owner en discutant à la pause café :
  - *"On veut ouvrir l'application à notre nouveau partenaire, ça devrait marcher sans developement supplémentaire, tu confirmes ?"*
  - *"J'ai eu un client sans date de naissance en production, ça ne devrait pas arriver! Comment c'est possible ?"*
  - *"L'ancien Product Owner m'a certifié qu'on ne peut pas créer un compte sans code de parrainage mais je ne vois rien dans la documentation. Est ce que ça te dis quelquechose ?"*

Démonstration - Startup Bubble Factory

- 2019 : 💡🫧 Prototype de la machine à bulle de savon
- 2020 : 🪙🏭 Levée de fond puis lancement de la première usine
- 2021 : 📈🏬 Les ventes explosent, ouverture du centième magasin
- 2022 : 💰🦄 Nouvelle levée de fond et multiples embauches
- 2022 - 09: 👨‍💻👔 Le développeur principal vient d'être promu CTO
- 2022 - 10: 👨‍💻👩‍💼 Nouveau Product Owner et Développeur arrivent sur le projet de gestion des cartes de fidélités
- 2022 - 26: 👩‍💼🤝🤵Le PO a un point en urgence avec le CEO
  - ❓❓❓ Il en sort avec trois questions urgentes
  - 🔥🤦 Réunion en urgence avec le développeur

Les 3 questions

- Est-ce qu'on peut ouvrir notre service à des clients Allemands ?
- Est-il possible qu'un client n'ait pas de réduction ?
- Est-ce que les clients de Grenoble ont une réduction de 6% ?

Conclusion

- Il y a des propriétés dans mon code
- Il faut les tester ! Avec un framework !
- Voir le code comme un interlocuteur fonctionnel