

# Tests unitaires – Rappels et compléments



# Objectifs

- Objectifs et bonnes propriétés des tests unitaires
- Mock vs Stub

# Tests unitaires automatisés

- Tests unitaires automatisés aussi souvent appelés plus simplement tests unitaires sont :
  - Des tests qui permettent de valider un ensemble de cas de tests sur un constructeur, une méthode
- Cas de tests = description d'un cas
  - Données d'entrée : objet dans un état X et valeurs des paramètres de la méthode à appeler
  - Données attendues : valeur renvoyée par le méthode si non void et état de l'objet après l'appel
- Exemple : test de la méthode « Demarrer » de la classe Voiture

Données d'entrée	Données attendues
État : sans énergie	Erreur
État : déjà démarrée	Erreur
État : avec énergie, non démarrée	Démarrée

# Tests unitaires – Pourquoi ?

- Les tests fonctionnels sont couteux, difficilement répétables, connaissances couteuses
- Protection contre les régressions, à chaque ajout/modification, on ne reteste pas tout, on oublie des choses
- Force le découplage de code sinon ce n'est pas testable
- Le test peut servir de documentation sur les résultats attendus et la façon d'utiliser des méthodes

# Tests unitaires automatisés - AAA

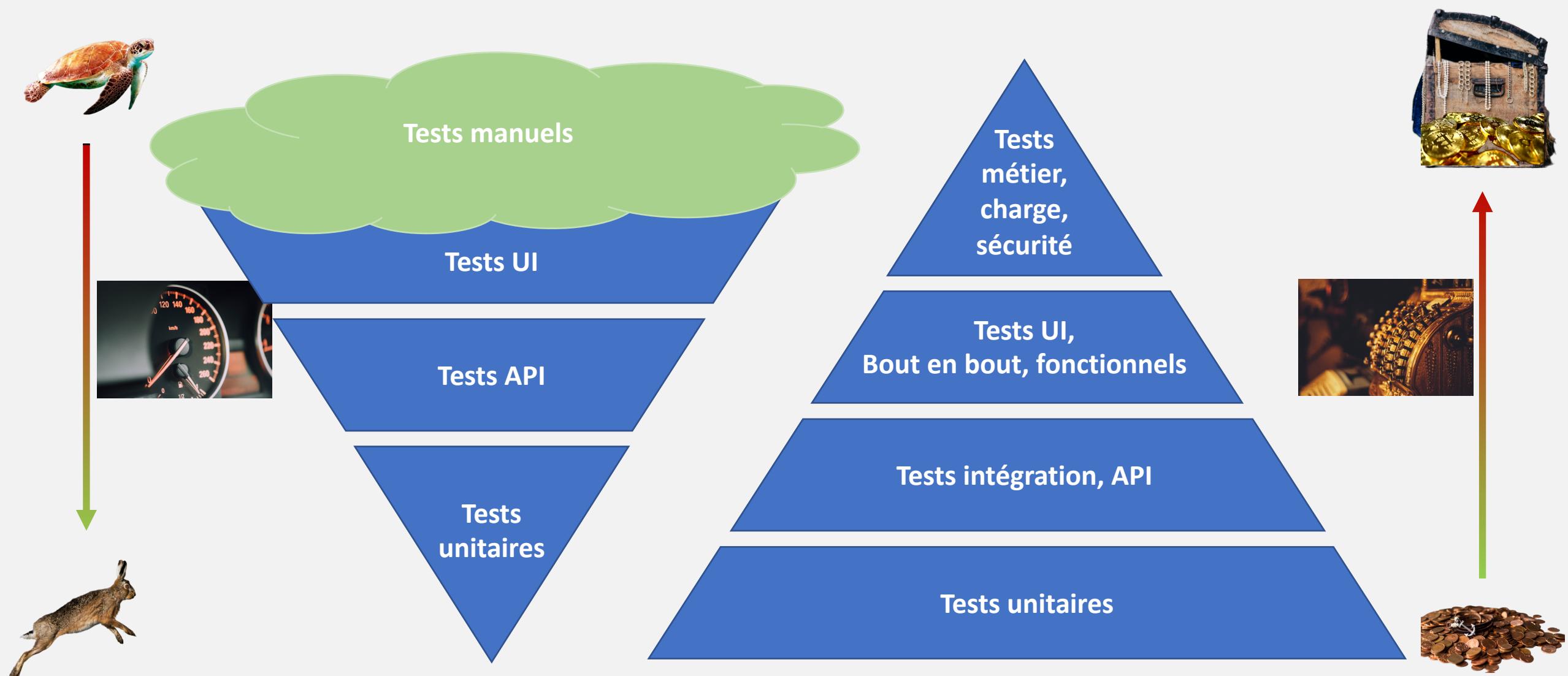
- Nous allons décrire nos tests en utilisant le découpage AAA :
  - **Arranger (Arrange)** : zone permettant de **décrire les valeurs initiales et les valeurs attendues et l'état initiale** de l'objet à tester
  - **Agir (Act)** : zone permettant **d'effectuer l'appel à la méthode** à tester
  - **Auditer (Assert)** : zone permettant de **valider**
    - les **valeurs attendues** sont bien **celles** que nous avons **obtenues** dans la zone « Agir »
    - **L'état de l'objet** après avoir appliqué la méthode
    - cette zone permet aussi de tester des **post-conditions** (exemple : le tableau est différent, etc.)

# Tests unitaires automatisés – Bonnes propriétés

Pour écrire des tests, Robert C. Martin indique qu'il faut respecter l'acronyme FIRST :

- **Fast (Rapide)** : Les tests doivent être rapides pour qu'ils soient lancés régulièrement
- **Independant (Indépendant)** : Les tests doivent pouvoir être lancés dans n'importe quel ordre
- **Repeatable (Reproductible)** : Les tests doivent pouvoir être reproduits dans n'importe quel environnement (votre poste de dev, la recette ou même la production si nécessaire)
- **Self-Validating (Auto validant)** : Les tests doivent avoir un résultat binaire (succès ou échec)
- **Timely (Au moment opportun)** : Les tests doivent être écrits avant le code de production. Si vous écrivez les tests après, vous remarquerez qu'il sera assez difficile de tester le code de production

# Tests – Manuels vs automatisation



# Tests unitaires – Stub / Mock

- Stub : remplacement d'une dépendance existante
- Mock : simule un objet du système qui **participe aux assertions**

```
var stubOrder = new FakeOrder();
var purchase = new Purchase(stubOrder);

purchase.ValidateOrders();

Assert.True(purchase.CanBeShipped);
```

```
var mockOrder = new FakeOrder();
var purchase = new Purchase(mockOrder);

purchase.ValidateOrders();

Assert.True(mockOrder.Validated);
```

# Tests unitaires - Outils

- Nous allons utiliser xUnit pour effectuer les tests
- Nous allons utiliser Moq pour créer nos Stub/Mock
- Vous allez pouvoir aussi utiliser Fluent Assertion pour faciliter la lecture des tests automatisés
- Pour installer Moq et Fluent Assertion, utilisez le gestionnaire de packages Nuget
  - À partir de l'explorateur de solution, faites un clic droit sur le projet et choisissez « Gérer les packages NuGet... »

# Exemple Moq

```
public class Questionnaire
{
    public List<IQuestion> Questions { get; private set; }
    public int TotalPoints {
        get {
            return this.Questions.Sum(q => q.NombrePointsTotal);
        }
    }
    public int Score { get; private set; }

    // ...
}
```

```
public interface IQuestion
{
    public int NombrePointsTotal { get; }
    public void PoserQuestion();
    public int CorrigerReponse();
}
```

# Exemple Moq

- Crédation d'un mock de l'interface IQuestion

```
Mock<IQuestion> mockQuestion1Point = new Mock<IQuestion>();
```

- Configuration de la méthode CorrigerReponse, elle va renvoyer tout le temps 1

```
mockQuestion1Point.Setup(q => q.CorrigerReponse()).Returns(1);
```

- Configuration de la propriété NombrePointsTotal, elle va renvoyer tout le temps 1

```
mockQuestion1Point.SetupGet(q => q.NombrePointsTotal).Returns(1);
```

# Exemple Moq

- Récupération de l'instance configurée

```
mockQuestion1Point.Object
```

- Validations que la méthode a été appelée une seule fois

```
mockQuestion1Point.Verify(q => q.CorrigerReponse(), Times.Once);
```

- Validations que la propriété a été appelée une seule fois

```
mockQuestion1Point.VerifyGet(q => q.NombrePointsTotal, Times.Once);
```

# Exemple Moq

```
[Fact]
public void TotalPoints_ListeAvecQuestions_SommePoints()
{
    // Arranger
    Mock<IQuestion> mockQuestion1Point = new Mock<IQuestion>();
    mockQuestion1Point.SetupGet(q => q.NombrePointsTotal).Returns(1);
    Mock<IQuestion> mockQuestion2Point = new Mock<IQuestion>();
    mockQuestion2Point.SetupGet(q => q.NombrePointsTotal).Returns(2);
    Mock<IQuestion> mockQuestion3Point = new Mock<IQuestion>();
    mockQuestion3Point.SetupGet(q => q.NombrePointsTotal).Returns(4);

    List<IQuestion> questions = new List<IQuestion>()
    {
        mockQuestion1Point.Object,
        mockQuestion2Point.Object,
        mockQuestion3Point.Object
    };
    Questionnaire questionnaire = new Questionnaire(questions);
    int totalPointsAttendu = 7;

    // Agir
    int totalPointsCalcule = questionnaire.TotalPoints;

    // Assert
    Assert.Equal(totalPointsAttendu, totalPointsCalcule);
    mockQuestion1Point.VerifyGet(q => q.NombrePointsTotal, Times.Once);
    mockQuestion2Point.VerifyGet(q => q.NombrePointsTotal, Times.Once);
    mockQuestion3Point.VerifyGet(q => q.NombrePointsTotal, Times.Once);
}
```

# Exemple Moq

```
[Fact]
public void CorrigerQuestions_QuestionnaireEffectue_SommeScore()
{
    // Arranger
    Mock<IQuestion> mockQuestion1Point = new Mock<IQuestion>();
    mockQuestion1Point.Setup(q => q.CorrigerReponse()).Returns(1);
    Mock<IQuestion> mockQuestion2Point = new Mock<IQuestion>();
    mockQuestion2Point.Setup(q => q.CorrigerReponse()).Returns(2);
    Mock<IQuestion> mockQuestion3Point = new Mock<IQuestion>();
    mockQuestion3Point.Setup(q => q.CorrigerReponse()).Returns(4);

    List<IQuestion> questions = new List<IQuestion>()
    {
        mockQuestion1Point.Object,
        mockQuestion2Point.Object,
        mockQuestion3Point.Object
    };
    Questionnaire questionnaire = new Questionnaire(questions);
    int scoreCalculeAttendu = 7;

    // Agir
    questionnaire.CorrigerQuestions();

    // Assert
    Assert.Equal(scoreCalculeAttendu, questionnaire.Score);
    mockQuestion1Point.Verify(q => q.CorrigerReponse(), Times.Once);
    mockQuestion1Point.VerifyNoOtherCalls();

    mockQuestion2Point.Verify(q => q.CorrigerReponse(), Times.Once);
    mockQuestion2Point.VerifyNoOtherCalls();

    mockQuestion3Point.Verify(q => q.CorrigerReponse(), Times.Once);
    mockQuestion3Point.VerifyNoOtherCalls();
```

# Exemple Fluent assertion

```
[Fact]
public void Ctor_ListeNulleVFluentAssertion_Exception()
{
    // Arranger
    List<IQuestion> questions = null;
    string nomParametreEnErreur = "p_listeQuestions";
    Action act = () => new Questionnaire(questions);

    // Agir && Assert
    act.Should().Throw<ArgumentNullException>()
        .And.ParamName.Should().Be(nomParametreEnErreur);
}
```

# Exemple Fluent assertion

```
[Fact]
public void
CorrigerQuestions_QuestionnaireEffectueFluentAssertion_SommePoints()
{
    // Arranger
    Mock<IQuestion> mockQuestion1Point = new Mock<IQuestion>();
    mockQuestion1Point.Setup(q => q.CorrigerReponse()).Returns(1);
    Mock<IQuestion> mockQuestion2Point = new Mock<IQuestion>();
    mockQuestion2Point.Setup(q => q.CorrigerReponse()).Returns(2);
    Mock<IQuestion> mockQuestion3Point = new Mock<IQuestion>();
    mockQuestion3Point.Setup(q => q.CorrigerReponse()).Returns(4);

    List<IQuestion> questions = new List<IQuestion>()
    {
        mockQuestion1Point.Object,
        mockQuestion2Point.Object,
        mockQuestion3Point.Object
    };
}
```

```
Questionnaire questionnaire = new Questionnaire(questions);
int scoreCalculeAttendu = 7;

// Agir
questionnaire.CorrigerQuestions();

// Assert
questionnaire.Score.Should().Be(scoreCalculeAttendu);
mockQuestion1Point.Verify(q => q.CorrigerReponse(), Times.Once);
mockQuestion1Point.VerifyNoOtherCalls();

mockQuestion2Point.Verify(q => q.CorrigerReponse(), Times.Once);
mockQuestion2Point.VerifyNoOtherCalls();

mockQuestion3Point.Verify(q => q.CorrigerReponse(), Times.Once);
mockQuestion3Point.VerifyNoOtherCalls();
}
```

# Références

- Aller plus loin avec Moq :  
<https://github.com/Moq/moq4/wiki/Quickstart>
- Fluent assertions – facilitez la lecture de vos cas de tests :  
<https://fluentassertions.com/introduction>