Ep2 : Cucumber

*crée par **Yongkang Zou** et **Yassine Kafraoui***

Récapitulatif de l'épisode 1 :

L'histoire se déroule dans une ville animée, où se trouve un concessionnaire automobile nommé **Java Auto**, dirigé par un passionné de programmation et d'automobiles du nom de Jeff. Java Auto se distingue par la vente de voitures de haute qualité et de systèmes audio innovants pour automobiles.

Jeff décide d'incorporer sa profonde compréhension de Java dans le processus de vente de voitures. Il crée une classe Java appelée **"Voiture"**, comprenant des attributs tels que **"Prix"** et **"Marque"**. En même temps, pour ajouter des fonctionnalités uniques au système audio des voitures, il crée une classe appelée **"AudioSystem"**. En utilisant

habilement les classes Java, Java Auto devient l'un des concessionnaires automobiles les plus populaires de la ville.

Quelques années plus tard, Java Auto prospère toujours, et le fils de Jeff, Jack, développe un intérêt pour le magasin de voitures de son père et aspire à devenir programmeur. Jeff commence à enseigner à Jack comment utiliser BlueJ pour créer sa première classe Java, "Voiture", et à définir des propriétés, créer des constructeurs, et implémenter des méthodes telles que **"promoVoiture"**.

Jeff souligne l'importance des tests et encourage Jack à adopter la bonne habitude de tester immédiatement après la création des méthodes. Ils créent également une deuxième classe, **"AudioSystem"**, en tant que complément aux voitures, permettant aux clients de personnaliser leurs systèmes audio de voiture.

À la fin de la journée, Jeff montre à Jack comment tester la méthode **"PrixBlackFriday"** qu'ils ont créée pour offrir des remises pendant le vendredi noir. Jeff félicite Jack pour son excellent travail et décide de poursuivre l'enseignement le lendemain.

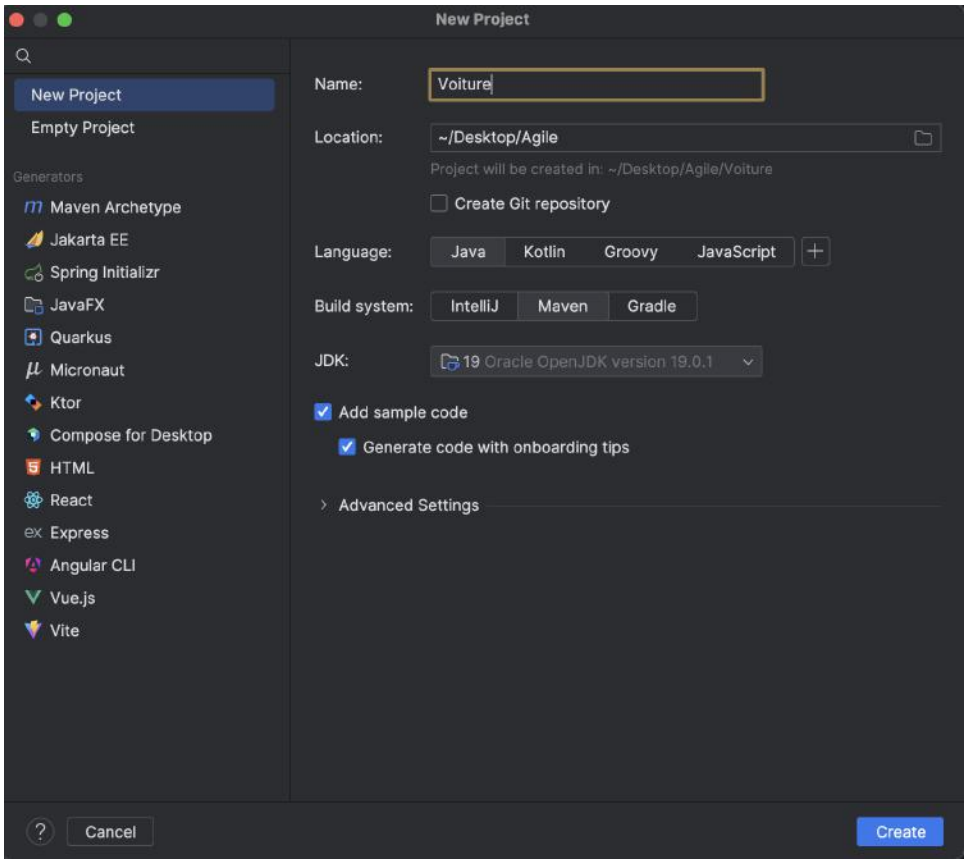
EP2 : Intégration Agile

Le deuxième jour commence, et Jeff décide de présenter en détail le processus de création de Java Auto à Jack.

Étape 1 : Création d'un projet Maven dans IntelliJ et déplacement des classes "Voiture" et "AudioSystem" dans BlueJ.

"Hier, je t'ai appris à utiliser **BlueJ**, tu as bien appris ?" demande Jeff. "C'est trop facile, c'est juste ça, j'ai déjà tout compris," répond joyeusement Jack. "Haha, ne sois pas trop fier, **BlueJ** n'est qu'un outil pour les débutants en apprentissage de la programmation Java. Maintenant, je vais t'apprendre comment continuer le développement de nos programmes Java dans un environnement de développement intégré (IDE) plus professionnel, **IntelliJ IDEA**," dit Jeff en caressant la tête de Jack.

"C'est du gâteau," dit Jack, et avant même que Jeff puisse dire quelque chose de plus, Jack crée déjà un nouveau projet **"Voiture"** dans **IntelliJ**.

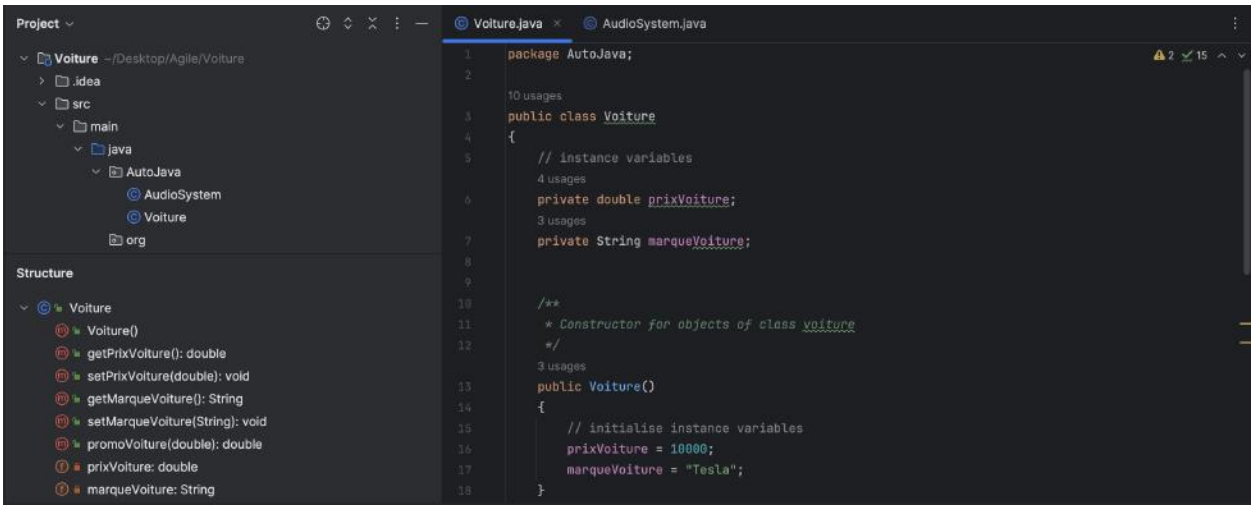


"Rappelle-toi de choisir le **système de construction (Building system)** comme **Maven**," rappelle Jeff à Jack avant que ce dernier ne clique sur le bouton **"Create"**.

"Ah, c'est quoi **Maven** ?" demande Jack en posant la question tout en créant déjà le projet **Maven**.

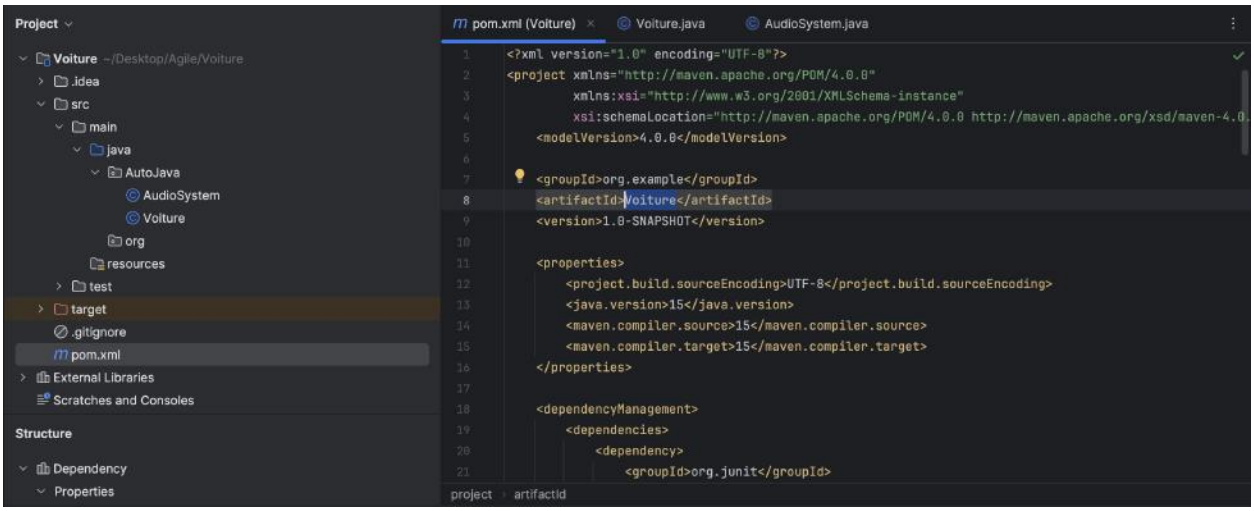
"**Maven**, par rapport à un simple projet **IntelliJ**, nous permet de travailler plus facilement avec le framework **Cucumber**," explique Jeff.

"D'accord," dit Jack, et avant même que Jeff puisse dire quelque chose de plus, Jack crée déjà un package **"Auto Java"** dans le dossier src/main de son nouveau projet **"Voiture"**. Ensuite, il importe les fichiers JAR des classes **"Voiture"** et **"AudioSystem"** dans ce package.



Étape 2 : Configuration du projet Maven, ajout des dépendances Cucumber et JUnit.

"Tu es plutôt rapide," dit Jeff en souriant, "mais avant de commencer réellement à travailler, nous devons ajouter les dépendances **Cucumber** et **JUnit** dans le fichier **pom.xml** de notre projet **Maven.**"



Sous la direction de Jeff, Jack a rapidement terminé l'édition du fichier **pom.xml**.

Étape 3 : Comprendre le concept des f**ichiers .feature** et des **classes Step Definition**.

"Jusqu'à présent, tu ne sais toujours pas ce qu'est **Cucumber**, n'est-ce pas ?" demande Jeff.

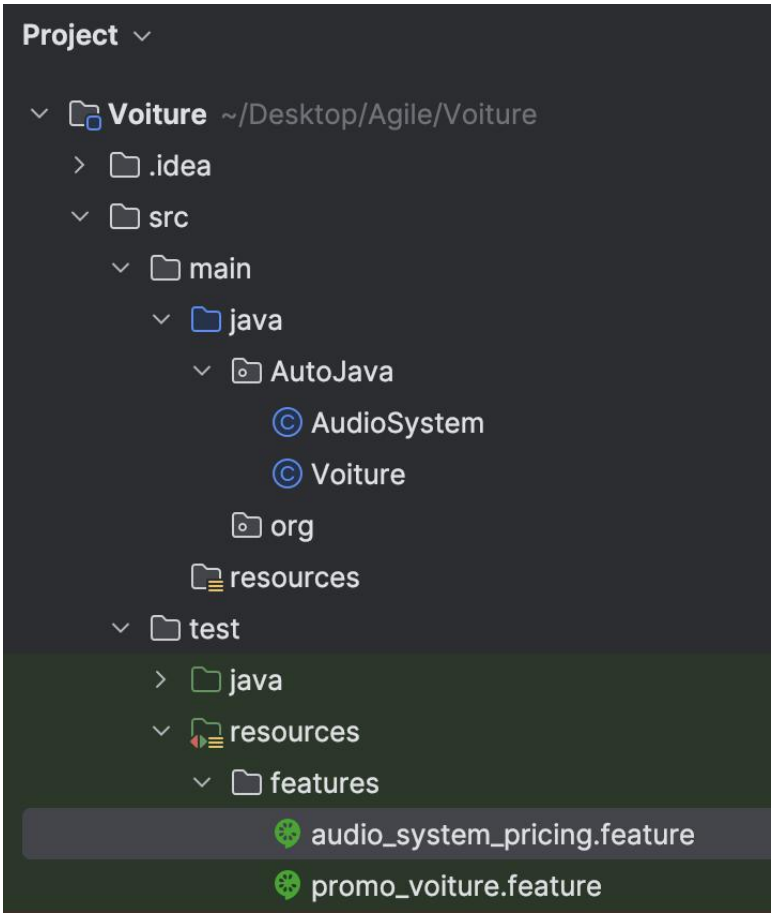
"Je sais, cucumber, c'est le concombre 🥒," répond Jack en riant.

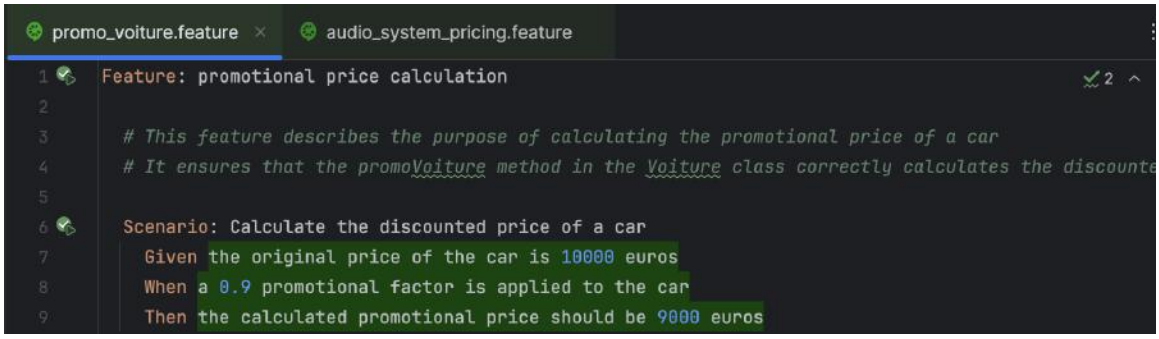
"Hahaha," Jeff éclate de rire face à la réponse innocente de Jack. "Le **framework Cucumber** est un outil de ***développement piloté par le comportement (BDD)*** utilisé pour écrire des tests plus lisibles. Dans le cadre de Cucumber, nous devons d'abord rédiger un fichier `.feature`, qui contient la description en langage naturel des scénarios utilisateur, ainsi qu'une classe `Step Definition` pour mettre en œuvre les étapes de test liées aux scénarios utilisateur."

Étape 4 : Créer un fichier `.feature`.

"C'est un concept un peu abstrait, j'ai l'impression de ne pas bien comprendre," dit Jack en regardant Jeff avec des yeux perplexes.

"Peut-être qu'un exemple rendra les choses plus claires. Laisse-moi d'abord t'enseigner comment créer ton premier fichier `.feature`," dit Jeff avec bienveillance.





"En décrivant le processus d'un **scénario**, nous avons progressivement étendu les déclarations **'Given' 'When' 'Then'**, éclaircissant ainsi notre logique d'acceptation couche par couche."

"Oh, d'accord." Jack hoche la tête, semblant comprendre vaguement. "J'ai l'impression de commencer à comprendre."

"La méthode **'Scenario'** nous permet de tester un ensemble de données lors de l'acceptation, mais nous pouvons également utiliser une méthode similaire appelée

'Scenario Outline' pour tester plusieurs ensembles de données.”

```
Scenario Outline: Calculate the discounted price of a car
  Given the original price of the car is <original_price> euros
  When a <discount_factor> promotional factor is applied to the car
  Then the calculated promotional price should be <promotional_price> euros

  Examples:
    | original_price | discount_factor | promotional_price |
    | 10000          | 0.9             | 9000              |
    | 20000          | 0.85            | 17000             |
    | 15000          | 0.8             | 12000             |
```

"Ah, d'accord, je commence à comprendre un peu," acquiesce Jack en inclinant la tête.

promo_voiture.feature | audio_system_pricing.feature

```
Feature: US_001: Calculating the Promotional Price of a Car
  As a car sales manager,
  I want to be able to calculate the promotional price of a car under different promotional factors,
  So that I can provide accurate discount information to customers and facilitate sales.

  # This feature describes the purpose of calculating the promotional price of a car
  # It ensures that the promoVoiture method in the Voiture class correctly calculates the discounted

  Scenario: Calculate the discounted price of a car
    Given the original price of the car is 10000 euros
    When a 0.9 promotional factor is applied to the car
    Then the calculated promotional price should be 9000 euros

  Scenario Outline: Calculate the discounted price of a car
    Given the original price of the car is <original_price> euros
    When a <discount_factor> promotional factor is applied to the car
    Then the calculated promotional price should be <promotional_price> euros

  Examples:
      | original_price | discount_factor | promotional_price |
      | 10000          | 0.9             | 9000              |
      | 20000          | 0.85            | 17000             |
      | 15000          | 0.8             | 12000             |
```

"De plus, lors de la description initiale du feature, nous pouvons directement utiliser le style de l'**histoire utilisateur (User Story)**. Ainsi, même les personnes comme les

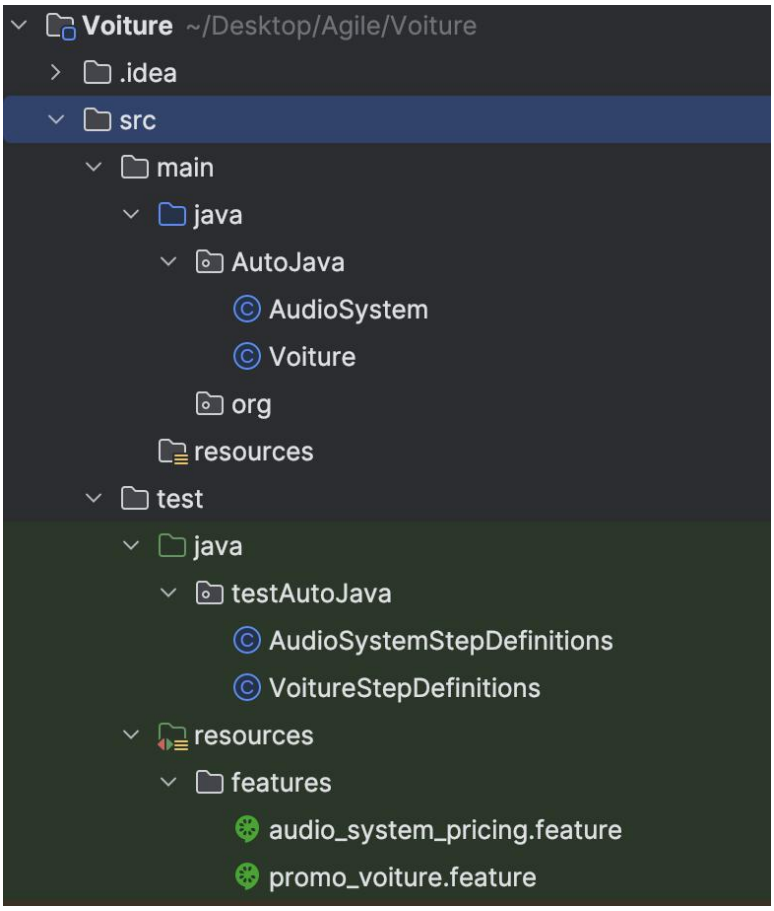
analystes métier qui ne comprennent pas très bien le code peuvent rapidement comprendre notre code de test."

"D'accord, maintenant je comprends," dit Jack.

Étape 5 : Créer la classe `Step Definition`.

"Ce n'est pas encore fini, le code ne peut pas encore être exécuté. Nous devons créer une classe `Step Definition` correspondant à ce fichier `.feature`."

"😵"Jack commence à se sentir étourdi à nouveau.



"Commençons par créer un package `testAuto` pour stocker nos classes `Step Definition`," dit Jeff patiemment en regardant Jack.

"Ensuite, en correspondance avec les déclarations 'given', 'when', et 'then' que nous venons d'écrire dans le fichier `.feature`, connectons progressivement l'histoire utilisateur aux classes Java réelles."

"Je commence à comprendre, laissez-moi essayer d'écrire le fichier `.feature` correspondant à AudioSystem et la classe `Step Definition`." Jack regagne un peu de confiance et commence à écrire en parlant.

```
Feature: US_101 Calculating Promotional Price of Audio System
  As an audio system salesperson,
  I want to be able to calculate the promotional price of the audio system with different promotional factors,
  So that I can offer attractive deals to customers and stimulate sales.

  Scenario: Calculating promotional price of an audio system
    Given the original price of the audio system is 5000 euros
    When a 0.9 promotional factor is applied to the audio system
    Then the promotional price of the audio system should be 4500 euros

  Scenario Outline: Calculating promotional price of an audio system
    Given the original price of the audio system is <audio_price> euros
    When a <promo_factor> promotional factor is applied to the audio system
    Then the promotional price of the audio system should be <promo_audio_price> euros

    Examples:
      | audio_price | promo_factor | promo_audio_price |
      | 5000        | 0.9          | 4500              |
      | 6000        | 0.85         | 5100              |
      | 7000        | 0.8          | 5600              |
```

```
Feature: US_102 Calculating the Total Price of the Car and Audio System
  As a sales manager for car and audio system combos,
  I want to be able to calculate the total price when buying a car and an audio system,
  So that I can provide clear pricing information to customers, facilitating their purchase decision.

  Scenario: Calculating total price of car and audio system
    Given the price of the AudioSystem's car is 10000 euros
    And the price of the audio system is 5000 euros
    When I calculate the sum of both prices
    Then the total price should be 15000 euros

  Scenario Outline: Calculating promotional price of an audio system
    Given the original price of the audio system is <audio_price> euros
    When a <promo_factor> promotional factor is applied to the audio system
    Then the promotional price of the audio system should be <promo_audio_price> euros

    Examples:
      | audio_price | promo_factor | promo_audio_price |
      | 5000        | 0.9          | 4500              |
      | 6000        | 0.85         | 5100              |
      | 7000        | 0.8          | 5600              |
```

```
Feature:US_103 Calculating Black Friday Price for Car and Audio System
  As a promotions manager,
  I want to offer additional discounts to customers purchasing a car and an audio system during Black Friday,
  So that I can boost sales and attract more customers.


  Scenario: Calculating Black Friday price for car and audio system
    Given the price of the AudioSystem's car is 10000 euros
    And the price of the audio system is 5000 euros
    When a 0.8 Black Friday promotional factor is applied
    Then the Black Friday total price should be 12000 euros

  Scenario Outline: Calculating Black Friday price for car and audio system
    Given the price of the AudioSystem's car is <car_price> euros
    And the price of the audio system is <audio_price> euros
    When a <bf_promo_factor> Black Friday promotional factor is applied
    Then the Black Friday total price should be <bf_total_price> euros

    Examples:
      | car_price | audio_price | bf_promo_factor | bf_total_price |
      | 10000     | 5000        | 0.8             | 12000          |
      | 12000     | 6000        | 0.75            | 13500          |
      | 15000     | 4000        | 0.7             | 13300          |
```

"Pas mal du tout." Jeff regarde avec satisfaction alors que Jack termine rapidement l'écriture du fichier `.feature` pour AudioSystem. "Maintenant, commençons à écrire la classe `Step Definition`."

```java
@Given("the original price of the audio system is {double} euros")
public void the_original_price_of_the_audio_system_is(double price) {
    audioSystem = new AudioSystem();
    audioSystem.setPrixAudio(price);
}

@Given("the price of the audio system is {double} euros")
public void the_price_of_the_audio_system_is(double price) {
    audioSystem = new AudioSystem();
    audioSystem.setPrixAudio(price);
}

@Given("the price of the AudioSystem's car is {double} euros")
public void the_price_of_the_audio_system_car_is(double price) {
    voiture = new Voiture();
    voiture.setPrixVoiture(price);
}

@When("a {double} promotional factor is applied to the audio system")
public void a_promotional_factor_is_applied_to_the_audio_system(double promo) {
    result = audioSystem.promoAudio(promo);
}

@When("I calculate the sum of both prices")
public void i_calculate_the_sum_of_both_prices() { result = audioSystem.sumPrix(voiture); }

@When("a {double} Black Friday promotional factor is applied")
public void a_black_friday_promotional_factor_is_applied(double promo) {
    result = audioSystem.PrixBlackFriday(voiture, promo);
}
```

```java
@Then("the promotional price of the audio system should be {double} euros")
public void the_promotional_price_of_the_audio_system_should_be(double expectedPrice) {
    assertEquals(expectedPrice, result, delta: 0.01);
}

@Then("the total price should be {double} euros")
public void the_total_price_should_be(double expectedPrice) { assertEquals(expectedPrice, result, delta: 0.01); }

@Then("the Black Friday total price should be {double} euros")
public void the_black_friday_total_price_should_be(double expectedPrice) {
    assertEquals(expectedPrice, result, delta: 0.01);
}
```

Étape 6 : Exécuter le test.

"Hmm, maintenant nous pouvons exécuter le test." Jeff regarde Jack qui vient de terminer la classe `Step Definition`.

```
✓ Done: Scenarios 4 of 4  (968 ms)  ⚠
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
Testing started at 23:54 ...

4 Scenarios (4 passed)
12 Steps (12 passed)
0m0.260s
```

"Vraiment bien, nous avons pratiquement terminé la mise en place de l'intégration agile, et il est presque l'heure. Nous nous arrêterons ici pour aujourd'hui." En regardant Jack déjà endormi, Jeff dit doucement, "Continuons notre apprentissage demain."