

# README CINEMA PROJECT - JAVA

Réalisation par :

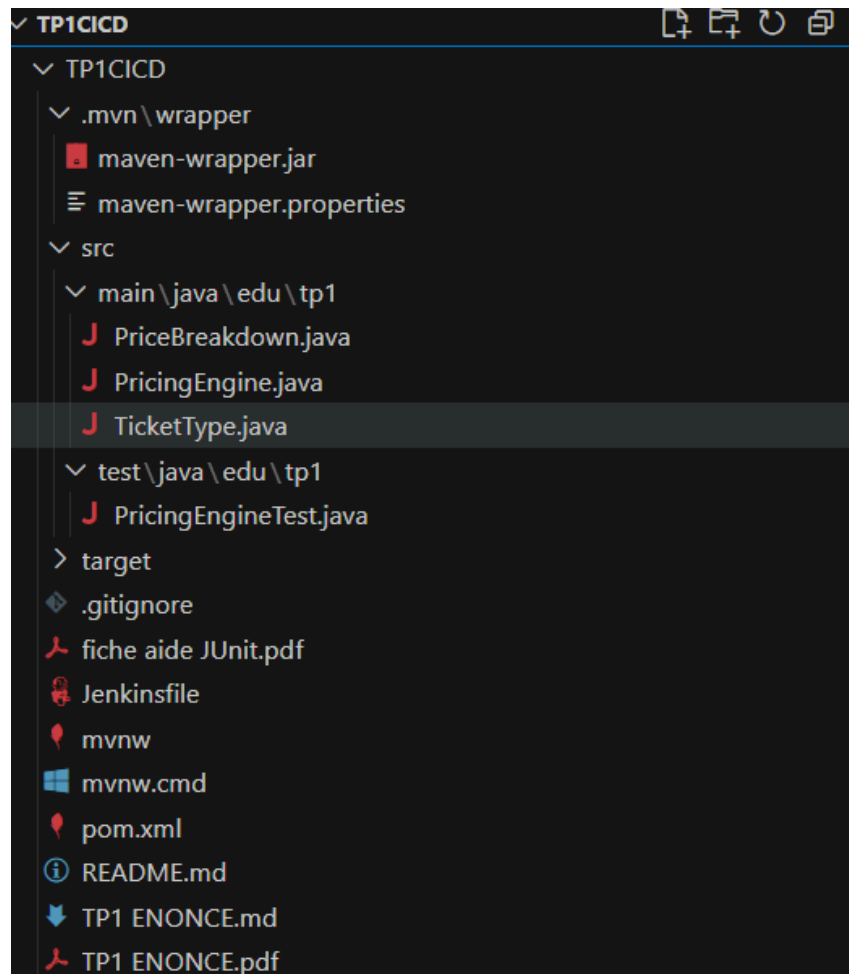
Floyd LAFORGE	Nathan VANDENBOSSCHE
Crystal BOUDAUD	Maxime LANGRAND

## README :

Projet Mini Moteur de Tarification Cinéma :

Ce projet a pour objectif de développer un module Java pour calculer le prix total d'une commande de billets de cinéma selon différentes règles tarifaires. Les critères pris en compte sont le type de billet (adulte, enfant, sénior, étudiant), l'utilisation éventuelle de la 3D, le jour de la séance (avec une remise spécifique le mercredi) et l'achat en groupe (à partir de 4 billets). Ce moteur de tarification doit être structuré avec un code clair, testé avec JUnit 5 et permettre la mesure de la couverture de tests via JaCoCo. Une intégration avec Jenkins via Maven est prévue en option.

Arborescence :



## Fichiers Java et leurs rôles :

### 1. TicketType.java

Ce fichier définit une énumération qui représente les différents types de billets disponibles : ADULT, CHILD, SENIOR, STUDENT. Chaque valeur correspond à un type spécifique de ticket.

```
TP1CICD > src > main > java > edu > tp1 > J TicketType.java > {} edu.tp1
1  package edu.tp1;
2
3  public enum TicketType {
4      ADULT(price:10.00),
5      CHILD(price:6.00),
6      SENIOR(price:7.50),
7      STUDENT(price:8.00);
8
9      private final double price;
10
11     TicketType(double price) {
12         this.price = price;
13     }
14
15     public double getPrice() {
16         return price;
17     }
18 }
19
```

## README CINEMA PROJECT - JAVA

### 2. PriceBreakdown.java

Cette classe encapsule le résultat détaillé du calcul du prix d'une commande. Elle contient :

subtotal : la somme des prix de base sans remises ni suppléments.

wednesdayDisc : la remise appliquée si la séance est le mercredi (-20%).

threeDSurcharge : le supplément appliqué par billet si la séance est en 3D (+2€ par billet).

groupDisc : la remise de groupe si 4 billets ou plus (-10€).

total : le montant final, arrondi au centime.

Cette structure permet de présenter un récapitulatif clair et détaillé du calcul du prix.

```
TP1CICD > src > main > java > edu > tp1 > PriceBreakdown.java > {} edu.tp1
1 package edu.tp1;
2
3 import java.util.StringJoiner;
4
5 public final class PriceBreakdown {
6     private final double subtotal;
7     private final double wednesdayDiscount;
8     private final double threeDSurcharge;
9     private final double groupDiscount;
10    private final double total;
11
12    public PriceBreakdown(double subtotal, double wednesdayDiscount, double threeDSurcharge, double groupDiscount,
13        this.subtotal = subtotal;
14        this.wednesdayDiscount = wednesdayDiscount;
15        this.threeDSurcharge = threeDSurcharge;
16        this.groupDiscount = groupDiscount;
17        this.total = total;
18    }
19
20    public double getSubtotal() { return subtotal; }
21    public double getWednesdayDiscount() { return wednesdayDiscount; }
22    public double getThreeDSurcharge() { return threeDSurcharge; }
23    public double getGroupDiscount() { return groupDiscount; }
24    public double getTotal() { return total; }
25
26    @Override
27    public String toString() {
28        return new StringJoiner(delimiter:", ", PriceBreakdown.class.getSimpleName() + "[", suffix:"]")
29            .add("subtotal=" + subtotal)
30            .add("wednesdayDiscount=" + wednesdayDiscount)
31            .add("threeDSurcharge=" + threeDSurcharge)
32            .add("groupDiscount=" + groupDiscount)
33            .add("total=" + total)
34            .toString();
35    }
36 }
```

## README CINEMA PROJECT - JAVA

### 3. PricingEngine.java

Il s'agit du cœur du projet. Cette classe contient :

La méthode `basePrice(TicketType)` qui retourne le prix de base pour chaque type de billet.

La méthode `computeTotal(List<TicketType>, boolean, DayOfWeek)` qui calcule le prix total en appliquant dans l'ordre :

Calcul du prix total sans remises ni suppléments (subtotal).

Application de la remise du mercredi (-20% si applicable).

Ajout du supplément 3D (+2€ par billet si séance 3D).

Application de la remise groupe (-10% si 4 billets ou plus).

Un arrondi systématique des montants au centime près.

Cette organisation garantit la lisibilité, la maintenabilité et la possibilité d'ajouter facilement des règles tarifaires.

```
TP1CICD > src > main > java > edu > tp1 > PricingEngine.java > {} edu.tp1
1 package edu.tp1;
2
3 import java.math.BigDecimal;
4 import java.math.RoundingMode;
5 import java.time.DayOfWeek;
6 import java.util.List;
7
8 public class PricingEngine {
9
10     public double basePrice(TicketType type) {
11         if (type == null) {
12             throw new IllegalArgumentException(s:"TicketType cannot be null.");
13         }
14         return type.getPrice();
15     }
16
17     public PriceBreakdown computeTotal(List<TicketType> tickets, DayOfWeek day, boolean is3D) {
18         if (tickets == null || day == null) {
19             throw new IllegalArgumentException(s:"Tickets list and day cannot be null.");
20         }
21
22         double subtotal = tickets.stream()
23             .mapToDouble(this::basePrice)
24             .sum();
25
26         double priceAfterWednesdayDiscount = subtotal;
27         double wednesdayDiscount = 0;
28         if (day == DayOfWeek.WEDNESDAY) {
29             wednesdayDiscount = roundToCents(subtotal * 0.20);
30             priceAfterWednesdayDiscount = subtotal - wednesdayDiscount;
31         }
32
33         double threeDSurcharge = 0;
34         if (is3D) {
35             threeDSurcharge = tickets.size() * 2.00;
36         }
37         double priceAfter3D = priceAfterWednesdayDiscount + threeDSurcharge;
```

## README CINEMA PROJECT - JAVA

```
38
39     double groupDiscount = 0;
40     if (tickets.size() >= 4) {
41         groupDiscount = roundToCents(priceAfter3D * 0.10);
42     }
43
44     double total = priceAfter3D - groupDiscount;
45
46     return new PriceBreakdown(
47         roundToCents(subtotal),
48         wednesdayDiscount,
49         threeDSurcharge,
50         groupDiscount,
51         roundToCents(total)
52     );
53 }
54
55 private double roundToCents(double value) {
56     return BigDecimal.valueOf(value).setScale(newScale:2, RoundingMode.HALF_UP).doubleValue();
57 }
58 }
59
```

## README CINEMA PROJECT - JAVA

### 4. PricingEngineTest.java

Ce fichier fournit une suite de tests unitaires avec JUnit 5 pour vérifier le bon fonctionnement de la classe PricingEngine. Les tests couvrent :

La validation des prix de base selon les types de billets.

Le comportement du moteur pour un panier vide.

L'application isolée des différentes remises et suppléments (mercredi, 3D, groupe).

La gestion combinée de plusieurs règles tarifaires.

La gestion des erreurs, notamment la présence d'un type de billet nul.

```
TP1CICD > src > test > java > edu > tp1 > J PricingEngineTest.java > {} edu.tp1
1  package edu.tp1;
2
3  import java.time.DayOfWeek;
4  import java.util.Arrays;
5  import java.util.Collections;
6  import java.util.List;
7  import java.util.stream.Collectors;
8
9  import static org.junit.jupiter.api.Assertions.assertEquals;
10 import static org.junit.jupiter.api.Assertions.assertThrows;
11 import org.junit.jupiter.api.DisplayName;
12 import org.junit.jupiter.api.Test;
13 import org.junit.jupiter.params.ParameterizedTest;
14 import org.junit.jupiter.params.provider.CsvSource;
15 import org.junit.jupiter.params.provider.EnumSource;
16
17 class PricingEngineTest {
18
19     private final PricingEngine engine = new PricingEngine();
20
21     @ParameterizedTest
22     @EnumSource(TicketType.class)
23     @DisplayName("basePrice() should return correct price for each ticket type")
24     void testBasePrice(TicketType type) {
25         assertEquals(type.getPrice(), engine.basePrice(type));
26     }
27
28     @Test
29     @DisplayName("basePrice() should throw exception for null type")
30     void testBasePriceNull() {
31         assertThrows(IllegalArgumentException.class, () -> engine.basePrice(type:null));
32     }
33
34     @Test
35     @DisplayName("computeTotal() should return 0.00 for empty basket")
36     void testEmptyBasket() {
37         PriceBreakdown breakdown = engine.computeTotal(Collections.emptyList(), DayOfWeek.MONDAY, is3D:false);
```

## README CINEMA PROJECT - JAVA

```
38         assertEquals(0.00, breakdown.getTotal());
39     }
40
41     @Test
42     @DisplayName("computeTotal() with Wednesday discount only")
43     void testWednesdayDiscount() {
44         List<TicketType> tickets = List.of(TicketType.ADULT, TicketType.SENIOR, TicketType.CHILD);
45         PriceBreakdown breakdown = engine.computeTotal(tickets, DayOfWeek.WEDNESDAY, is3D:false);
46         assertEquals(18.80, breakdown.getTotal());
47     }
48
49     @Test
50     @DisplayName("computeTotal() with 3D surcharge only")
51     void test3DSurcharge() {
52         List<TicketType> tickets = List.of(TicketType.ADULT, TicketType.CHILD);
53         PriceBreakdown breakdown = engine.computeTotal(tickets, DayOfWeek.MONDAY, is3D:true);
54         assertEquals(20.00, breakdown.getTotal());
55     }
56
57     @Test
58     @DisplayName("computeTotal() with Group discount only")
59     void testGroupDiscount() {
60         List<TicketType> tickets = Collections.nCopies(n:4, TicketType.STUDENT);
61         PriceBreakdown breakdown = engine.computeTotal(tickets, DayOfWeek.MONDAY, is3D:false);
62         assertEquals(28.80, breakdown.getTotal());
63     }
64
65     @ParameterizedTest
66     @CsvSource({
67         // On utilise des apostrophes pour que la liste de tickets soit lue comme un seul argument
68         "'ADULT,CHILD',      true,  WEDNESDAY, 16.80",
69         "'ADULT:4',          true,  WEDNESDAY, 36.00",
70         "'STUDENT:4',        false, MONDAY,   28.80",
71         "'ADULT,SENIOR,CHILD', false, WEDNESDAY, 18.80"
72     })
73     @DisplayName("computeTotal() should handle combinations of rules correctly")
74     void testCombinedRules(String ticketsStr, boolean is3D, DayOfWeek day, double expectedTotal) {
75         List<TicketType> tickets;
76         // Gère la notation "TYPE:QUANTITE"
77         if (ticketsStr.contains(s":")) {
78             String[] parts = ticketsStr.split(regex:":");
79             TicketType type = TicketType.valueOf(parts[0]);
80             int count = Integer.parseInt(parts[1]);
81             tickets = Collections.nCopies(count, type);
82         } else {
83             // Gère la notation "TYPE1,TYPE2,..." pour n'importe quel nombre de tickets
84             tickets = Arrays.stream(ticketsStr.split(regex:","))
85                 .map(TicketType::valueOf)
86                 .collect(Collectors.toList());
87         }
88
89         PriceBreakdown breakdown = engine.computeTotal(tickets, day, is3D);
90         // On utilise une tolérance (delta) pour les comparaisons de doubles
91         assertEquals(expectedTotal, breakdown.getTotal(), 0.01);
92     }
93 }
```

# README CINEMA PROJECT - JAVA

## Webhook :

Mise en place d'un Webhook pour avoir un véritable flux de travail d'intégration continue (CI).

## Phase 1 : Préparation du Projet Local et GitHub

### 1. Initialisation du projet local :

- Le projet Java Maven existant a été transformé en dépôt Git avec la commande `git init -b main`.
- Tous les fichiers du projet ont été ajoutés (`git add .`) et sauvegardés dans un premier commit (`git commit -m "Premier commit"`).

### 2. Création du dépôt distant :

- Un nouveau dépôt entièrement vide a été créé sur GitHub. Aucune option (`README`, `.gitignore`) n'a été sélectionnée pour éviter les conflits d'historiques.

### 3. Connexion et premier `push` :

- Le dépôt local a été lié au dépôt distant GitHub avec `git remote add origin <URL_du_dépôt>`.
- Le contenu local a été envoyé vers GitHub avec `git push -u origin main`.

## Phase 2 : Configuration de Jenkins

### 1. Installation des plugins :

- Les plugins essentiels ont été installés via `Administrer Jenkins > Plugins` :
  - `GitHub Integration`
  - `Pipeline`
  - `HTML Publisher plugin`

### 2. Création du `Jenkinsfile` :

- Un fichier `Jenkinsfile` a été ajouté à la racine du projet pour décrire le pipeline.



## README CINEMA PROJECT - JAVA

```
Jenkinsfile
1 pipeline {
2   agent any
3   stages {
4     stage('Build & Test') {
5       steps {
6         script {
7           if (isUnix()) {
8             sh './mvnw -B -V clean verify'
9           } else {
10            bat './mvnw.cmd -B -V clean verify'
11          }
12        }
13      }
14    }
15    stage('Archive') {
16      steps {
17        archiveArtifacts artifacts: 'target/*.jar', fingerprint: true
18        publishHTML(target: [allowMissing: true, alwaysLinkToLastBuild: true, keepAll: true,
19          reportDir: 'target/site/jacoco', reportFiles: 'index.html', reportName: 'JaCoCo'])
20      }
21    }
22  }
23 }
24
```

### Contenu du fichier Jenkinsfile :

- Ce fichier a été versionné et poussé sur GitHub.

### 3. Création du Pipeline Jenkins :

- Un nouvel item de type Pipeline a été créé.
- Dans la configuration, la Définition a été passée à "Pipeline script from SCM".
- Le SCM a été configuré sur Git, pointant vers l'URL du dépôt GitHub.

Pipeline script from SCM

SCM ?

Git ?

Repositories ?

Repository URL ?

https://github.com/galaxie-off/TPGit231025v2.git

Credentials ?

Bucun

+ Ajouter

Avancé ▾

+ Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

\*/main

+ Add Branch

Navigateur de la base de code ?

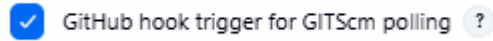
(Auto)

Additional Behaviours

+ Ajouter

## README CINEMA PROJECT - JAVA

Le déclencheur "GitHub hook trigger for GITScm polling" a été activé.



### Phase 3 : Configuration du Webhook et Connexion Externe

1. Exposition de Jenkins local via `ngrok` :
  - `ngrok` a été téléchargé et installé.
  - Un tunnel a été créé pour exposer le port `8080` de Jenkins à une URL publique avec la commande `ngrok http 8080`.

```
ngrok (Ctrl-
♦ Block threats before they reach your services with new WAF actions → https://ngrok.com/r/waf

Session Status      online
Account             xxgalaxie89@gmail.com (Plan: Free)
Update             update available (version 3.31.0, Ctrl-U to update)
Version            3.24.0-msix
Region             Europe (eu)
Latency            59ms
Web Interface       http://127.0.0.1:4040
Forwarding          https://frowsiest-prefraternally-katy.ngrok-free.dev -> http://localhost:8080

Connections          ttl    opn    rt1    rt5    p50    p90
                   25     0     0.00   0.00   30.03  33.03

HTTP Requests
-----
15:39:45.088 CEST POST /github-webhook/ 200 OK
15:38:12.246 CEST POST /github-webhook/ 200 OK
15:34:02.868 CEST POST /github-webhook/ 200 OK
15:28:47.215 CEST POST /github-webhook/ 200 OK
15:26:51.145 CEST POST /github-webhook/ 200 OK
15:23:28.394 CEST POST /github-webhook/ 200 OK
15:21:20.225 CEST GET  /i18n/resourceBundle 200 OK
15:21:20.207 CEST GET  /static/cce0a9af/images/rage.svg 200 OK
15:21:19.717 CEST GET  /github-webhook/ 405 Method Not Allowed
15:18:57.109 CEST POST /github-webhook/ 200 OK
```

## README CINEMA PROJECT - JAVA

### 2. Création du Webhook sur GitHub :

- Dans les Settings > Webhooks du dépôt GitHub, un nouveau webhook a été ajouté.
- Payload URL : L'URL publique fournie par `ngrok` a été utilisée, suivie de `/github-webhook/` (ex: `https://<id-ngrok>.ngrok-free.app/github-webhook/`).
- Content type : `application/json`.
- Trigger : "Just the push event" a été sélectionné.

Settings

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

**Payload URL \***

`https://frowsiest-prefraternally-katy.ngrok-free.dev/github-webhook/`

**Content type \***

`application/json`

**Secret**

**SSL verification**

By default, we verify SSL certificates when delivering payloads.

☒ Enable SSL verification ☐ Disable (not recommended)

**Which events would you like to trigger this webhook?**

☒ Just the push event.

☐ Send me everything.

☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

**Update webhook** **Delete webhook**

# README CINEMA PROJECT - JAVA

## Phase 4 : Sécurité et Finalisation

### 1. Gestion des permissions :

- Un premier build a échoué car il a été déclenché par un utilisateur "anonyme" sans les droits nécessaires.
- Correction : Dans [Administrer Jenkins](#) > [Configurer la sécurité globale](#) > [Autorisation](#), la permission Job > Build a été accordée au groupe Anonymous Users pour permettre au webhook de fonctionner sur une installation locale non sécurisée.

Résultat : Le système est maintenant pleinement opérationnel. Chaque [git push](#) vers la branche [main](#) du dépôt GitHub déclenche automatiquement le pipeline Jenkins, qui compile, teste (avec génération des rapports JaCoCo et JUnit) et archive les artéfacts du projet.

The screenshot shows the Jenkins web interface for a pipeline named 'TPGit231025'. On the left is a sidebar with navigation links: Status, Changes, Lancer un build, Configurer, Supprimer Pipeline, JaCoCo, Stages, Renommer, Pipeline Syntax, GitHub Hook Log, and Identifiants. The main area shows the pipeline's status as 'TPGit231025' with a green checkmark. Below this, it lists 'Last Successful Artifacts' with a file named 'string-toolkit-1.0.0-SNAPSHOT.jar' (5,37 KiB) and a 'view' link. A section titled 'Liens permanents' lists several build links with their status and timestamps. At the bottom, a 'Builds' panel shows a list of recent builds, including build #10 (15:39) and build #11 (16:02), each with a status icon and a dropdown arrow.

**Jenkins** / TPGit231025

**Status**

**TPGit231025**

**Changes**

**Lancer un build**

**Configurer**

**Supprimer Pipeline**

**JaCoCo**

**Stages**

**Renommer**

**Pipeline Syntax**

**GitHub Hook Log**

**Identifiants**

**Last Successful Artifacts**

**string-toolkit-1.0.0-SNAPSHOT.jar** 5,37 KiB [view](#)

**Liens permanents**

- [Dernier build \(#10\)](#), il y a 21 mn
- [Dernier build stable \(#10\)](#), il y a 21 mn
- [Dernier build avec succès \(#10\)](#), il y a 21 mn
- [Dernier build en échec \(#7\)](#), il y a 32 mn
- [Dernier build non réussi \(#8\)](#), il y a 27 mn
- [Dernier build complété \(#10\)](#), il y a 21 mn

**Builds**

Filter

Today

- ✓ #10 15:39
- ✓ #9 15:38

Today

- ✓ #11 16:02