TP – Cinema Pricing Engine (énoncé)

> *Un mini-moteur de tarification pour un cinéma. Objectif : **implémenter**, **tester** (JUnit 5) et **mesurer la couverture** (JaCoCo) d'un petit module Java. Optionnel : exécuter sous **Jenkins** via Maven Wrapper.*

Objectifs pédagogiques

- Structurer un petit **module Java** (classes claires, responsabilités simples).
- Écrire des **tests JUnit 5** couvrant cas nominaux, bords et erreurs.
- Lire un **rapport JaCoCo** et augmenter la couverture (viser **≥ 75–85 %**).
- (Option) Intégrer l'exécution dans **Jenkins** (Freestyle + Maven Wrapper).

Périmètre fonctionnel

On calcule le **prix total** d'une commande de billets selon :

- Liste de billets (types) : `ADULT`, `CHILD`, `SENIOR`, `STUDENT`.
- Séance **3D** (oui/non).
- **Jour** de la séance (`DayOfWeek`).

Prix de base (par billet)

- `ADULT` = **10.00 €**
- `CHILD` = **6.00 €**
- `SENIOR` = **7.50 €**
- `STUDENT` = **8.00 €**

Règles (dans cet **ordre**)

1. **Mercredi** (`DayOfWeek.WEDNESDAY`) : **−20 %** *sur tout le panier*.
2. **3D** : **+2.00 € par billet** si séance 3D.
3. **Groupe** (≥ **4** billets) : **−10 %** *sur le panier*.

> *Le total final est **arrondi au centime** (2 décimales).*
> ***Pas de TVA** ni de devises multiples, on reste simple.*

Design technique (recommandé)

```
src/main/java/edu/cinema/pricing/
├── TicketType.java (enum)
├── PriceBreakdown.java (POJO résultat)
└── PricingEngine.java (logique de calcul)
```

`enum TicketType`

```
public enum TicketType { ADULT, CHILD, SENIOR, STUDENT }
```

`PriceBreakdown`

Expose un récapitulatif clair pour l'affichage/test :

```
public final class PriceBreakdown {
    private final double subtotal;         // somme des prix 
    private final double wednesdayDisc;    // remise -20% si m
    private final double threeDSurcharge;  // +2€ * nbBillets
    private final double groupDisc;        // remise -10% si 
    private final double total;            // total final arr

    // + constructeur, getters, toString()
}
```

`PricingEngine`

```
 import java.time.DayOfWeek;
import java.util.List;

public class PricingEngine {

    public double basePrice(TicketType type) {
        // ADULT=10, CHILD=6, SENIOR=7.5, STUDENT=8
        // IllegalArgumentException si type null
        return 0.0; // à implémenter
    }

    public PriceBreakdown computeTotal(List<TicketType> ticke
        // Préconditions: tickets != null, day != null sinon
        // 1) subtotal
        // 2) appliquer remises/suppléments dans l'ordre : me
        // 3) arrondir au centime
        // 4) retourner PriceBreakdown
        return null; // à implémenter
    }
}
```

Conseil : factorisez une méthode `roundToCents(double)` pour les arrondis.

Plan de tests (JUnit 5)

Couvrir nominaux + bords + erreurs :

`basePrice()`

- 1 test par type (4 tests) + `null` → `IllegalArgumentException`.

`computeTotal()`

- Panier vide → total **0.00**, pas d'erreur.
- 1 billet sans options (chaque type).
- Mercredi uniquement (−20 %).
- 3D uniquement (+2 €/billet).

- Groupe uniquement (≥ 4 billets : −10 %).
- **Combinaisons** : Mercredi + 3D, Mercredi + 3D + Groupe (vérifier l'ordre).
- **Arrondis** : cas amenant des fractions répétées.
- **Erreurs** : `tickets == null`, `day == null` → `IllegalArgumentException`.

Utilisez `@DisplayName` et `@ParameterizedTest` (CSV) pour multiplier des cas rapidement.

Étapes guidées

Cloner le dépôt et vérifier le build :

```
./mvnw -V -B clean verify   # (Windows: mvnw.cmd)
```

1. Coder `TicketType`, `PriceBreakdown` (POJO), squelette `PricingEngine`.
2. Implémenter `basePrice()` puis tests associés.
3. Implémenter `computeTotal()` par étapes : subtotal → mercredi → 3D → groupe → arrondi.
4. Écrire les tests (nominaux, bords, erreurs, combinaisons).
5. Lancer JaCoCo :

```
./mvnw verify
```

6. Ouvrir `target/site/jacoco/index.html` et compléter la couverture si nécessaire.
7. (Option) Jenkins Freestyle : exécuter `mvnw verify` + publier JUnit/JaCoCo.

Exemples d'affaires (attendus approximatifs)

Tickets	3D	Jour	Calcul (résumé)	Total
`[ADULT]`	non	Lundi	`10.00`	10.00
`[ADULT, CHILD]`	oui	Lundi	`(10+6) + 2×2 = 20.00`	20.00
`[STUDENT×4]`	non	Lundi	`32.00 → groupe −10% = 28.80`	28.80
`[ADULT, SENIOR, CHILD]`	non	Mercredi	`23.50 → mercredi −20% = 18.80`	18.80
`[ADULT×4]`	oui	Mercredi	`40.00 → -20% = 32.00 → +4×2 = 40.00 → groupe -10% = 36.00`	36.00

> *L'ordre **Mercredi** → **3D** → **Groupe** doit être respecté.*

Les différentes livrables que vous pouvez me rendre

- URL du dépôt Git (public ou accès fourni).
- Code `TicketType`, `PriceBreakdown`, `PricingEngine`.
- Tests JUnit 5 (≥ 12 cas, lisibles, nominatifs + bords + erreurs).
- **Capture du rapport JaCoCo** (`index.html`) après `mvnw verify`.
- **README** : commandes d'exécution, chemin des rapports.
- (Option) **Capture Jenkins** (onglets *Test Result* + *JaCoCo*).

Commandes utiles

```
# Build + tests + rapport JaCoCo
./mvnw -V clean verify
```

```
# Lancer uniquement les tests
./mvnw test

# Ouvrir le rapport JaCoCo (fichier à ouvrir dans un navigateu
target/site/jacoco/index.html
```

Bon TP & amusez-vous bien