

Partie 3 : Qualité et patrons de conception

Exercice 1 : Soit la classe « Eleve » suivante et son code java :

<<Java Class>> Eleve examGL24
▣ nom: String ▣ age: int ▣ niveau: String ▣ moyenneGenerale: float
⦿ Eleve(String,int,String,float) ⦿ afficherInfos():void ⦿ estAdmis():boolean ⦿ envoyerMessageAuxParents(String,String):void

```
1 package examGL24;
2 class Eleve {
3     private String nom;    private int age;
4     private String niveau;    private float moyenneGenerale;
5
6     public Eleve(String nom, int age, String niveau, float moyenneGenerale) {
7         this.nom = nom;    this.age = age;    this.niveau = niveau;
8         this.moyenneGenerale = moyenneGenerale;    }
9
10    public void afficherInfos() {
11        System.out.println("Nom: " + nom);    System.out.println("Age: " + age);
12        System.out.println("Niveau: " + niveau);
13        System.out.println("Moyenne générale: " + moyenneGenerale);    }
14
15    public boolean estAdmis() {return moyenneGenerale >= 10.0; }
17    public void envoyerMessageAuxParents(String contenu, String methodeEnvoi) {
18        switch (methodeEnvoi) {
19            case "SMS":System.out.println("SMS envoyé aux parents de " +
20                nom + ": " + contenu);    break;
21            case "WhatsApp": System.out.println("Message WhatsApp envoyé aux
22                nom + ":" + contenu);break;
23            case "Email":    System.out.println("Email envoyé aux parents de "+
24                " avec le contenu:" + contenu);br
25            case "Lettre": System.out.println("Lettre postale envoyée aux par
26                nom + ": " + contenu);    break;
27            default:System.out.println("Autre méthode d'envoi de messages.");
```

Document signé numériquement

Signataire: Kamal

Date: 09/08/2025 13:43:25

Signature électronique certifiée

1- En analysant le code ci-dessus et le diagramme de classes correspondant, est-ce que la classe «**Eleve**» respecte « le principe de responsabilité unique : SRP » ? justifier votre réponse ?

[illegible]

2- Afin de respecter le principe SRP, diviser la classe d'origine «Eleve» en des classes homogènes et cohérentes et donner le **nouveau diagramme de classes** qui respecte le principe SRP (sans faire le nouveau code java).

Blank page.

3- En analysant le code ci-dessus et le diagramme de classes correspondant, est-ce que la classe «Eleve» respecte « le principe de : Ouvert pour l'extension, fermé pour la modification : OCP » ? justifier votre réponse ?

012 034 056 078 090 112 134 156 178 190 212 234 256 278 290 312 334 356 378 390 412 434 456 478 490 512 534 556 578 590 612 634 656 678 690 712 734 756 778 790 812 834 856 878 890 912 934 956 978 990
 001 003 005 007 009 011 013 015 017 019 021 023 025 027 029 031 033 035 037 039 041 043 045 047 049 051 053 055 057 059 061 063 065 067 069 071 073 075 077 079 081 083 085 087 089 091 093 095 097 099
 000 002 004 006 008 010 012 014 016 018 020 022 024 026 028 030 032 034 036 038 040 042 044 046 048 050 052 054 056 058 060 062 064 066 068 070 072 074 076 078 080 082 084 086 088 090 092 094 096 098 099
 000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 054 055 056 057 058 059 060 061 062 063 064 065 066 067 068 069 070 071 072 073 074 075 076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091 092 093 094 095 096 097 098 099
 000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 054 055 056 057 058 059 060 061 062 063 064 065 066 067 068 069 070 071 072 073 074 075 076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091 092 093 094 095 096 097 098 099

4- Si la classe «Eleve» ne respecte pas « le principe de :Ouvert pour l'extension, fermé pour la modification : OCP », avec quel Design Pattern on peut régler ce problème (cochez la bonne réponse) :

- ☐ Design Pattern Strategy
 - ☐ Design Pattern Decorator

Exercice 2 :

Vous êtes chargé de développer un système de gestion des inscriptions d'élèves dans une école privée. Pour cela, vous devez utiliser le patron de conception Décorateur afin de permettre aux élèves et leurs parents de choisir différentes fonctionnalités supplémentaires lors de leur inscription à l'école.

Pour **simplifier l'énoncé** de cet exercice et se focaliser uniquement sur l'implémentation du Design pattern Decorator, **on fixe le prix de base** de l'inscription à l'école à 2000 MAD et on fixe le prix des fonctionnalités supplémentaires (les décorateurs) actuellement disponibles comme suit :

- ✓ Transport scolaire : Ajoute 500 MAD aux frais d'inscription.
- ✓ Cantine scolaire : Ajoute 350 MAD aux frais d'inscription.
- ✓ Langue supplémentaire : Ajoute 450 MAD aux frais d'inscription.

Votre tâche consiste à implémenter ce système en utilisant le pattern Décorateur et en offrant aux élèves et leurs parents la possibilité de choisir les options qu'ils veulent au moment d'inscription.

On aura besoin d'une interface **Inscription** avec les deux méthodes suivantes :

- **float calculerPrixTotalInscription()** : qui renvoie le prix total à payer pour l'inscription selon les options choisies, comme le montre l'exemple d'exécution ci-dessous.
- **String description()** : qui affiche les détails de l'inscription comme le montre l'exemple d'exécution ci-dessous.

On aura besoin aussi d'une classe **InscriptionScolaire** qui représente l'inscription de l'élève aux cours officiels du ministère.

```
interface Inscription { float calculerPrixtotalInscription(); String description();}

//Classe de base InscriptionScolaire

class InscriptionScolaire implements Inscription {

@Override

public float calculerPrixTotalInscription() { return 2000; }

@Override

public String description() {return " Inscription aux cours officiels du ministère"; }}

////////////////////////////////////
//

//ici sont ajoutées les autres classes.....

public class MainExamenGL {

public static void main(String[] args) {

// Création d'une inscription scolaire de base avec un prix initial de 2000 MAD

Inscription inscription = new InscriptionScolaire();
```

```
// Décoration de l'inscription avec différentes fonctionnalités

inscription = new TransportScolaireDecorator(inscription);

inscription = new CantineScolaireDecorator(inscription);

inscription = new LangueSupplementaireDecorator(inscription);

float prixTotal = inscription.calculerPrixTotalInscription();

System.out.println("Prix total de l'inscription : " + prixTotal + " MAD");

System.out.println("Description de l'inscription:\n" + inscription.description()); }}
```






```
Prix total de l'inscription :3300.0 MAD
Description de l'inscription :
  Inscription aux cours officiels du ministère
  Avec Transport scolaire
  Avec Cantine scolaire
  Avec Langue supplémentaire
```

Travail à faire :

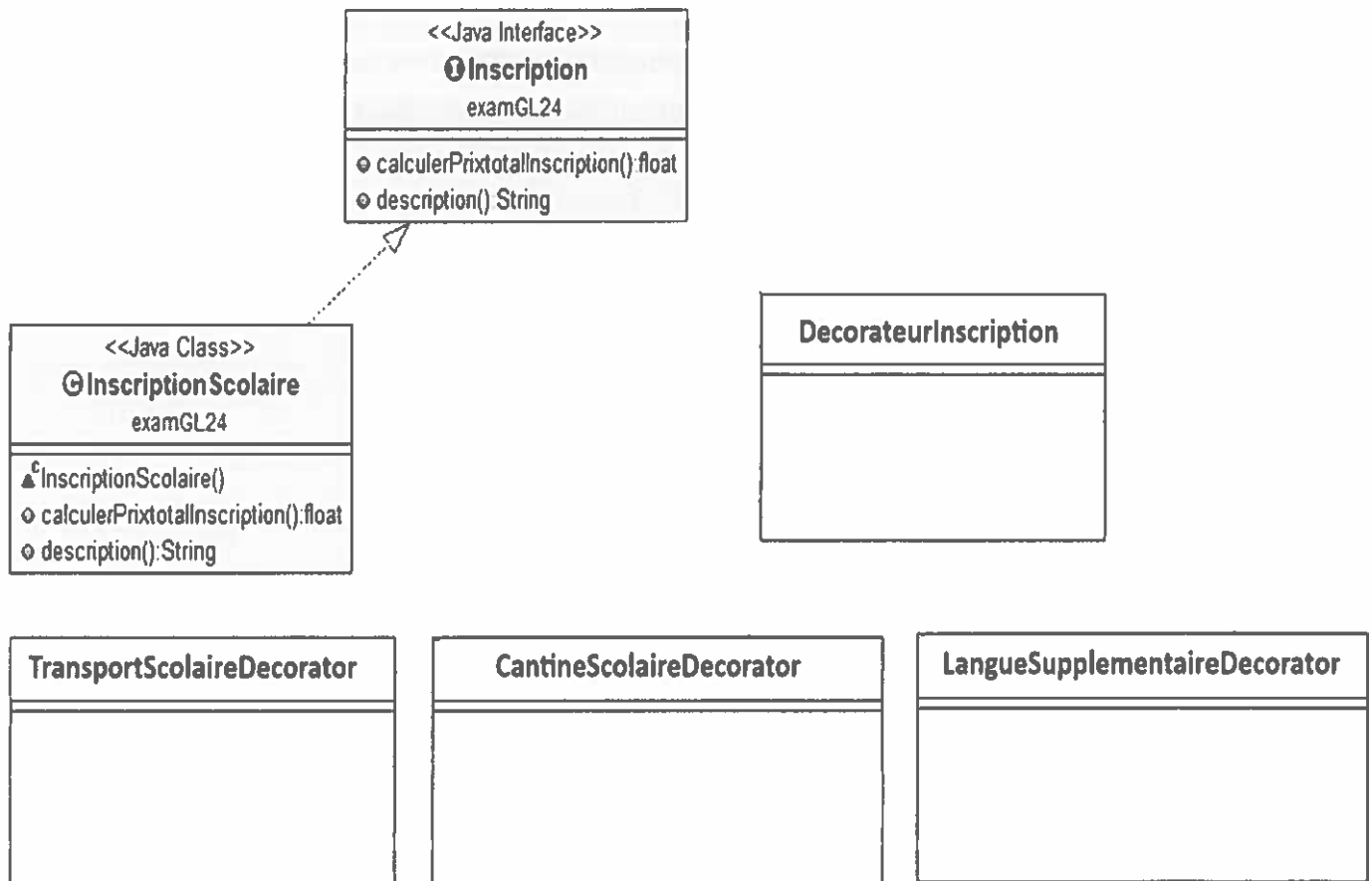
- 1) Compléter le diagramme de classes suivant en précisant les **relations** (héritage, composition, implémentation, ...) entre les différentes classes, les **attributs** et les **méthodes** de chaque classe.

Rappels sur les symboles à utiliser :

Rappels sur les symboles à utiliser :

	Composition
	Héritage/spécialisation (extends)
	Héritage d'interface(implements)

NB : L'interface Inscription peut être remplacée par une classe abstraite



2) Compléter le tableau suivant en précisant le contenu de la classe **DecorateurInscription**

Nom de la classe	Liste des attributs	Liste des méthodes
DecorateurInscription		