
Modèles de Conception

GUIDE DE L'ÉTUDIANT S3 – APP1

Été 2023 – Semaines 1 et 2

Auteur : Domingo Palao Muñoz, d'après une problématique du Pr. Daniel Dalle.
Version : 1.01 (27 avril 2023 à 18:00:00)

Ce document est réalisé avec l'aide de \LaTeX et de la classe `gegi-app-guide`.

©2023 Tous droits réservés. Département de génie électrique et de génie informatique, Université de Sherbrooke.

TABLE DES MATIÈRES

1	ACTIVITÉS PÉDAGOGIQUES ET COMPÉTENCES	1
2	SYNTHÈSE DE L'ÉVALUATION	2
3	QUALITÉS DE L'INGÉNIEUR	3
4	ÉNONCÉ DE LA PROBLÉMATIQUE	4
5	CONNAISSANCES NOUVELLES	6
6	GUIDE DE LECTURE	7
7	LOGICIELS ET MATÉRIEL	8
8	SANTÉ ET SÉCURITÉ	9
9	SOMMAIRE DES ACTIVITÉS	10
10	PRODUCTIONS À REMETTRE	11
11	ÉVALUATIONS	12
12	POLITIQUES ET RÈGLEMENTS	14
13	INTÉGRITÉ, PLAGIAT ET AUTRES DÉLITS	15
14	PRATIQUE PROCÉDURALE 1	16
15	PRATIQUE EN LABORATOIRE	21
16	PRATIQUE PROCÉDURALE 2	35
17	VALIDATION AU LABORATOIRE	44
A	ANNEXE	45

LISTE DES FIGURES

A.1 Diagramme de Classes.	48
-----------------------------------	----

LISTE DES TABLEAUX

2.1	Synthèse de l'évaluation de l'unité	2
2.2	Calcul d'une cote et d'un niveau d'atteinte d'une qualité	2
11.1	Sommaire de l'évaluation du rapport	12
11.2	Grille d'indicateurs utilisée pour les évaluations	13

1 ACTIVITÉS PÉDAGOGIQUES ET COMPÉTENCES

GIF350 – Modèles de Conception

1. Décomposer un problème et définir une architecture logicielle.
2. Mettre en œuvre des modèles de conception.

Description officielle : <https://www.usherbrooke.ca/admission/fiches-cours/GIF350/modeles-de-conception/>

2 SYNTHÈSE DE L'ÉVALUATION

Évaluation	GIF350-1	GIF350-2	Qualités
Rapport d'APP et livrables associés	30	30	Q 4.2
Validation	30	30	Q 1.3
Évaluation sommative théorique	120	0	Q 4.1
Évaluation sommative pratique	0	120	Q 4.5
Évaluation finale théorique	120	0	Q 4.1
Évaluation finale pratique	0	120	Q 4.5
Total	300	300	

Tableau 2.1 Synthèse de l'évaluation de l'unité

On peut aussi donner ici la grille de correspondances des notes avec les cotes et le niveau d'atteinte global d'une qualité. La class `gegi-app-guide` utilise le package `gegi-qualites` pour afficher les douze qualités de l'ingénieur, on le réutilise pour afficher la grille des cotes : voir la documentation de `gegi-qualites`.

Tableau 2.2 Calcul d'une cote et d'un niveau d'atteinte d'une qualité

Note(%)	<50	50	53	57	60	64	68	71	75	78	81	85
Cote	E	D	D+	C-	C	C+	B-	B	B+	A-	A	A+
Niveau	N0	N1	N1	N1	N2	N2	N2	N3	N3	N3	N4	N4
Libellé	Insuffisant	Passable (seuil)			Bien			Très bien (cible)			Excellent	

3 QUALITÉS DE L'INGÉNIEUR

Les qualités de l'ingénieur visées et évaluées par cette unité d'APP sont données dans le tableau un peu plus bas. D'autres qualités peuvent être présentes sans être visées ou évaluées dans cette unité. Pour une description détaillée des qualités et leur provenance, consultez le lien suivant :

<https://www.usherbrooke.ca/genie/etudiants-actuels/au-baccalaureat/bcapg/>

Qualité	Libellé	Touchée	Évaluée
Q01	Connaissances en génie	✓	✓
Q02	Analyse de problèmes		
Q03	Investigation		
Q04	Conception	✓	✓
Q05	Utilisation d'outils d'ingénierie		
Q06	Travail individuel et en équipe		
Q07	Communication		
Q08	Professionnalisme		
Q09	Impact du génie sur la société et l'environnement		
Q10	Déontologie et équité		
Q11	Économie et gestion de projets		
Q12	Apprentissage continu		

4 ÉNONCÉ DE LA PROBLÉMATIQUE

MenuFact. Modèles de Conception

Un collègue de votre entreprise APPsoft vous demande assistance pour développer un code en Java pour une itération de développement d'un logiciel. Le directeur commercial a rencontré le client et il a été convenu de modifier une version prototype du logiciel en utilisant les Modèles de Conception (Design Patterns).

Le directeur sait qu'il y a plusieurs types de Modèles de Conception à utiliser, tel que présentés dans la liste suivante :

— =====	— Adapter
— *Singleton	— Bridge
— *Factory	— Composite
— *Observer	— Decorator
— *State	— Facade
— *MVC	— Chain of Responsibility
— *Flyweight	— Iterator
— =====	— etc.
— Prototype	

Il souhaiterait que les cinq (6) modèles énumérés dans la liste avec un étoile () soient inclus au projet et que vous intégriez en plus, au moins deux (2) autres Modèles de Conception .

L'entreprise emploie une méthodologie orientée objet avec la notation UML (*Unified Modeling Language*). L'approche Orientée Objet est instaurée, de l'analyse à l'implémentation, afin d'améliorer la qualité des produits et d'en faciliter l'entretien. Le processus de développement du projet qui doit mener au prototype à soumettre au client est du type itératif.

Le logiciel en question, nommé **MenuFact**, gère la facturation d'un restaurant. La fonction de base est de maintenir à jour le menu des plats offerts par le restaurant, de permettre la confection des factures et de conclure une transaction de caisse. Pour certains plats, le menu doit inclure des renseignements de nature diététique. La version initiale du logiciel est d'envergure limitée, mais elle permettra néanmoins de valider certains choix de modélisation, d'implémentation des Modèles de Conception et d'en estimer la faisabilité sur la base d'un prototype qui sera soumis au client.

Votre collègue a fait une analyse préliminaire de l'application et il a réalisé aussi la conception et l'implémentation d'une première itération du logiciel, nommée MenuFact01. Ce programme procure une charpente de base avec une classe de tests qui est adaptée pour les premières itérations. Comme votre collègue se rend chez un client pour régler un problème très urgent d'entretien sur un autre logiciel, votre supérieur vous confie le mandat de travailler sur la prochaine itération, nommée MenuFact02.

Pour cette deuxième itération votre superviseur vous demande d'explorer le concept de Modèles de Conception (Design Patterns) pour savoir comment ce concept pourrait améliorer la qualité du logiciel. Votre superviseur a entendu parler des principes SOLID, qui pourraient aider à ce sujet.

De nouvelles fonctionnalités de gestion d'une facture sont à concevoir et à coder pour l'itération MenuFact02. Comme pour tous les projets de l'entreprise, votre travail d'analyse, de conception et d'implémentation de l'itération MenuFact02 doit être documenté adéquatement à l'aide des diagrammes UML du modèle objet. Une première tâche à faire est de préparer le nouveau diagramme de classes avec l'ajout des Modèles de Conception à implémenter.

Plusieurs contraintes de conception de l'application sont imposées selon le document d'analyse préliminaire (voir l'Annexe A) ; elles portent essentiellement sur la hiérarchie des classes et leur implémentation.

Les classes principales seront livrées dans plusieurs packages java bien identifiés et documentés à l'aide de la JavaDoc. Il faudra aussi préparer l'ensemble de Tests Unitaires pour l'application.

À la fin du développement de cette itération vous devez préparer un rapport à votre superviseur en répondant aux critères suivants :

1. Le nouveau diagramme de classes avec les Modèles de Conception à implémenter. Il serait très intéressant d'utiliser une couleur par pattern appliqué.
2. Un tableau pour expliquer l'utilisation de chaque Modèle de Conception utilisé et la justification de son utilisation.
3. Quel est l'avantage d'avoir utilisé des Modèles de Conception pour développer l'application ? Est-ce que vous recommandez cette manière de créer une application ?
4. Quel est l'avantage d'avoir utilisé les Tests Unitaires dans le développement de l'application ? Est-ce que les Tests Unitaires ont changé votre manière de développer l'application ?

X Faire

5 CONNAISSANCES NOUVELLES

Connaissances déclaratives (quoi)

- Utiliser les principes de la programmation orientée objet
- Utiliser la notation UML
- Connaître et utiliser les Modèles de Conception :
 - =====
 - *Singleton
 - *Factory
 - *Observer
 - *State
 - *MVC
 - *Flyweight
 - =====
 - Prototype
 - Adapter
 - Bridge
 - Composite
 - Decorator
 - Facade
 - Chain of Responsibility
 - Iterator
 - etc.
- Connaître et utilise les Tests Unitaires

Connaissances procédurales (comment)

- Choisir le Modèle de Conception le mieux adapté à la situation
- Savoir implémenter un Modèle de Conception dans un projet
- Expliquer les liens entre les classes qui forment un Modèle de Conception

Connaissances conditionnelles (quand)

- Savoir reconnaître les situations qui nécessitent l'utilisation d'un Modèle de Conception
- Utiliser les tests unitaires pour éviter la régression dans le code

6 GUIDE DE LECTURE

6.1 Références essentielles

- A Solid Guide to SOLID Principles
<https://www.baeldung.com/solid-principles>
- Design Patterns. The Catalog of Java Examples
<https://refactoring.guru/design-patterns/java>
- Design Patterns. Catalog
<https://www.oodeesign.com/>
- Référence pour les Test Unitaires
<https://www.jetbrains.com/help/idea/tdd-with-intellij-idea.html>

6.2 Documents complémentaires

- Head First Design Patterns. A Brain-Friendly Guide. Eric Freeman & Elisabeth Robson. Éditions O'Reilly
- Design Patterns. Elements of Reusable Object-Oriented Software. Eric Gamma, Richard Helm, Ralph Johnson & John Vlissides.
- Site de session : <https://www.gel.usherbrooke.ca/s3i>
- Site de logiciel : IntelliJ. <https://www.jetbrains.com/idea/>

7 LOGICIELS ET MATÉRIEL

— L’environnement de travail sera l’IDE IntelliJ de la compagnie Jet Brains.

<https://www.jetbrains.com/community/education/#students>.

La version à utiliser sera la version Ultimate. Cette version peut être installée sur Windows ou sur Linux ou sur Mac OS. Vous devez utiliser votre courriel de l’université pour vous inscrire.

8 SANTÉ ET SÉCURITÉ

8.1 Dispositions générales

Dans le cadre de la présente activité, vous êtes réputés avoir pris connaissance des politiques et directives concernant la santé et la sécurité. Ces documents sont disponibles sur les sites web de l'Université de Sherbrooke, de la Faculté de génie et du département. Les principaux sont mentionnés ici et sont disponibles dans la section *Santé et sécurité* du site web du département : <https://www.gel.usherbrooke.ca/santesecurite/>.

- Politique 2500-004 : Politique de santé et sécurité en milieu de travail et d'études
- Directive 2600-042 : Directive relative à la santé et à la sécurité en milieu de travail et d'études
- Sécurité en laboratoire et atelier au département de génie électrique et de génie informatique

8.2 Dispositions particulières

Aucune.

9 SOMMAIRE DES ACTIVITÉS

Semaine 1

- Première rencontre de tutorat
- Étude personnelle
- Formation à la pratique procédurale 1
- Formation à la pratique en laboratoire
- Formation à la pratique procédurale/laboratoire

Semaine 2

- Consultation facultative
- Étude personnelle et exercices
- Validation pratique de la solution
- Rédaction du rapport d'APP
- Remise des livrables d'APP
- Deuxième rencontre de tutorat
- Consultation facultative
- Évaluation formative théorique écrite et pratique
- Évaluation sommative théorique écrite et pratique

10 PRODUCTIONS À REMETTRE

- Les productions se font par équipe de 2, sauf lorsque indiqué autrement.
- L'identification des membres des équipes doit être faite sur la page web de l'unité avant 16h30, le lendemain de votre premier tutorat. Les personnes qui n'ont pas d'équipe d'APP à ce moment seront mis en équipe par le tuteur.
- La date limite pour le dépôt électronique est le jour de votre deuxième tutorat avant le début du premier groupe. Les retards seront pénalisés.
- Les productions soumises à l'évaluation doivent être originales pour chaque équipe, sinon l'évaluation sera pénalisée en cas de non-respect de cette consigne.

10.1 Productions à remettre

Rapport

Un rapport nommé `rapport.pdf` qui doit répondre aux critères suivantes :

1. Le nouveau diagramme de classes avec les Modèles de Conception à implémenter. Il serait très intéressant d'utiliser une couleur par pattern appliqué dans les classes.
2. Un tableau pour expliquer l'utilisation de chaque Modèle de Conception utilisé dans le projet et la justification de son utilisation.
3. Quel est l'avantage d'avoir utilisé des Modèles de Conception pour développer l'application ? Est-ce que vous recommandez cette manière de créer une application ?
4. Quel est l'avantage d'avoir utilisé les Test Unitaires dans le développement de l'application ? Est-ce que les Tests Unitaires ont changé votre manière de développer l'application ?

Fichiers

Tous les fichiers de code développés. Seulement les fichiers du code source (.java) dans un fichier compressé (code.zip).

Tout dans un fichier compressé nommé `cip1-cip2.zip`, seulement le format .zip sera accepté.

11 ÉVALUATIONS

11.1 Rapport et livrables associés

L'évaluation du rapport portera sur les compétences figurant dans la description des activités pédagogiques. Ces compétences ainsi que la pondération de chacune d'entre elles dans l'évaluation du rapport sont indiquées au tableau 11.1. L'évaluation est directement liée aux livrables demandés à la section 10.1 et le tableau 11.1 y réfère à l'aide d'une courte description.

Élément	GIF350-1	GIF350-2
Diagramme de classes	15	
Tableau pour l'utilisation	15	
Avantages de l'utilisation des Modèles de Conception		15
Avantages de l'utilisation des Tests Unitaires		15
Total	30	30

Tableau 11.1 Sommaire de l'évaluation du rapport

Quant à la qualité de la communication technique elle ne sera pas évaluée de façon sommative, mais si votre rapport est fautif sur le plan de la qualité de la communication et de la présentation, il vous sera retourné et vous devrez le reprendre pour être noté.

11.2 Évaluation sommative

L'évaluation de l'APP se fait en deux étapes :

1. L'évaluation sommative théorique est un examen écrit qui porte sur tous les éléments de compétences de l'unité.
2. L'évaluation sommative pratique porte sur l'utilisation et l'implantation des Modèles de Conception tels que vus pendant l'unité.

Pour l'évaluation vous avez droit au livre de Patterns, celui sera déposé en format PDF sur les postes de travail.

11.3 Qualités de l'ingénieur

La grille d'indicateurs utilisée aux fins de l'évaluation est donnée au tableau 11.2. Il est à noter qu'un niveau d'atteinte d'un *indicateur* dans cette grille n'a pas la même signification qu'un niveau d'atteinte d'une *qualité* dans le tableau 2.2. Cela est normal, un indicateur et une qualité, ce sont deux choses différentes.

Tableau 11.2 Grille d'indicateurs utilisée pour les évaluations

Indicateur	AP	Qualité	Aucun (N0)	Insuffisant (N1)	Seuil (N2)	Cible (N3)	Excellent (N4)
Démontrer, à un niveau universitaire, l'acquisition de connaissances en sciences du génie (Programmation Java).	GIF350-1	Q01.3	... n'applique pas ou très peu de concepts fondamentaux en sciences du génie (Programmation Java).	... applique correctement peu de concepts fondamentaux en sciences du génie (Programmation Java).	... est capable d'appliquer correctement certains des concepts fondamentaux en sciences du génie (Programmation Java).	... applique aisément les concepts fondamentaux en sciences du génie (Programmation Java).	... applique aisément et efficacement les concepts fondamentaux en sciences du génie (Programmation Java).
Décomposer un problème de création de logiciel.	GIF350-1	Q04.1	... ne démontre pas une compréhension de la décomposition d'un problème de création de logiciel.	... démontre une compréhension insuffisante de la décomposition d'un problème de création de logiciel.	... démontre une compréhension minimale de la décomposition d'un problème de création de logiciel.	... démontre une bonne compréhension de la décomposition d'un problème de création de logiciel.	... démontre une excellente compréhension de la décomposition d'un problème de création de logiciel.
Définir une architecture logicielle.	GIF350-1	Q04.2	... ne démontre pas une compréhension de la définition d'une architecture logicielle.	... démontre une compréhension insuffisante de la définition d'une architecture logicielle.	... démontre une compréhension minimale de la définition d'une architecture logicielle.	... démontre une bonne compréhension de la définition d'une architecture logicielle.	... démontre une excellente compréhension de la définition d'une architecture logicielle.
Valider et implémenter la solution retenue.	GIF350-1	Q04.5	... élabore un design incomplet. Ne vérifie pas que la solution répond aux besoins et exigences.	... élabore un design partiellement complet. Vérifie sommairement que la solution répond aux besoins et aux exigences.	... élabore un design complet. Valide la solution répond aux besoins et aux exigences.	... élabore un design complet et de qualité. Valide adéquatement que la solution répond aux besoins et aux exigences.	... élabore un design complet et de grande qualité. Valide adéquatement que la solution répond aux besoins et aux exigences et propose des améliorations potentielles.

12 POLITIQUES ET RÈGLEMENTS

Dans le cadre de la présente activité, vous êtes réputés avoir pris connaissance des politiques, règlements et normes d'agrément suivants.

Règlements de l'Université de Sherbrooke

- Règlement des études :
<https://www.usherbrooke.ca/registraire/>

Règlements facultaires

- Règlement facultaire d'évaluation des apprentissages / Programmes de baccalauréat
- Règlement facultaire sur la reconnaissance des acquis

Normes d'agrément

- Informations pour les étudiants au premier cycle :
<https://www.usherbrooke.ca/genie/etudiants-actuels/au-baccalaureat/bcapg>
- Informations sur l'agrément :
<https://engineerscanada.ca/fr/agrement/a-propos-de-l-agrement>

Si vous êtes en situation de handicap, assurez-vous d'avoir communiqué avec le *Programme d'intégration des étudiantes et étudiants en situation de handicap* à l'adresse de courriel prog.integration@usherbrooke.ca.

13 INTÉGRITÉ, PLAGIAT ET AUTRES DÉLITS

Dans le cadre de la présente activité, vous êtes réputés avoir pris connaissance de la déclaration d'intégrité relative au plagiat :

<https://www.usherbrooke.ca/enseigner/passeurs-dintegrite/ressources/antiplagiat>

14 PRATIQUE PROCÉDURALE 1

But de l'activité

- Connaître l'utilité des Modèles de Conception .
- Identifier quelques situations pour appliquer des Modèles de Conception .
- Faire des diagrammes de classe avec l'application de Modèles de Conception .

14.1 Exercices préparatoires

P.1 Introduction.

1. Qu'est-ce qu'un Modèle de Conception ?
2. Quel est l'utilité de la standardisation dans la conception du logiciel ?

14.2 Exercices

E.1 Rappel Conception Orientée Objet.

1. Faire le diagramme de classe de la classe qui représente les étudiants. Un étudiant a un nom, une date de naissance, une adresse et un programme. Dans notre système un étudiant est capable d'étudier, de manger, de se déplacer et de dormir.
2. Faire le diagramme de classes de la hiérarchie qui montre que un étudiant est une personne et un professeur est aussi une personne. Dans notre système une personne a un nom, une date de naissance et une adresse. Les étudiants ont un programme et les professeurs un numéro de bureau. Les étudiants sont capables d'étudier, de manger, de se déplacer et de dormir. Les professeurs sont capables de manger, de se déplacer, de dormir et d'enseigner.

E.2 Rappel des concepts de la Programmation Orientée Objet.

1. Expliquez les concepts suivants de la Programmation Orientée Objet :
 - (a) Abstraction
 - (b) Polymorphisme
 - (c) Héritage
 - (d) Encapsulation
2. Décrire le type d'association entre classes suivantes :
 - (a) Association
 - (b) Dépendance
 - (c) Composition

- (d) Agrégation
- (e) Héritage

E.3 Introduction aux Modèles de Conception

Répondre aux questions suivantes :

1. C'est quoi un Modèle de Conception ?
2. Quel est la différence entre un Modèle de Conception et un algorithme ?
3. Comment peut-on représenter un Modèle de Conception ?
4. Donnez la classification des Modèles de Conception. Expliquez chacune des catégories.
5. Expliquez chacune des caractéristiques d'une bonne Conception du logiciel
 - (a) Réutilisation du code
 - (b) Extensibilité
6. Expliquez les principes SOLID.
 - (a) **S**ingle Responsibility Principle
 - (b) **O**pen/Closed Principle
 - (c) **L**iskov Substitution Principle
 - (d) **I**nterface Segregation Principle
 - (e) **D**ependency Inversion Principle

E.4 Tests Unitaires

Répondez aux questions suivantes :

1. C'est quoi un Test Unitaire ?
2. Quel est l'utilité des Test Unitaires ?
3. Pour la classe Calculatrice suivante :

```
1 public class Calculatrice {
2     public int addition (int i, int j)
3     {
4         return i+j;
5     }
6
7     public int multiplication (int i, int j)
8     {
9         return i*j;
10    }
11 }
```

```

12     public int division (int i, int j)
13     {
14         if(j==0)
15             throw new IllegalArgumentException("Le deuxieme argument
16         else
17             return i/j;
18     }
19 }

```

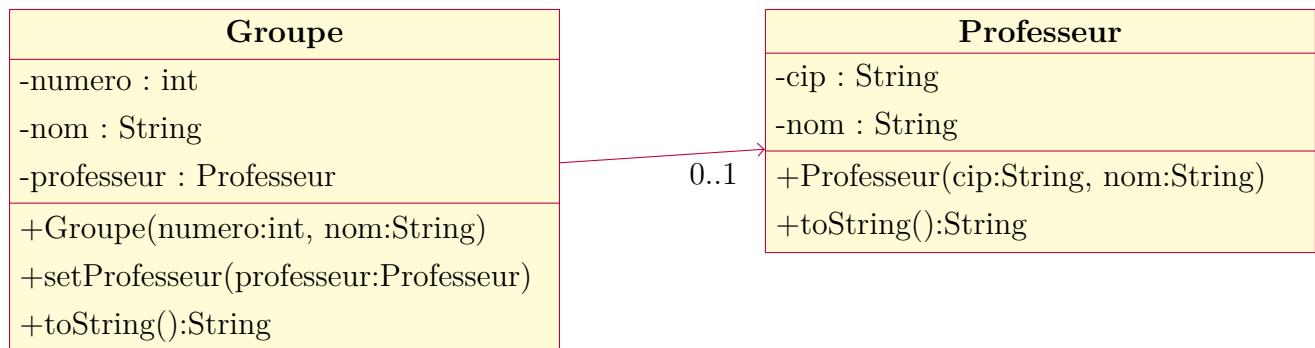
Faire les Test Unitaires nécessaires pour valider le correct fonctionnement de la classe.

E.5 Utilisation du Modèle de conception Singleton

Quel est le problème qui peut résoudre le Modèle de Conception Singleton ?

Situation : L'école de programmation DP Inc. Offre une variété de cours. Pour une groupe d'étudiants il y a un seul professeur tuteur pour tout le parcours du groupe. Le professeur ne peut pas changer au cours de route. Nous avons à développer une application qui sera utilisé sur un téléphone cellulaire qui a l'information d'un seul groupe à la fois.

Nous avons le diagramme de classe suivant :



Le code qu'implemente les classes est :

*Le fichier : **Groupe.java***

```

1 package codeInitial;
2
3 public class Groupe {
4     private int numero;
5     private String nom;
6     private Professeur professeur;
7
8     public Groupe(int numero, String nom) {
9         this.numero = numero;
10        this.nom = nom;

```

```

11     }
12     public void setProfesseur(Professeur professeur) {
13         this.professeur = professeur;
14     }
15
16
17     @Override
18     public String toString() {
19         return "Groupe{" +
20             "numero=" + numero +
21             ", nom='" + nom + '\'' +
22             ", professeur=" + professeur +
23             '}'';
24     }
25 }

```

*Le fichier : **Professeur.java***

```

1 package codeInitial;
2
3 public class Professeur {
4
5     private String cip;
6     private String nom;
7
8     public Professeur(String cip, String nom) {
9         this.cip = cip;
10        this.nom = nom;
11    }
12
13    @Override
14    public String toString() {
15        return "Professeur{" +
16            "hash code=" + this.hashCode() + " \n"+
17            "cip='" + cip + '\'' +
18            ", nom='" + nom + '\'' +
19            '}'';
20    }
21 }

```

*Le fichier : **TestGroupeProfesseur.java***

```

1 package codeInitial;
2
3 public class TestGroupeProfesseur {

```



```

4
5     public static void main(String[] args)
6     {
7         Groupe groupe = new Groupe(1,"Design Patterns");
8         Professeur professeur1 = new Professeur("abcd1234","Mr Professeur
9         Professeur professeur2 = new Professeur("bcde2345","Mr Professeur
10
11         System.out.println("Le groupe : " + groupe);
12         groupe.setProfesseur(professeur1);
13         System.out.println("Le groupe : " + groupe);
14
15         groupe.setProfesseur(professeur2);
16         System.out.println("Le groupe : " + groupe);
17
18     }
19 }

```

Quel est le problème potentiel de cette implémentation ?

Faire le diagramme de classes approprié pour garantir la demande d'implémentation avec un Modèle de Conception.

Préparez le code avec un Modèle de Conception approprié pour garantir la demande d'implémentation.

15 PRATIQUE EN LABORATOIRE

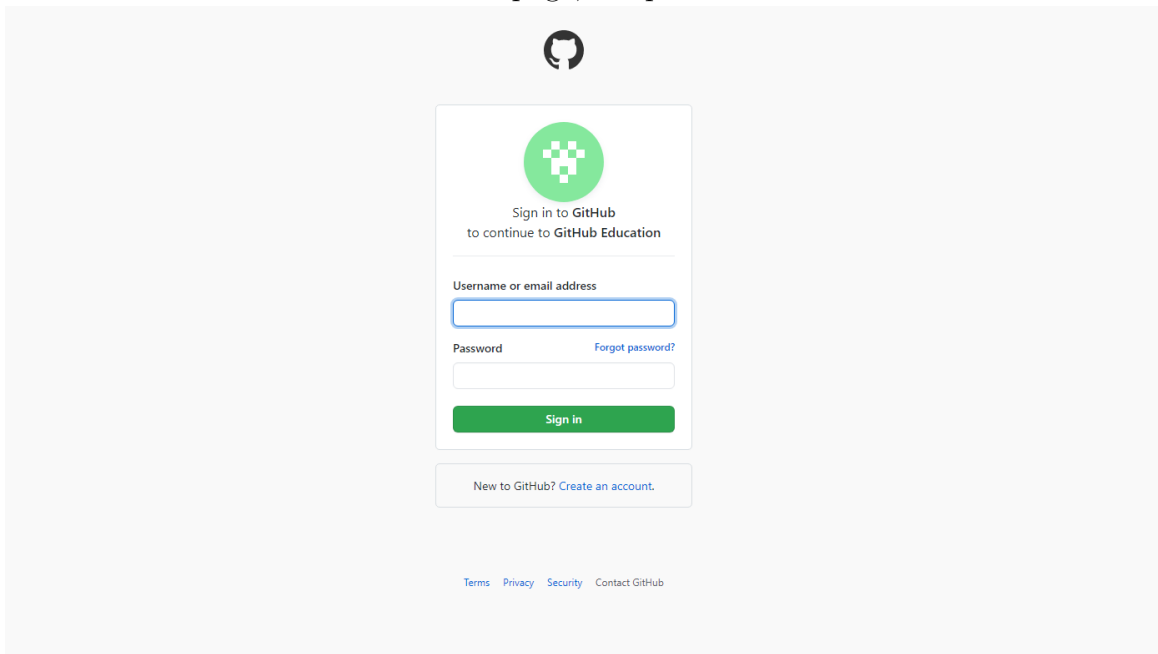
Buts de l'activité

Le but de cette activité est d'expérimenter avec :

- Installation de Github
- Installation de GitKraken
- Utilisation de Git
- IntelliJ
- Modèle de Conception Singleton

E.1 Installation de Github

1. Création d'un compte Github. Rendez-vous au : <https://github.com/>
2. Il est suggéré d'utiliser votre courriel de l'Université comme identifiant.
3. Pour ceux qui veulent avoir un compte premium, suivez les points suivants. Sinon allez au point 15 pour continuer le laboratoire.
4. Suivez ce lien pour appliquer pour un student pack https://education.github.com/discount_requests/student_application. Ceci vous donnera accès à une multitude de services. Une fois rendu à la page, remplissez les informations demandées.



5. Pour la partie *How do you plan to use Github?* vous pouvais marquer ce que vous voulez. Si vous n'avez pas d'idée, alors marquer seulement «étudier».
6. Il est important que vous utilisiez votre adresse courriel de l'université. Vous allez voir le rectangle vert marqué Université de Sherbrooke.

What e-mail address do you use for school?

Note: Selecting a school-issued email address gives you the best chance of a speedy review.

☐ [Redacted]

☒ [Redacted] @usherbrooke.ca ✓ Université de Sherbrooke

[+ Add an email address](#)

What is the name of your school?

Note: If your school is not listed, then enter the full school name and continue. You will be asked to provide further information about your school on the next page.

Université de Sherbrooke

How do you plan to use GitHub?


[Empty text box]

Please note, your request cannot be edited once it has been submitted, so please verify your details for accuracy before sending them to us.

[Submit your information](#)

7. Si tout va bien, alors vous aller voir ceci :

Back to GitHub.com GitHub Support Contact GitHub

 Education Students Teachers Schools Events

[Home](#) / [Get benefits](#)

Thanks for submitting!

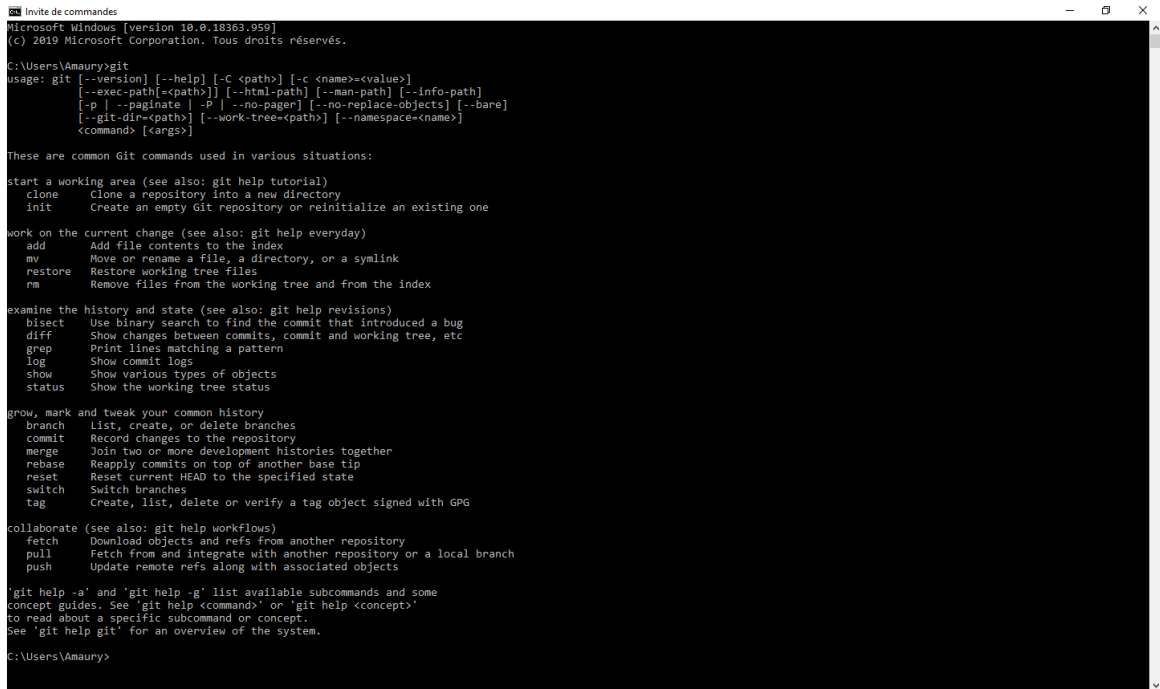
Be sure to check your email. If you don't hear from us within the hour, you should receive an email from us in fewer than **about 1 month**. Have an Octotastic day!

GitHub	Product Features Security Enterprise Customer stories Pricing Resources	Platform Developer API Partners Atom Electron GitHub Desktop	Support Docs Community Forum Professional Services Learning Lab Status Contact GitHub	Company About Blog Careers Press Social Impact Shop
---------------	--	--	--	--

8. **Ne paniquez pas la vérification est presque instantané.** Une fois le courriel reçu, alors vous avez le *student pack* de Github.

Pour voir la liste complète, des services et ainsi leurs avantages suivez ce lien : <https://education.github.com/pack/offers>.

9. Si vous avez déjà installé git, continuez au point 15. Si vous n'êtes pas certains d'avoir installé git, alors vous devez ouvrir une fenêtre d'invite de commandes. Dans cette fenêtre écrivez la commande `git`. Si git est installé, alors une liste comme suit s'affichera. Sinon, il y aura un message d'erreur. Si vous avez déjà installé git alors passez au point suivant.



```
Invite de commandes
Microsoft Windows [version 10.0.18363.959]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\Amaury>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
       [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
       [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
       [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
       <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone Clone a repository into a new directory
  init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add Add file contents to the index
  mv Move or rename a file, a directory, or a symlink
  restore Restore working tree files
  rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect Use binary search to find the commit that introduced a bug
  diff Show changes between commits, commit and working tree, etc
  grep Print lines matching a pattern
  log Show commit logs
  show Show various types of objects
  status Show the working tree status

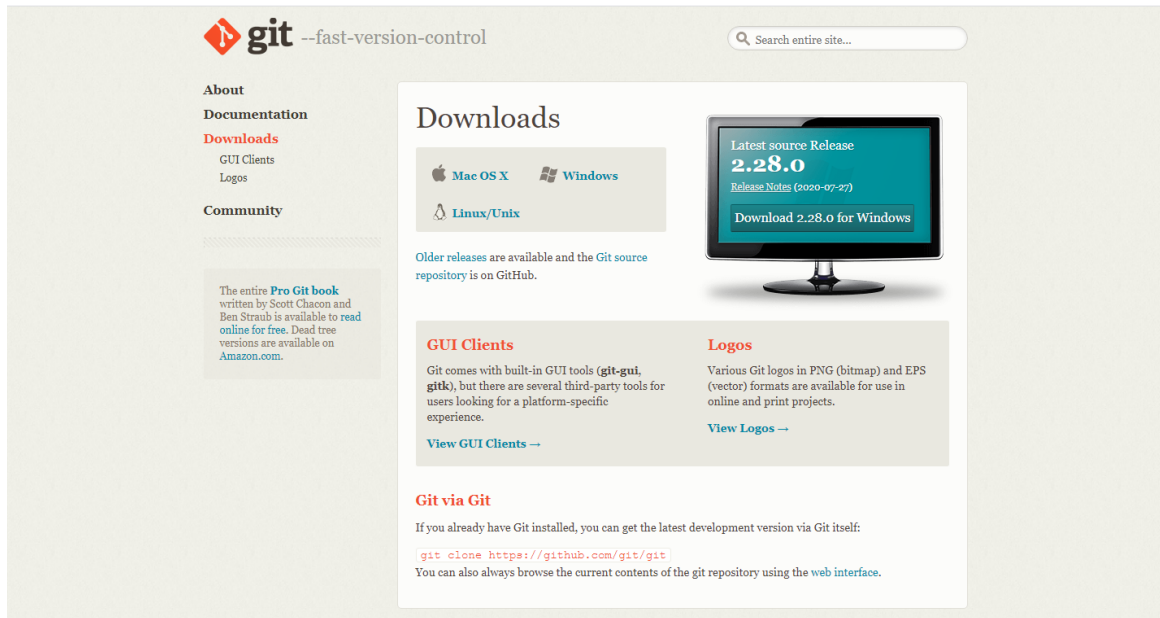
grow, mark and tweak your common history
  branch List, create, or delete branches
  commit Record changes to the repository
  merge Join two or more development histories together
  rebase Reapply commits on top of another base tip
  reset Reset current HEAD to the specified state
  switch Switch branches
  tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch Download objects and refs from another repository
  pull Fetch from and integrate with another repository or a local branch
  push Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

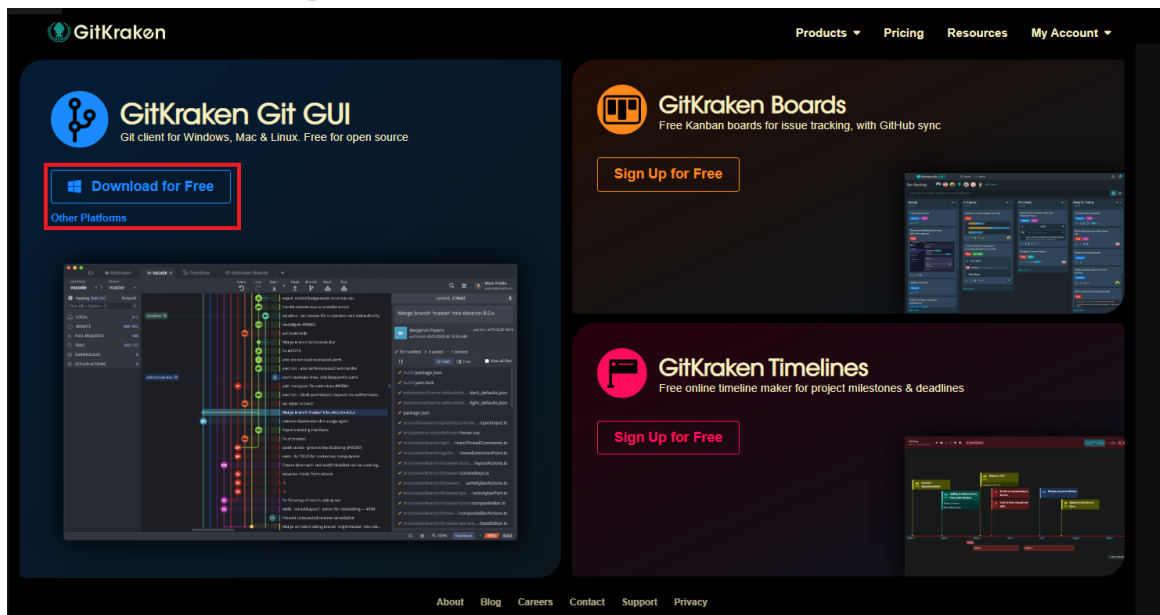
C:\Users\Amaury>
```

10. Pour installer git allez le télécharger pour votre système d'exploitation sur le lien suivant : <https://git-scm.com/downloads>. Suivez les instructions du wizard de git et laissez tout par défaut.

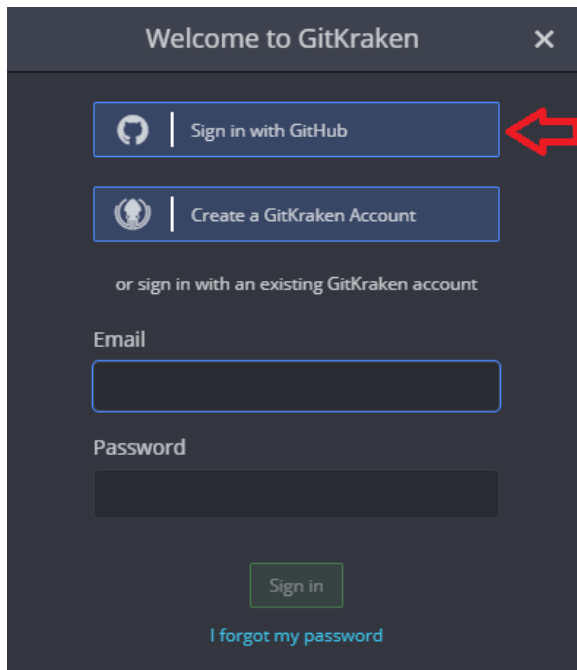


E.2 Installation de GitKraken

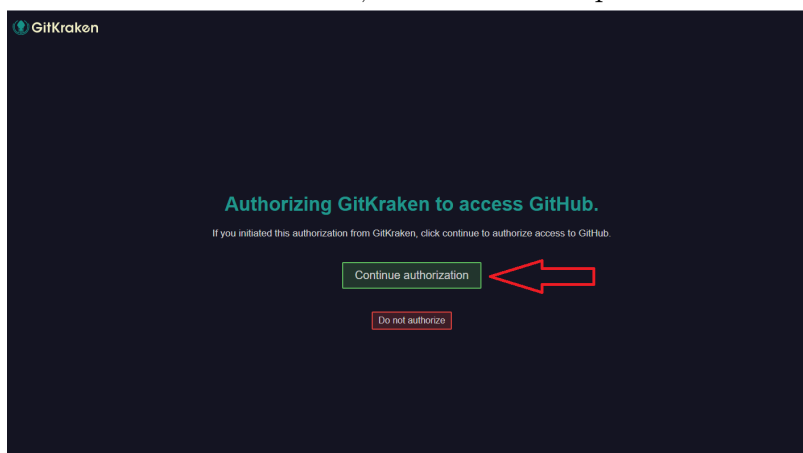
1. Installez GitKraken avec le lien suivant : <https://www.gitkraken.com/>. Pour ceux qui utilisent un autre système d'exploitation, il y a une option juste en dessus du bouton de téléchargement Windows. Une fois, téléchargé, suivez les instructions du wizard et laissez tout par défaut.



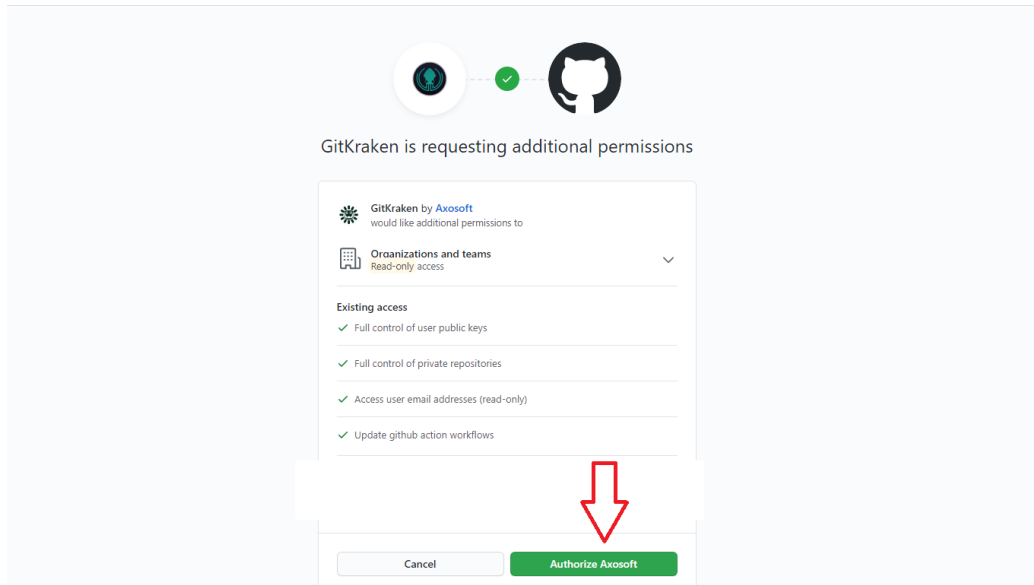
2. Une fois l'application démarrée, vous pouvez entrer vos informations ou cliquer sur la connexion de GitHub.



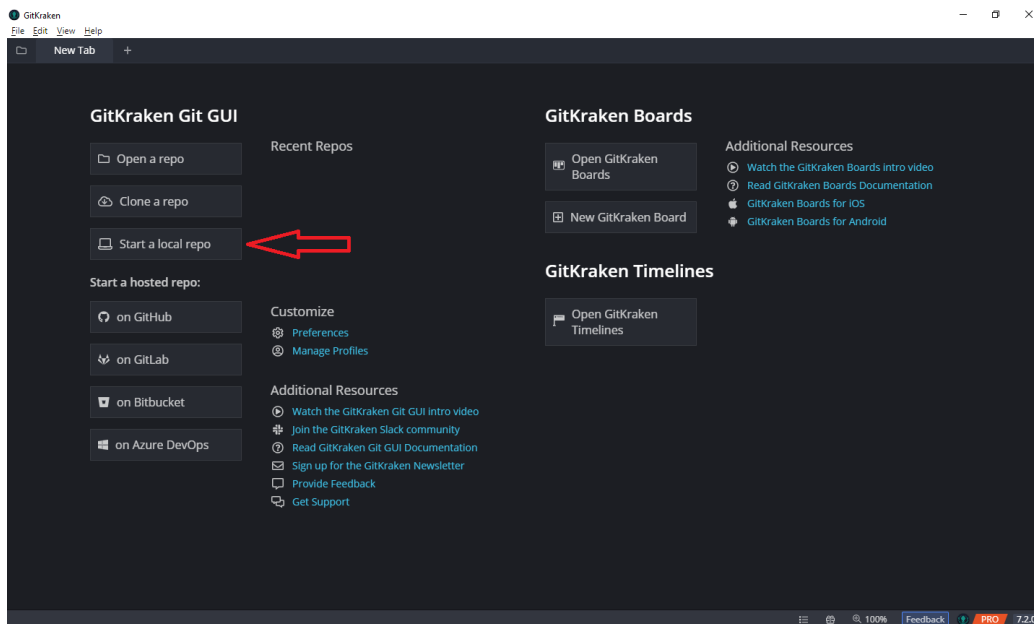
3. Vous allez voir une fenêtre, comme suit. Cliquez sur le bouton *Continue authorization*.



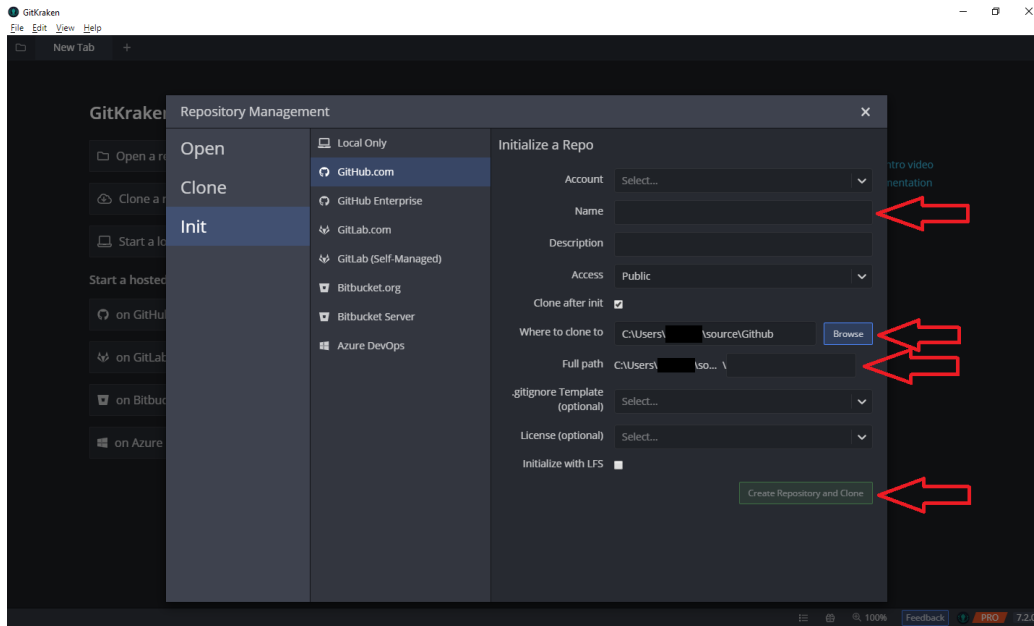
4. Cliquez sur le bouton Authorize Axosoft. Ensuite remplissez les informations demandées.



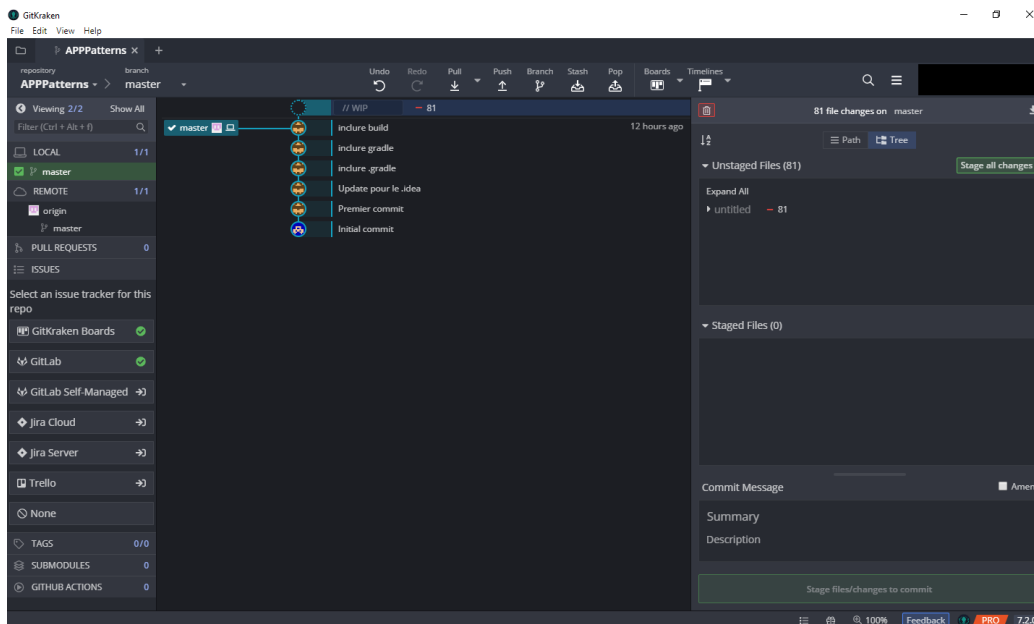
5. Pour commencer un répertoire, cliquez sur *Start a local repo*.



6. Ceci ouvrira une fenêtre contextuelle. Il faut remplir les trois sections suivantes, les autres sont optionnels.
- La section *Name* sera le nom de votre répertoire.
 - La section *Where to clone to* sera l'emplacement de ce qu'on appelle le fichier Git.
 - La section *Full path* c'est le nom du fichier Git, par défaut le nom sera le nom du répertoire.



- Une fois le répertoire créé, la page de gestion de répertoire s'affichera.

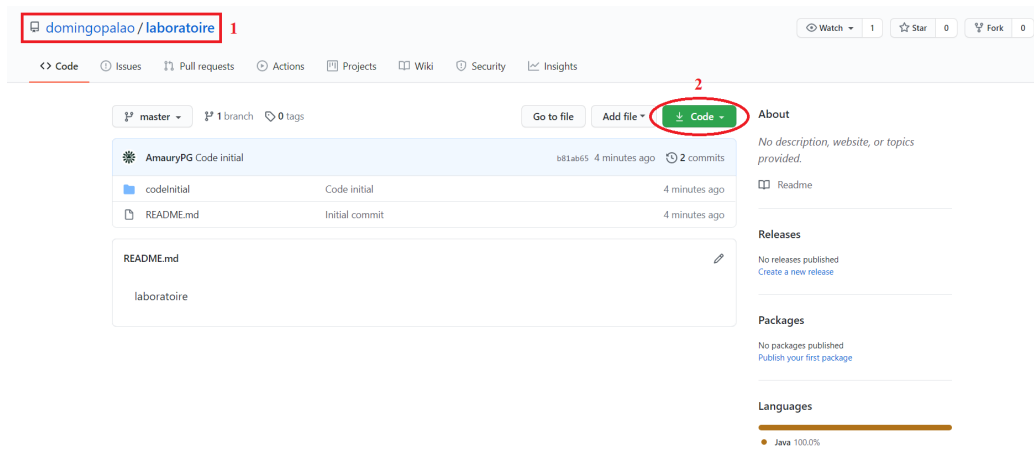


Vous avez réussi à installer git et il est prêt à être utilisé.

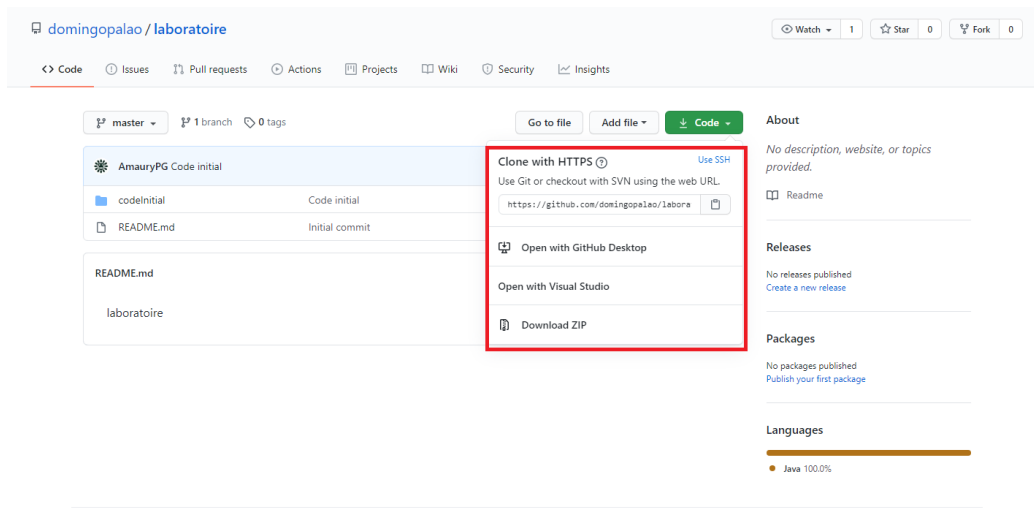
E.3 Utilisation de Git

- Suivez le lien suivant : <https://github.com/domingopalao/laboratoire>.

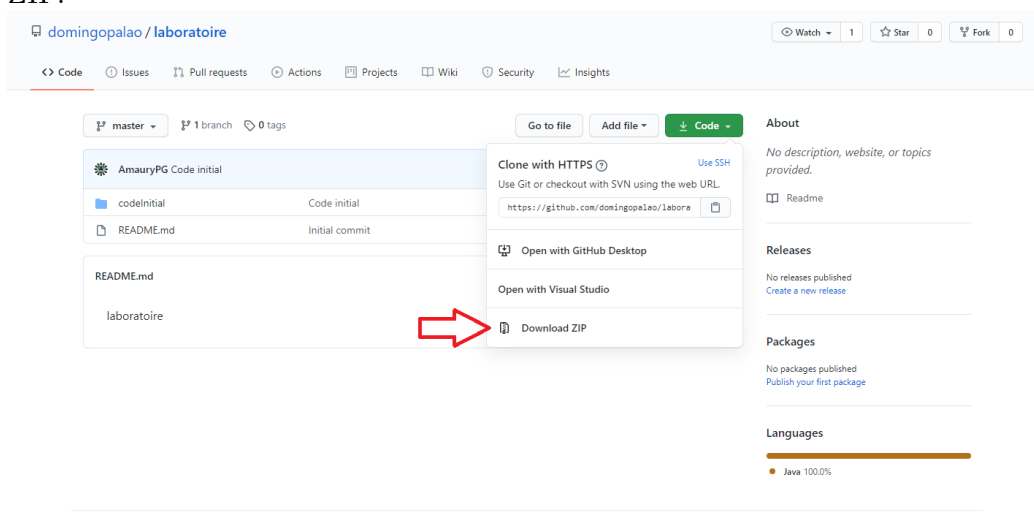
Le titre du répertoire doit être domingopalao/laboratoire. Cliquez sur le bouton Code.



2. Une fois cliqué, une fenêtre contextuelle, comme suit, doit apparaître.



3. Cliquez sur le bouton Download ZIP. Ceci téléchargera le projet de base en format ZIP.

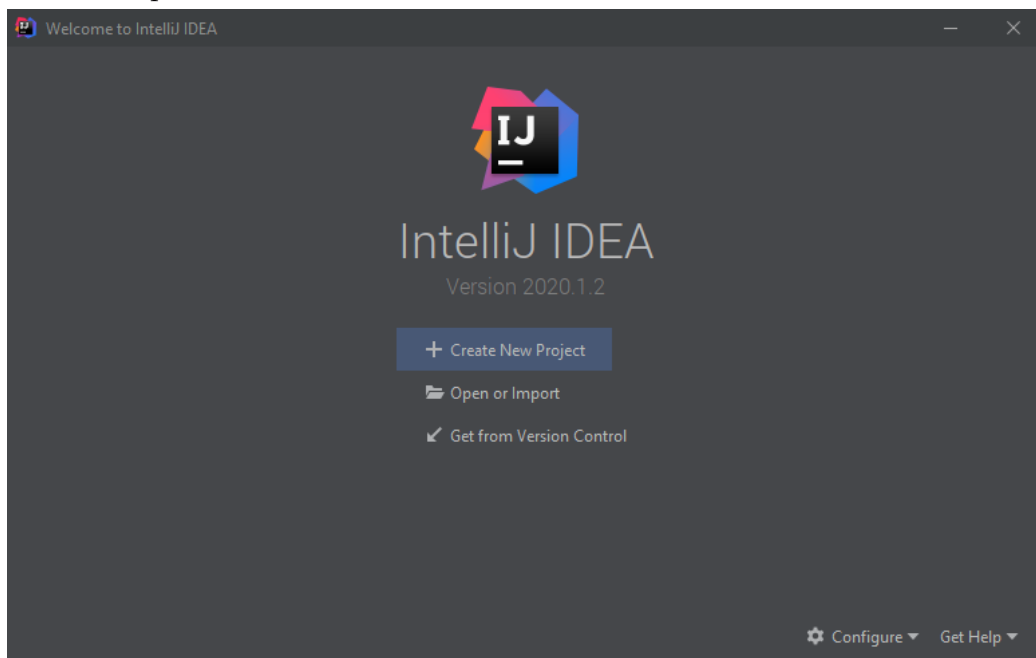


4. Allez dans vos téléchargements et décompressez le fichier dans le lieu de travail que vous avez choisi, avec l'aide de votre décompresseur déjà installé. Prenez ce qui est à l'intérieur du fichier APPPatterns-master et copiez-collez dans le fichier Git que vous avez créé. Pour créer le fichier Git allez voir la documentation gitKraken.

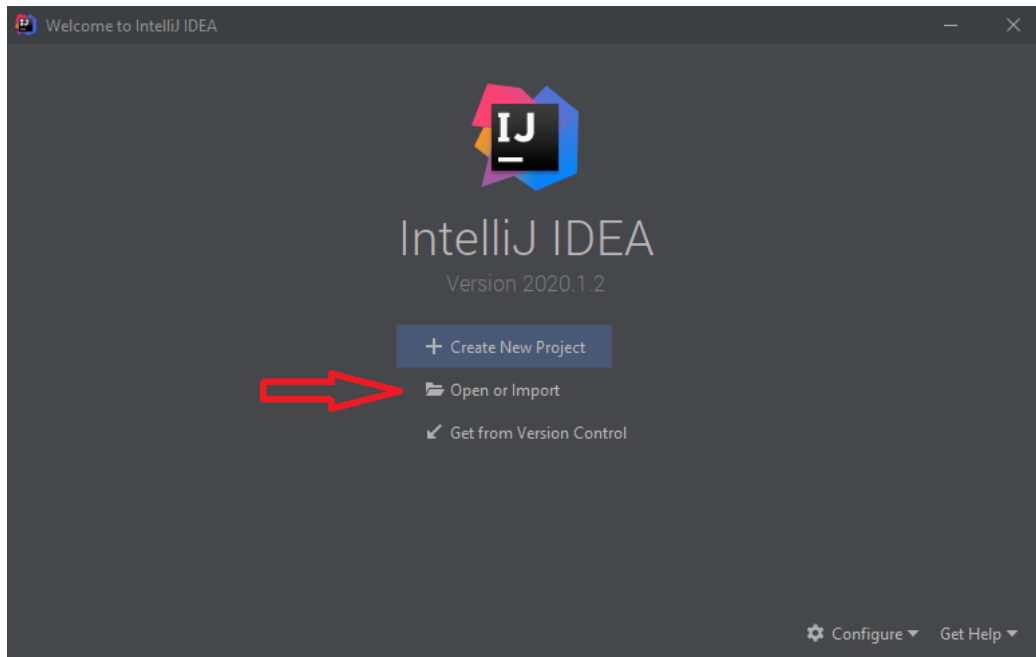
Normalement, il devrait déjà avoir la possibilité de décompresser les fichiers ZIP dans tous les systèmes d'exploitation. Sinon, vous pouvez installer 7Zip ou tout autre décompresseur de votre choix.

E.4 Utilisation de IntelliJ

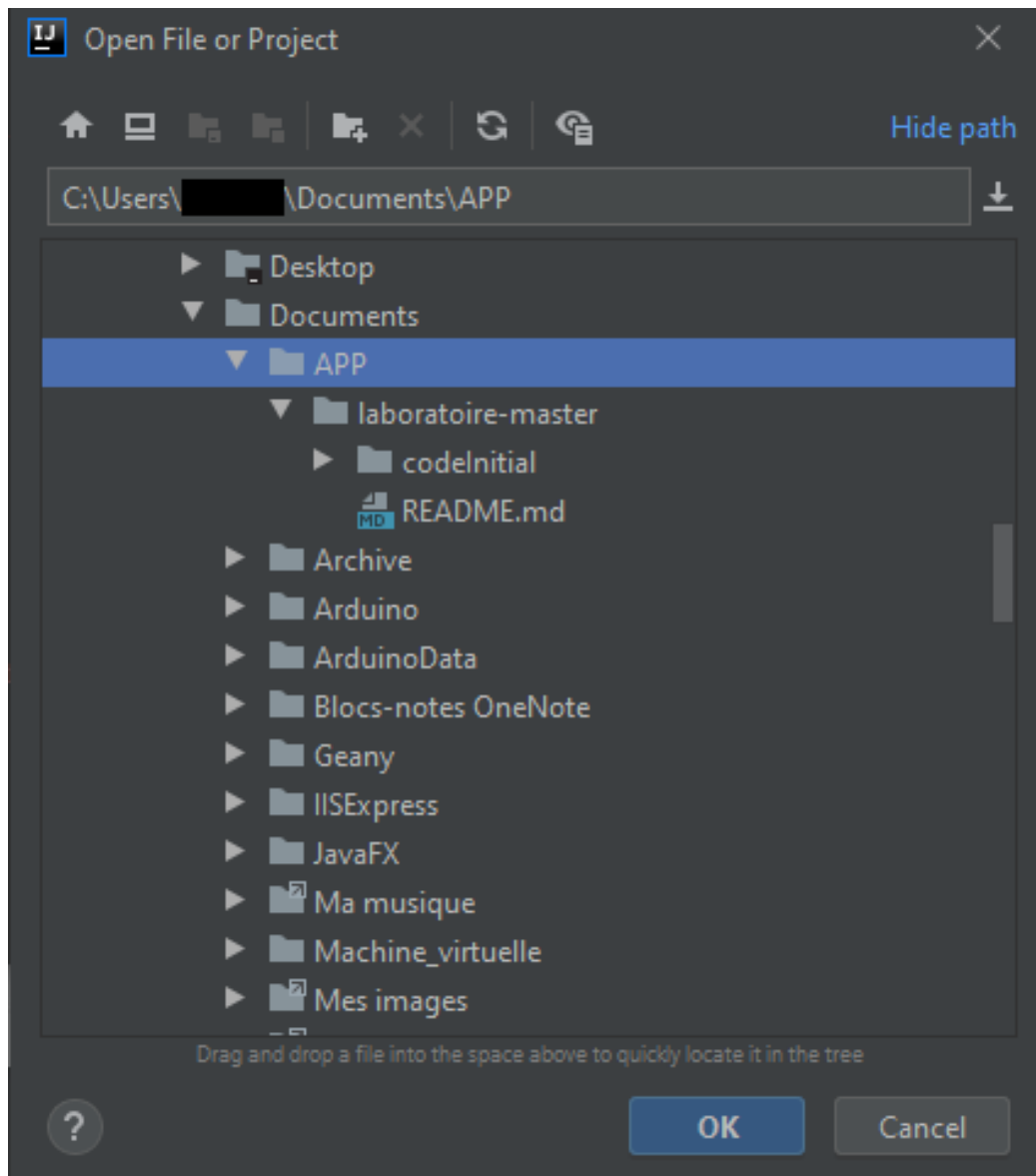
1. Une fois que vous avez décompressé le fichier dans le lieu de travail, ouvrez IntelliJ. Si c'est la première utilisation d'IntelliJ la fenêtre devrait ressembler à ceci :



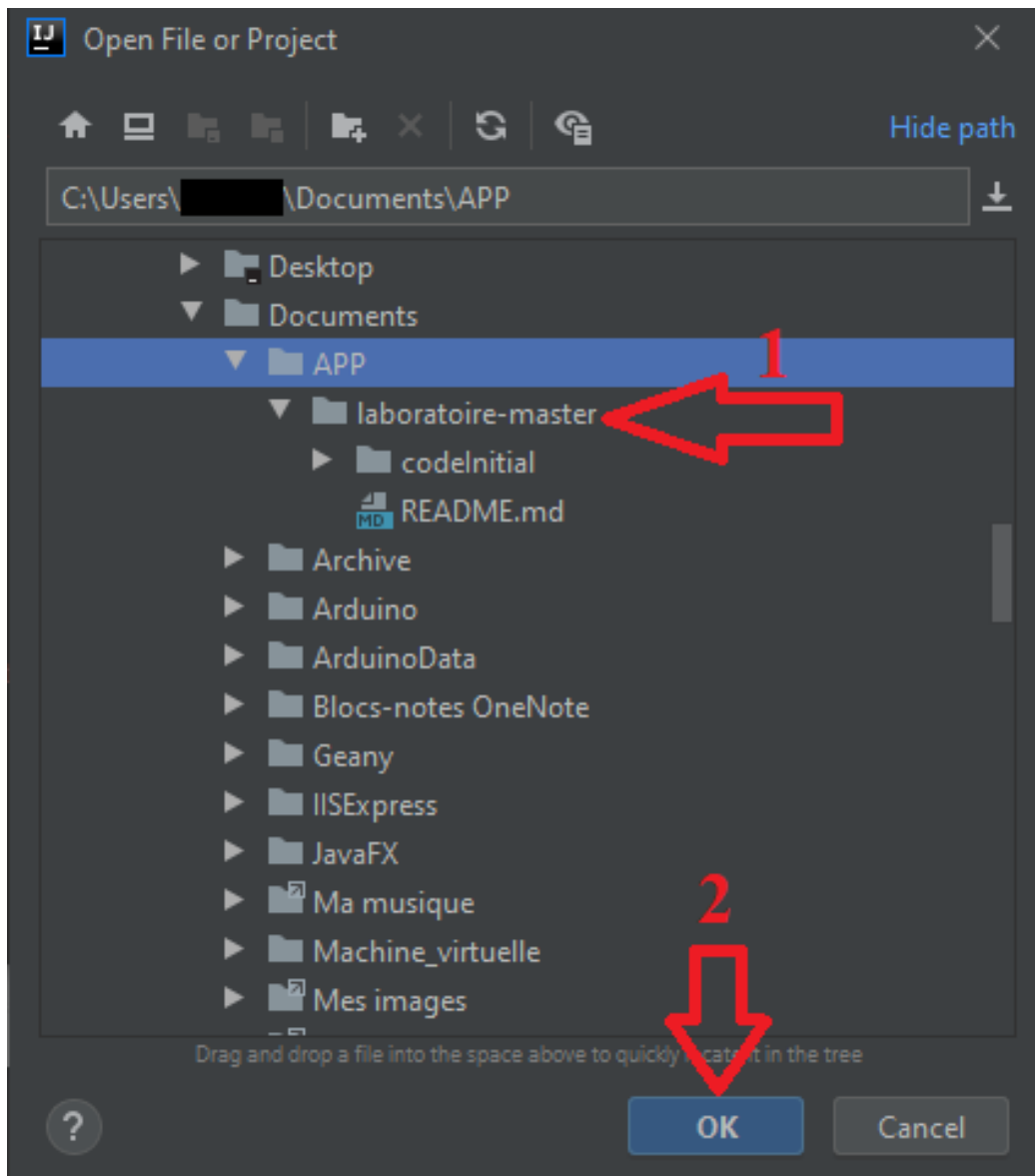
2. Cliquez sur le bouton *Open or Import*



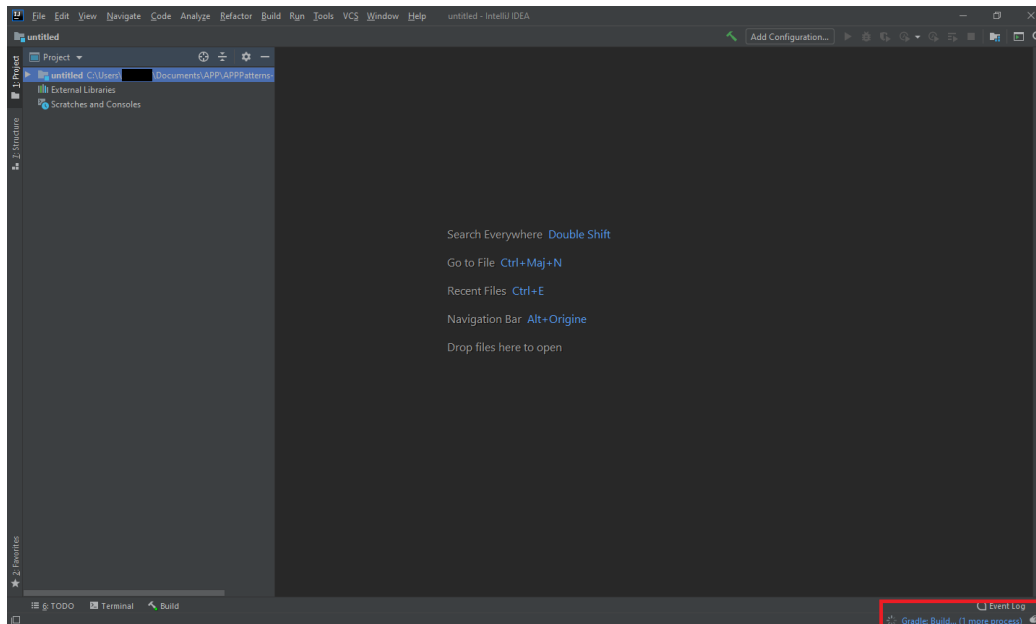
3. Une fois cliquée, une fenêtre contextuelle, comme suit, devrait s'être ouvert. Allez chercher l'emplacement de votre lieu de travail. Il devrait contenir le fichier `codeInitial`.



4. Sélectionnez le fichier nommé `laboratoire-master`. Une fois, sélectionné, cliquer sur *OK*.

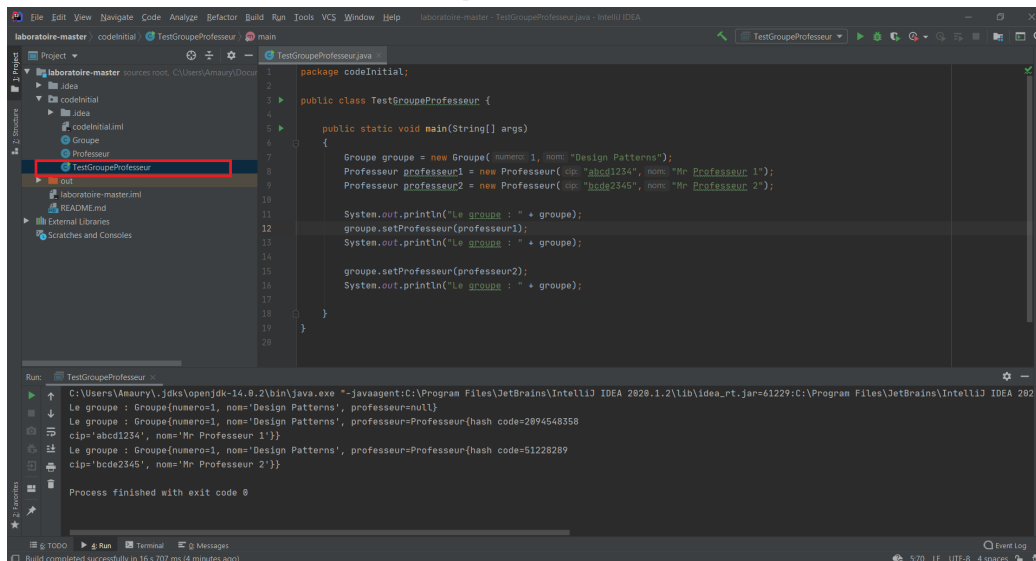


5. Attendez que le Gradle:Build termine de faire la compilation de votre code

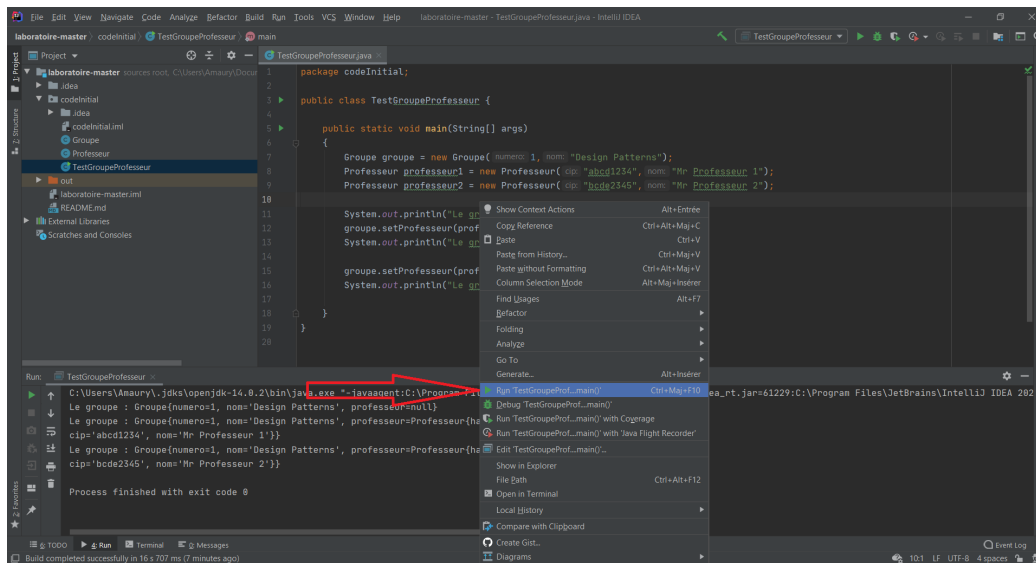


Les fichiers avec lesquels vous allez travailler sont dans le fichier main. Il est important que vous ne touchiez pas les autres fichiers.

- Pour exécuter un fichier, il faut que le fichier java contienne une fonction **public static void main(String[] args)**. Dans le laboratoire, la classe qui contient la dite fonctions s'appelle **TestGroupeProfesseur**.



- Pour exécuter un fichier, on clique droit dans le fichier qu'on veut exécuter. Ensuite on clique le bouton Run `nom_du_fichier.main()` et c'est fait.



E.5 Programmation du Modèle de Conception Singleton

Vous avez le code de base de l'exercice du procédural. Vous pouvez créer un nouveau package pour programmer votre solution avec le Modèle de Conception Singleton.

E.6 Ajout de la JavaDoc

À partir du code réaliser à l'exercice E5, assurer de faire l'ajout de la JavaDoc (voir la référence ci-après).

Lien en référence : <https://www.jetbrains.com/help/idea/working-with-code-documentation.html#generate-javadoc-dialog-controls>

E.7 Programmation de tests unitaires

À partir du code réaliser à l'exercice E5, effectuer l'intégration de tests unitaires (il sera important d'ajouter les tests dans un package distinct). La **couverture de code** attendue pour les tests unitaires est de **100%**.

Lien en référence : <https://www.jetbrains.com/help/idea/tests-in-ide.html>

16 PRATIQUE PROCÉDURALE 2

But de l'activité

- Connaître et implementer les Modèles de Conception suivants :
 1. State
 2. Observer
 3. Factory
 4. MVC (Modèle-Vue-Contrôleur)

16.1 Exercices

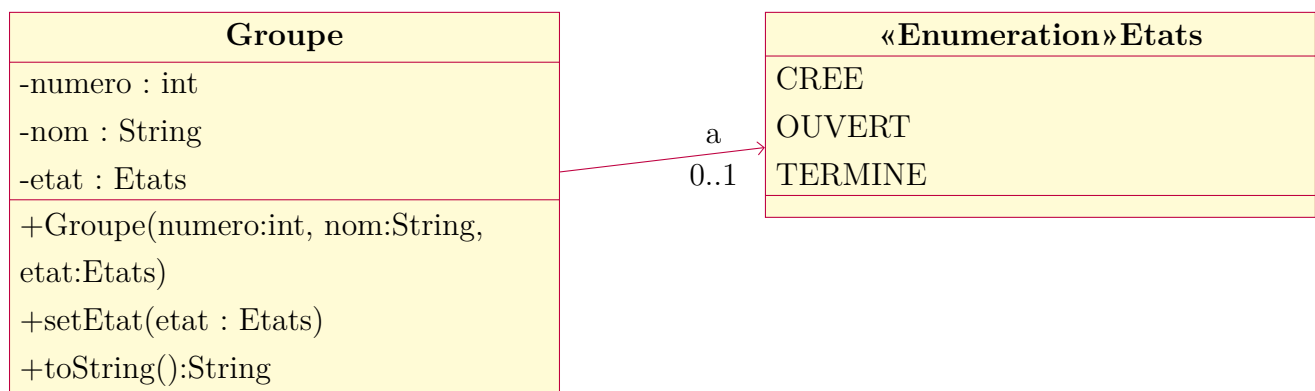
E.1 Modèle de Conception : State

Quel est le problème que peut résoudre le Modèle de Conception State ?

Situation : L'école de programmation DP Inc. offre une variété de cours. Le cours peut avoir plusieurs états et il y a des opérations qui peuvent se réaliser ou pas selon l'état du cours. Les états d'un cours sont :

1. Créé. Quand l'administration de l'école a créé le cours.
2. Ouvert. Quand les étudiants peuvent s'inscrire au cours.
3. Terminé. Quand le cours a été terminé.

Nous avons le diagramme de classe suivant :



Le code qui implemente les classes est :

*Le fichier : **Groupe.java***


```

1 package CodeInitial;
2
3 public class Groupe {
4     private int numero;
5     private String nom;
6     private Etats etat;
7
8     public Groupe(int numero, String nom, Etats etat) {
9         this.numero = numero;
10        this.nom = nom;
11        this.etat = etat;
12    }
13
14    public void setEtat(Etats etat) {
15        this.etat = etat;
16    }
17
18    @Override
19    public String toString() {
20        return "Groupe{" +
21            "numero=" + numero +
22            ", nom='" + nom + '\'' +
23            ", etat='" + etat + '\'' +
24            '}';
25    }
26 }

```

*Le fichier : **Etats.java***

```

1 package CodeInitial;
2
3 public enum Etats {CREE, OUVERT, TERMINE}

```

*Le fichier : **TestGroupe.java***

```

1 package CodeInitial;
2
3 public class TestGroupe {
4
5     public static void main(String[] args) {
6         Groupe groupe = new Groupe(1, "Dessign Patterns", Etats.CREE);
7         System.out.println("Le groupe a été créé est: " + groupe);
8         groupe.setEtat(Etats.OUVERT);
9         System.out.println("Le groupe est ouvert: " + groupe);
10        groupe.setEtat(Etats.TERMINE);

```

```

11         System.out.println("Le groupe termine est: " + groupe);
12     }
13
14 }

```

Quel est le problème potentiel de cette implémentation ?

Faire le diagramme de classes approprié pour garantir la demande d'implémentation avec un Modèle de Conception.

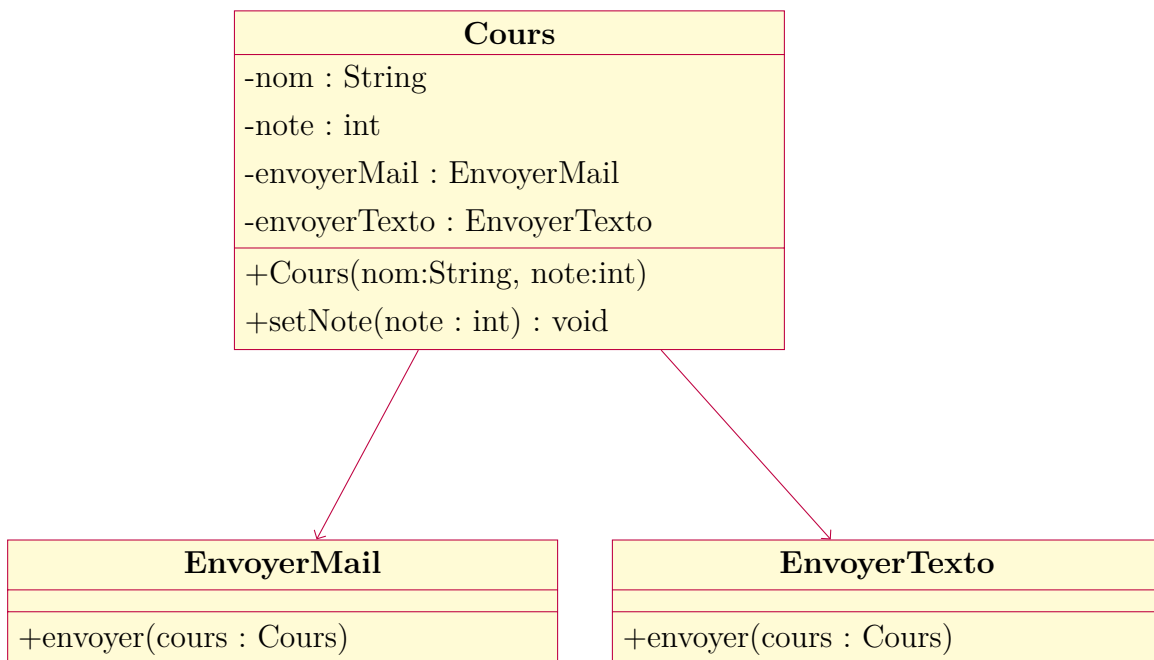
Préparez le code avec un Modèle de Conception approprié pour garantir la demande d'implémentation.

E.2 Modèle de Conception : Observer

Quel est le problème qui peut résoudre le Modèle de Conception Observer ?

Situation : L'école de programmation DP Inc. offre une variété de cours. Les étudiants aiment recevoir de notifications quand une note a changé. Pour l'instant le système permet d'envoyer un courriel ou un message texto quand une note a changée.

Nous avons le diagramme de classe suivant :



Le code qu'implémente les classes est :

Le fichier : Cours.java

```

1 package codeInitial;

```

```

2
3 public class Cours {
4     private String nom;
5     private int note;
6     private EnvoyerMail envoyerMail = new EnvoyerMail();
7     private EnvoyerTextto envoyerTextto = new EnvoyerTextto();
8
9     public int getNote() {
10         return note;
11     }
12
13     public Cours(String nom, int note) {
14         this.nom = nom;
15         this.note = note;
16     }
17
18     public void setNote(int note) {
19         this.note = note;
20         envoyerMail.envoyer(this);
21         envoyerTextto.envoyer(this);
22     }
23 }

```

*Le fichier : **EnvoyerMail.java***

```

1 package codeInitial;
2
3 public class EnvoyerMail {
4
5     void envoyer (Cours cours)
6     {
7         System.out.println( "La note du cours est: " + cours.getNote() + "
8
9     }
10 }

```

*Le fichier : **EnvoyerTextto.java***

```

1 package codeInitial;
2
3 public class EnvoyerTextto {
4
5     void envoyer (Cours cours)
6     {
7         System.out.println( "La note du cours est: " + cours.getNote() + "

```

```

8
9     }
10 }

```

Le fichier : ***TestCoursMailTexte.java***

```

1 package codeInitial;
2
3 public class TestCoursMailTexte {
4
5     public static void main(String[] args) {
6         Cours cours = new Cours("Patrons de Conception", 90);
7         System.out.println("Nouvelle note au cours : 95");
8         cours.setNote(85);
9         System.out.println("Changement de note: 95");
10        cours.setNote(95);
11    }
12
13 }

```

Quel est le problème potentiel de cette implémentation ?

Faire le diagramme de classes approprié pour garantir la demande d'implémentation avec un Modèle de Conception.

Préparez le code avec un Modèle de Conception approprié pour garantir la demande d'implémentation.

E.3 Modèle de Conception : Factory

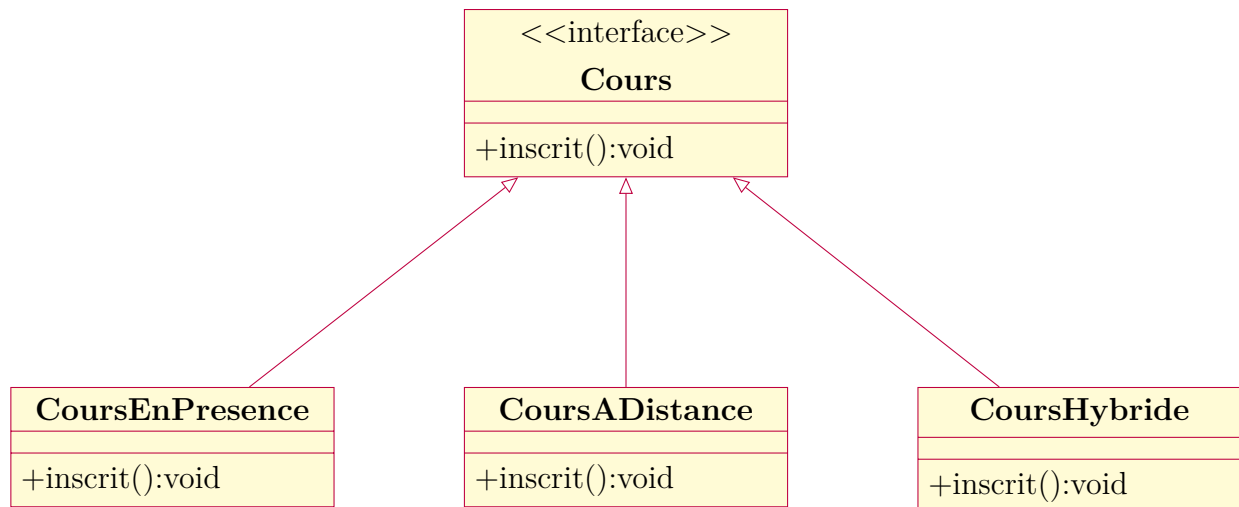
Quel est le problème qui peut résoudre le Modèle de Conception Factory ?

Situation : L'école de programmation DP Inc. offre une variété de cours. À cause de la Covid-19, l'école offre maintenant plusieurs types de cours. Il faut s'assurer que l'application sera capable de supporter ces types de cours et autres qui pourraient s'ajouter plus tard.

Les types des cours à ce moment sont :

1. En présence. Un cours donné aux installations de l'école.
2. A distance. Un cours donné 100% à distance.
3. Hybride. Un cours qui se donne en présence et à distance.

Nous avons le diagramme de classe suivant :



Le code qu'implemente les classes est :

Le fichier : Cours.java

```

1 package codeInitial;
2
3 public interface Cours {
4     void inscrire();
5 }
  
```

Le fichier : CoursADistance.java

```

1 package codeInitial;
2
3 public class CoursADistance implements Cours {
4
5     @Override
6     public void inscrire() {
7         System.out.println("Vous êtes inscrit à un cours a distance");
8     }
9 }
  
```

Le fichier : CoursEnPresence.java

```

1 package codeInitial;
2
3
4 public class CoursEnPresence implements Cours {
5
6     @Override
7     public void inscrire() {
8         System.out.println("Vous êtes inscrit à un cours en presence");
9     }
10 }
  
```

```
9     }
10 }
```

Le fichier : CoursHybride.java

```
1 package codeInitial;
2
3
4 public class CoursHybride implements Cours {
5     @Override
6     public void inscrire() {
7         System.out.println("Vous êtes inscrit à un cours hybride");
8     }
9 }
```

Le fichier : TestCours.java

```
1 package codeInitial;
2
3 public class TestCours {
4
5     public static void main (String[] args) {
6
7         // Faire une inscription a un cours a distance
8         Cours cours1 = new CoursADistance();
9         cours1.inscrire();
10
11        // Faire une inscription a un cours en presence
12        Cours cours2 = new CoursEnPresence();
13        cours2.inscrire();
14
15        // Faire une inscription a un cours hybride
16        Cours cours3 = new CoursHybride();
17        cours3.inscrire();
18
19    }
20 }
```

Quel est le problème potentiel de cette implémentation ?

Faire le diagramme de classes approprié pour garantir la demande d'implémentation avec un Modèle de Conception.

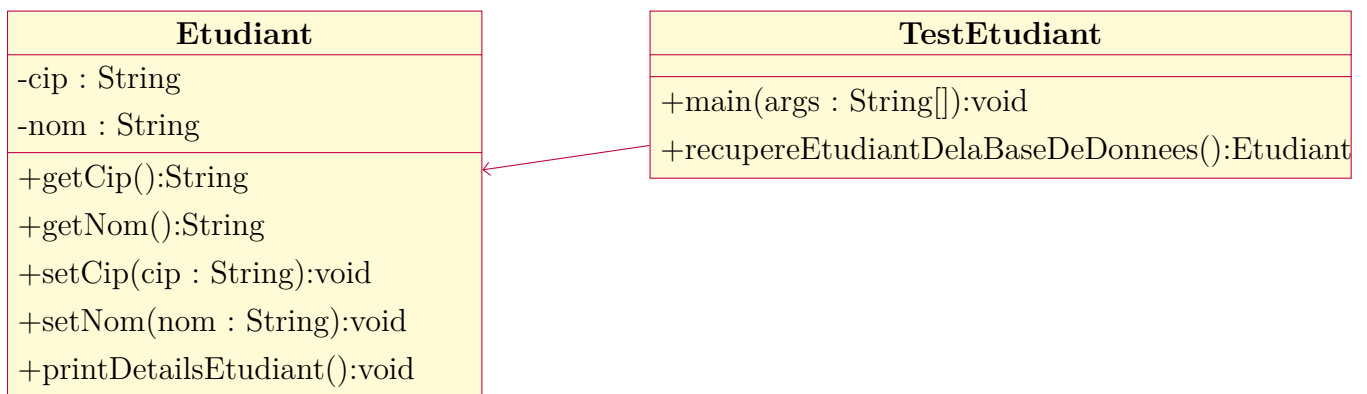
Préparez le code avec un Modèle de Conception approprié pour garantir la demande d'implémentation.

E.4 Modèle de Conception : MVC

Quel est le problème qui peut résoudre le Modèle de Conception MVC ?

Situation : L'école de programmation DP Inc. offre une variété de cours. Il y a plusieurs développements qui se font pour avoir des versions logiciel différentes. Par exemple, on aimerait avoir une application Desktop, une application Web et une application pour des téléphones intelligents. Comme une preuve de concept, nous voulons faire une implémentation MVC avec un contrôleur et une vue simplifiée.

Nous avons le diagramme de classe suivant :



Le code qui implémente les classes est :

*Le fichier : **Etudiant.java***

```
1 package codeInitial;
2
3 public class Etudiant {
4     private String cip;
5     private String nom;
6
7     public String getCip() {
8         return cip;
9     }
10
11     public void setCip(String cip) {
12         this.cip = cip;
13     }
14
15     public String getNom() {
16         return nom;
17     }
18 }
```

```

18
19     public void setNom(String nom) {
20         this.nom = nom;
21     }
22
23     public void printDetailsEtudiant(){
24         System.out.println("Etudiant: ");
25         System.out.println("Name: " + cip);
26         System.out.println("Roll No: " + nom);
27     }
28
29 }

```

Le fichier : ***TestEtudiant.java***

```

1 package codeInitial;
2
3 public class TestEtudiant {
4
5     public static void main(String[] args) {
6
7         //charcher un etudiant de la base de donnees
8         Etudiant etudiant = recupereEtudiantDelaBaseDeDonnees();
9         etudiant.printDetailsEtudiant();
10
11     }
12
13     private static Etudiant recupereEtudiantDelaBaseDeDonnees(){
14         Etudiant etudiant = new Etudiant();
15         etudiant.setCip("abcd1234");
16         etudiant.setNom("Paul");
17         return etudiant;
18     }
19 }

```

Quel est le problème potentiel de cette implémentation ?

Faire le diagramme de classes approprié pour garantir la demande d'implémentation avec un Modèle de Conception.

Préparez le code avec un Modèle de Conception approprié pour garantir la demande d'implémentation.

17 VALIDATION AU LABORATOIRE

Le but de cette activité est de vous permettre de valider la solution proposée à la problématique de cette unité.

- Assurez-vous d’avoir tout fonctionnel et prêt à être validé.
- Une période de temps sera allouée à chaque équipe lors de l’activité de validation.
- Le moment de cette période et une directive sur la logistique de la validation seront publiés sur la page web de l’unité.

Vous devez montrer à un formateur :

- La nouvelle version du diagramme de classes **MenuFact02**.
- Le code de **MenuFact02**.
- L’exécution des tests unitaires.
- L’exécution de l’itération.

A ANNEXE

A.1 Avant-propos

Ce texte est un document de travail rédigé dans le contexte du problème **MenuFact** pour spécifier et documenter le passage de l'itération **MenuFact01** à **MenuFact02**. Certaines parties relatives à l'itération **MenuFact02** sont donc incomplètes ou absentes.

Ce document n'est pas un gabarit de préparation du rapport : pour les contenus du rapport d'APP, il faut se référer au guide de l'étudiant. Le travail réalisé dans le cadre de l'unité d'apprentissage **MenuFact** n'aboutira pas à la confection complète de ce document, mais à la solution de certains éléments clefs définis dans l'énoncé du problème.

Ce document est structuré selon le niveau de détail d'un modèle UML associé aux étapes d'un processus de développement du système :

- Une brève analyse du modèle du domaine ;
- Une analyse partielle de l'application ;
- Une liste de nouvelles fonctionnalités ;

A.2 Analyse du domaine

Cette partie de l'analyse modélise le système dans son contexte d'utilisation, le domaine (certaines méthodologies utilisent le vocabulaire modélisation à métier). Les classes sont définies à haut niveau, généralement sans préciser comment la solution informatique sera développée.

A.2.1 Spécification initiale

Le logiciel **MenuFact** est un système de facturation pour un restaurant qui est à l'état de prototype dont les fonctionnalités sont très limitées. Pour la problématique, dans les versions des itérations concernées, tous les tests seront définis dans un code implémenté dans une classe spécifique qui simule les interactions d'un utilisateur.

La fonction du logiciel est centrée sur les menus et les factures. Il s'agit de créer, mettre à jour et afficher un menu pour, ensuite, initialiser, mettre à jour, afficher, fermer et régler et réinitialiser une facture. Le logiciel doit afficher les menus et les factures dans un seul fichier de sortie et les messages issus de l'exécution sont affichés sur le terminal (la console). L'interface graphique sera développé ultérieurement.

A.2.2 Modèle de classes d'analyse préliminaire du domaine

Plusieurs classes sont identifiées pour la description au niveau du domaine :

Utilisateur

Dans le contexte de **MenuFact01** à **MenuFact02**, il n'y a pas de rôle d'utilisateur externe en interaction autre que celui de contrôler le programme de test. Les interactions aux classes sont générées par la classe de test.

Application

Programme exécutant l'interaction avec les commandes des utilisateurs et les objets traités. L'application n'est pas vue comme une classe dans ce projet.

Classe **TestMenuFact**

Pour les itérations **MenuFact01** et **MenuFact02**, l'application est constituée essentiellement d'une classe de test qui implémente et dirige des séquences de test programmées pour les classes **Menu** et **Facture**. Les scénarios de test envoient les résultats à la console. Ces classes se nomment **TestMenuFact01** et **TestMenuFact02**. Il peut y avoir autres au besoin.

Classe **Menu**

Le **Menu** centralise l'information relative aux **Plats au menu**, qu'ils soient ordinaires ou bien de la catégorie dérivée **Plat santé**.

Le **Menu** comprend une description et une liste de plats choisis. À ce niveau du modèle, on ne spécifie pas comment la liste est mise en œuvre ; on emploie ici le terme liste au sens abstrait, car il est spécifié dans les contraintes d'implémentation de **MenuFact02** que la mise en uvre du Menu soit basée sur une structure d'un **ArrayList**.

Les opérations associées au **Menu** sont :

1. Ajouter des **Plats au menu** en spécifiant un code, une description, un prix ;
2. Afficher le **Menu** complet ou une sélection selon un code (affichage en format texte de tous les attributs des **Plats au menu**) ;
3. Extraire l'information d'un **Plat au menu** identifié par son code.
4. Modifier le menu

Classe **Plat au menu**

Un **Plat au menu** est caractérisé par plusieurs attributs initialisables et modifiables : un code, une description (texte), et un prix (nombre avec fractions).

Un **Plat au menu** doit être affiché au complet lorsque cette opération est invoquée pour l'affichage du **Menu**. S'il s'agit d'un **Plat santé** (catégorie spéciale de **Plat au menu**), alors, tous les attributs spécifiques de **Plat santé** doivent être affichés aussi.

Classe **Plat santé**

C'est un **Plat au menu** ayant des attributs supplémentaires procurant des informations sur les valeurs diététiques du plat. Comme attributs spécifiques, on ne retient que : calories (Kcal), gras (g), cholestérol (mg).

Un **Plat santé** doit être affiché comme tout **Plat au menu**, avec ses attributs spécifiques en plus, pour l’affichage du **Menu**.

Classe Facture

La **Facture** a une liste de plats choisis.

La **Facture** contient plusieurs attributs :

1. Description (champ de texte, par exemple l’identification de la table) ;
2. La date (champ de type Date) du système au moment de l’initialisation de la Facture ;
3. Une liste des **Plats choisis** sous forme de références à des **Plats au menu** ainsi que leur quantité.

Plusieurs opérations sont associées à la **Facture** au niveau du domaine :

1. Initialiser la **Facture** ;
2. Ajouter un **Plat choisi** selon un code du **Menu** en spécifiant la quantité ;
3. Afficher la **Facture** ;
4. Faire des modifications sur le contenu de la **Facture** en parcourant les **Plats choisis** ;
5. Fermer la **Facture**, faire l’addition ;
6. Ré-ouvrir la **Facture** ;
7. Régler (payer) la **Facture**.

Une **Facture** fermée est affichée de manière à faire apparaître les montants de taxes et le sous-total sans taxe et le total avec taxes. Le règlement de la **Facture** est effectué par l’opération régler (payer) la Facture. L’affichage d’une facture indique si elle est ouverte, fermée ou payée.

Classe Plat choisi

Un **Plat choisi** est un élément de la liste associée à une **Facture**. Un **Plat choisi** fait référence à un **Plat au menu** et comprend un attribut spécifique : la quantité.

L’affichage de la **Facture** fait apparaître la quantité d’un **Plat choisi**. Un **Plat choisi** doit être affiché de manière sommaire (seulement le code et le prix) lorsque cette opération est invoquée pour l’affichage d’une facture.

A.2.3 Modèle d’états du domaine

Facture

Une **Facture**, une fois initialisée se trouve dans l’état *Ouverte* ; on peut alors faire les opérations telles qu’ajouter un **Plat choisi**, afficher la facture, sélectionner un **Plat choisi**, retirer un **Plat choisi** (sélectionné), fermer la facture, ouvrir la facture.

L’opération fermer la facture fait passer la **Facture** à l’état *fermée*. Une **Facture fermée** peut être ouverte à nouveau et revenir à l’état *ouverte*.

Une **Facture** *fermée* passe à l'état *payée* lorsque le règlement de **Facture** est effectué. Une fois dans l'état *payée*, elle ne peut plus être ré-ouverte.

Une **Facture** *fermée* ou *payée* est affichée de manière à faire apparaître les montants du sous-total sans taxe, des taxes et du total avec taxes.

A.2.4 Analyse de l'application

Cette partie de l'analyse s'intéresse à la l'application informatique et apporte donc plus de détail sur la spécification de l'application.

Modèle de classe d'application MenuFact01

Voici le diagramme de classes initial de l'itération **MenuFact01**.

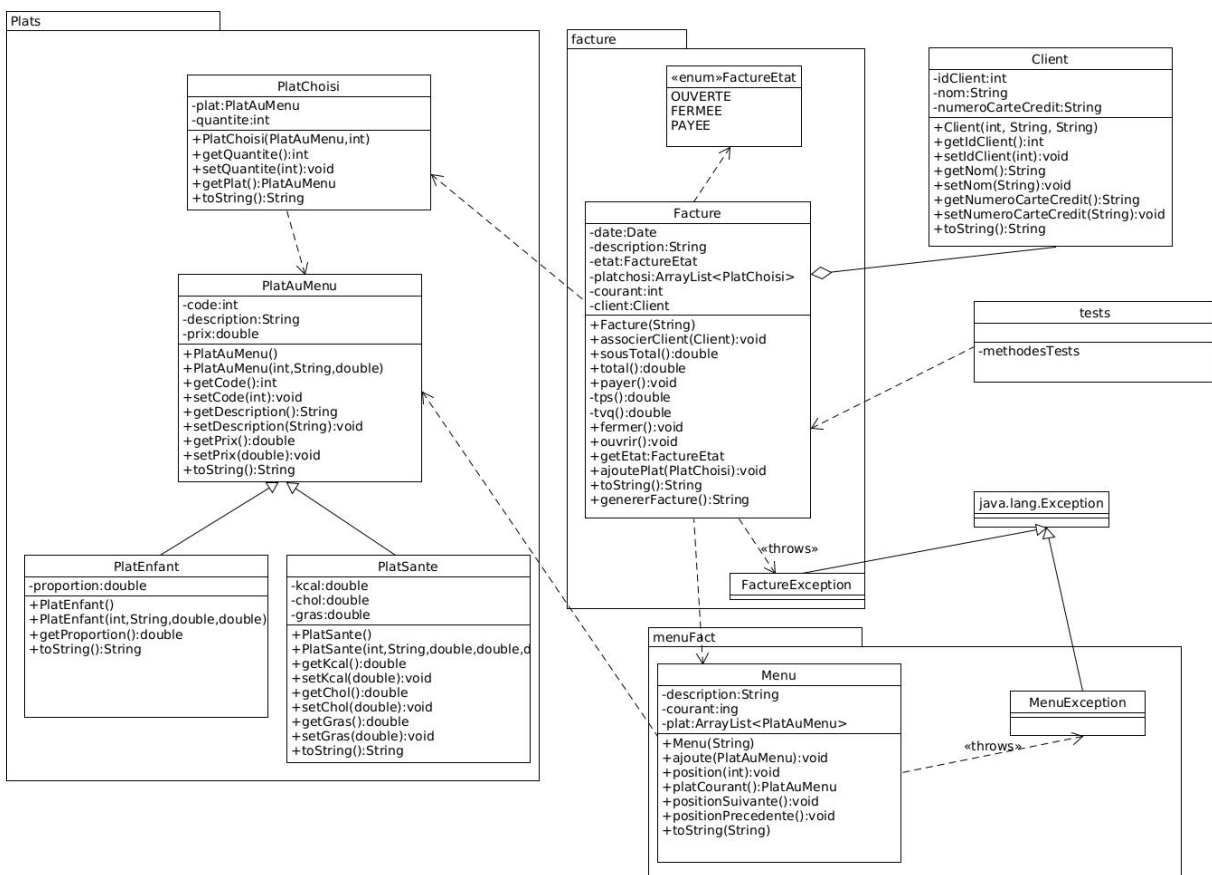


Figure A.1 Diagramme de Classes.

Ce modèle est disponible pour le logiciel Umlet sur le site de l'APP.

A.3 Nouvelles fonctionnalités

Voici une liste de nouvelles fonctionnalités à ajouter à l'itération **MenuFact01** pour l'itération **MenuFacto02**.

Il faut préciser que cette nouvelle version doit nécessairement adapté à l'utilisation des **Modèles de Conception** .

A.3.1 PlatEnfant

Il faut implementer un nouveau type de **Plats au Menu**, les **Plats Enfant**. Ces plats doivent indiquer la proportion (valeur numérique avec fractions) d'un plat "normal".

A.3.2 Ingrédients

Il faut implementer une structure d'héritage pour les ingrédients des **Plats au Menu**.

Les types à considerer sont :

- Fruits
- Legumes
- Viande
- Laitier
- Épice

Voir l'énumération **TypeIngredient**.

Il faut garder l'inventaire des ingrédients pour pouvoir valider si la préparation des plats est possible avant de faire l'ajout à la **Facture**.

Il faut modifier les Plats pour avoir la composition des ingrédients de chacun. Il faut garder l'ingrédient et la quantité à utiliser.

Il faut faire attention aux unités des ingrédients. Il peut avoir des ingrédients liquides ou solides, alors les unités de mesure vont changer.

A.3.3 Chef

À chaque fois qu'un **Plat choisi** est ajouté à la **Facture**, il faut notifier au **Chef** qu'il faut procéder à la préparation du plat.

A.3.4 États des plats

Suite à l'ajout du **Chef**, il faut modifier les Plats pour garder son état de préparation. Les états des plats peuvent être :

1. Commandé
2. En préparation
3. Terminé
4. Servi
5. Impossible de servir (manque d'ingrédients)

A.3.5 Exceptions

Chaque classe doit faire la gestion des **Exceptions** de manière appropriée.