جـامعـة الحسن الثـاني بالدار البيـضـاء
UNIVERSITÉ HASSAN II DE CASABLANCA

ECOLE NORMALE SUPERIEURE
DE L'ENSEIGNEMENT TECHNIQUE
DE MOHAMMEDIA

ENSET

DÉPARTEMENT MATHÉMATIQUE INFORMATIQUE

Programmation orientée objet Java

Rapport

ACTIVITÉ PRATIQUE :
Héritage, Redéfinition, Polymorphisme, Classes abstraites et interfaces

Réaliser par :
ETOULLALI Ayoub

Professeur :
Mr. BOUSSELHAM Abdelmajid

2ème année II-BDCC

Filière d'ingénieur : Ingénieur informatique, Big Data et Cloud Computing

2022/2023

SOMMAIRE

Introduction

Le langage Java a été conçu pour permettre l'exécution du même code sur diverses plate-formes. En particulier, mais pas uniquement, sur le web. Il y a plusieurs types de programmes Java, dont en particulier les applets Java, qui sont intégrées à des pages web et doivent respecter des règles très strictes pour ne pas risquer de causer des dégâts sur les machines d'innocents surfers, et les applications Java, qui fonctionnent comme d'autres programmes, en local sur une machine, et qui ne sont pas limités comme les applets.

Dans les deux cas, le code Java est "compilé", mais les fichiers résultant de la compilation nécessitent encore une interprétation différente suivant chaque plate-forme: cette opération est réalisée par la JVM (Java Virtual Machine).

EXERCICE 1

On souhaite créer une application JAVA pour la gestion des livres et des adhèrents d'une bibliothèque.

1. Créez une classe Personne avec les attributs privés : nom, prenom, email, tel, et age. Ajoutez le constructeur avec paramètres pour initialiser les différents attributs et la méthode afficher() pour afficher ces attributs.

2. Créez une deuxième classe Adherent qui hérite de la classe Personne et qui contient l'attribut numAdherent et redéfinit la méthode afficher().

3. Créez une troisième classe Auteur qui hérite de la classe Personne, qui contient l'attribut numAuteur et redéfinit la méthode afficher().

4. Créez la classe Livre qui contient un attribut ISBN (entier) et un auteur. Ajoutez également la méthode afficher() qui affiche le ISBN, le titre et les informations de l'auteur.

5. Créez une application qui contient une méthode main() pour tester les différentes classes, dans laquelle :

— déclarez et intentiez un adhèrent ;

— déclarez et instanciez un livre qui est écrit par un auteur ;

— affichez les informations de l'adhèrent et du livre.

Projet

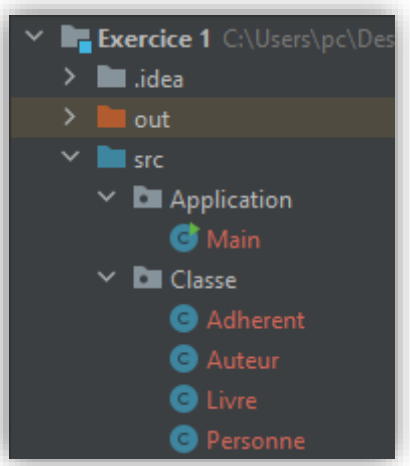
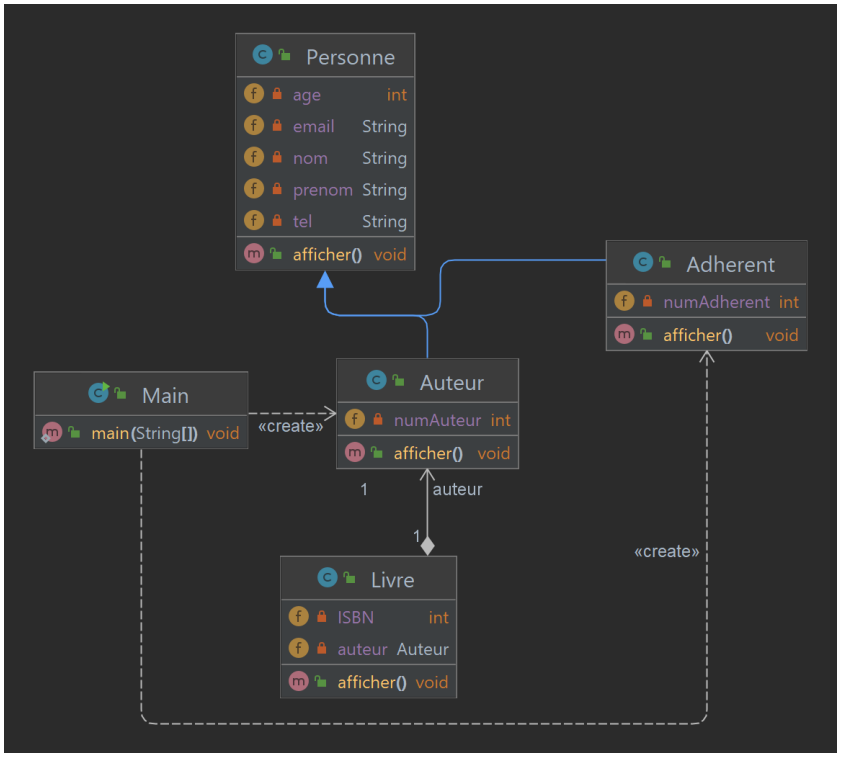


Diagramme de classes



Exécution :

```
C:\Users\pc\.jdks\openjdk-18.0.2.1\bin\java.exe
________Adhérant________
Nom: ETOULLALI
Prenom: radouan
email: radouan@gmail.com
Telephone: +212 6 23 45 78
Age: 30
Numéro Adherant: 1

_________Auteur_________
Nom: ETOULLALI
Prenom: ayoub
email: ayoub@gmail.com
Telephone: +212 6 58 71 20 11
Age: 20
Numéro Auteur: 15

Process finished with exit code 0
```

EXERCICE 2

On souhaite créer une application en java qui permet de gérer les salaires des ingénieurs et des managers d'une entreprise de développement informatique.

1. Créez la classe abstraite **Employe** avec les attributs nom, prenom, email, telephone, et salaire. Ajoutez les constructeurs avec et son paramètres, puis la méthode abstraite calculerSalire() qui retourne le salaire d'un employé.
2. Créez la classe **Ingénieur** avec l'attribut spécialité. Redéfinissez la méthode calculerSalire() sachant qu'on prévoit une augmentation de 15% par rapport à son salaire normal.

3. Créez la classe **Manager** avec l'attribut service. Redéfinissez la méthode calculerSalire() sachant qu'on prévoit une augmentation de 20% par rapport à son salaire normal.
4. Créez une application qui contient une méthode main() pour tester les différentes classes, dans laquelle :
   - — déclarez et intentiez un ingénieur ;
   - — déclarez et intentiez un manager ;
   - — affichez les informations de l'ingénieur et du manager (nom, prénom, salaire, service, et spécialité).

Projet

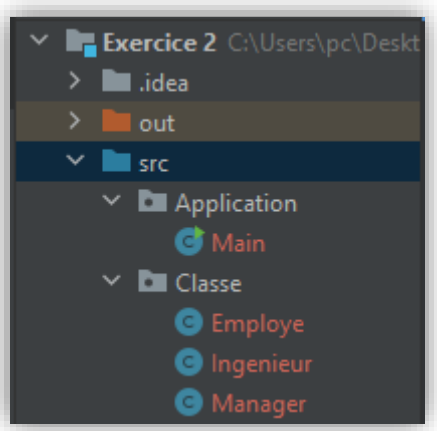
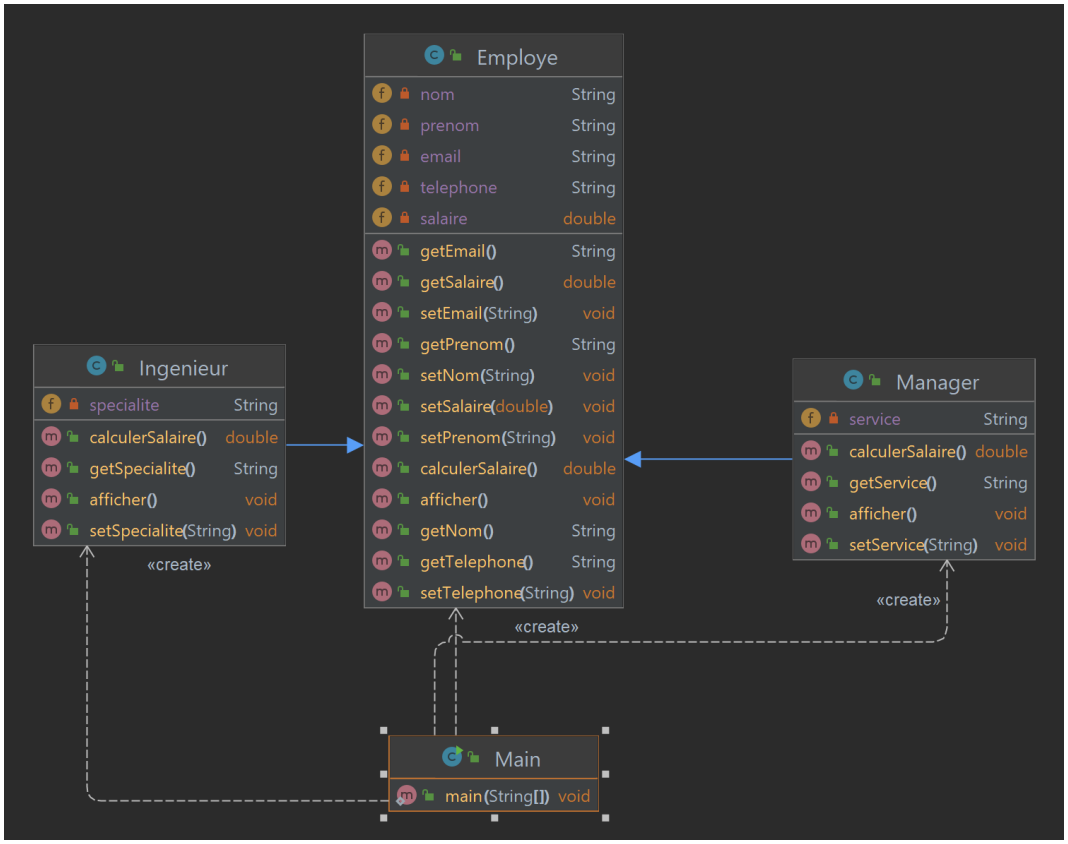


Diagramme de classes



Exécution :

```
Salaire Employe: 10000

Remplir les informations d'un ingénieur :
Nom: ayoub
Prenom: ETOULLALI
Email: ayoub@gmail.com
Telephone: 0658712011
Specialite: Dev


{nom='ayoub', prenom='ETOULLALI', email='ayoub@gmail.com', telephone='0658712011', salaire=11500.0}
{specialite='Dev'}

Remplir les informations d'un manager :
Nom: ihssan
Prenom: FTAH
Email: ihssan@gmail.com
Telephone: 0612345678
Service: Arch


{nom='ihssan', prenom='FTAH', email='ihssan@gmail.com', telephone='0612345678', salaire=12000.0}
{service='Arch'}
```

EXERCICE 3

L'objectif de cet exercice et de concevoir et de réaliser une application JAVA qui gère les commandes des clients d'une entreprise qui vend des ordinateurs. L'application demandée doit donner la possibilité de gérer les ordinateurs, les catégories, et les commandes de l'entreprise.

- Créez une classe **Ordinateur** avec les attributs nom, marque, prix, description, et nombre en stock. Chaque ordinateur appartient à une catégorie. Ajoutez une méthode qui retourne le prix pour une quantité donnée.
- Créez une classe **Catégorie** avec les attributs nom, description et une liste d'ordinateurs. Ajoutez la méthode ajouterOrdinateur() pour ajouter un nouveau ordinateur à la liste (vous devez vérifier s'elle existe déjà avant de l'ajouter), une méthode supprimerOrdinateur() pour supprimer un ordinateur, et une méthode rechercherParPrix() qui retourne la liste des ordinateurs par un prix donné en paramètre.
- Créez une classe **Commande** avec les attributs référence, le client, la date de commande, et l'état de la commande.
- Créez une classe **LigneCommande** avec les attributs quantité, la commande et l'ordinateur commandé.
- Créez une classe **Client** avec les attributs nom, prénom, adresse, email, ville, téléphone, et une liste de commandes effectuées. Ajoutez la méthode ajouterCommande() pour ajouter une nouvelle commande à la liste (vous devez vérifier s'elle existe déjà avant de l'ajouter), et une méthode supprimerCommande() pour supprimer une commande.

Modélisez cette application à l'aide d'un diagramme de classes et implémentez toutes les classes avec leurs attributs. Ajoutez également les constructeurs avec et sans paramètres, les getters, les setters et la méthodes toString pour chaque classe.

Créez une application qui contient une méthode main() pour tester les différentes classes, dans laquelle :

— déclarez et intentiez une liste de trois ordinateurs ;
— déclarez et intentiez une catégorie ;
— déclarez et intentiez un client ;
— déclarez et instanciez une commande du client ;
— déclarez et instanciez une liste de trois lignes de commandes pour la commande et les ordinateurs créés ;
— affichez toutes les informations de la commande.

Exercice 3

Projet

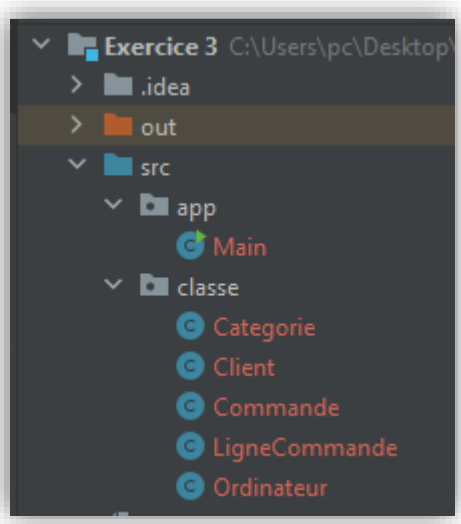
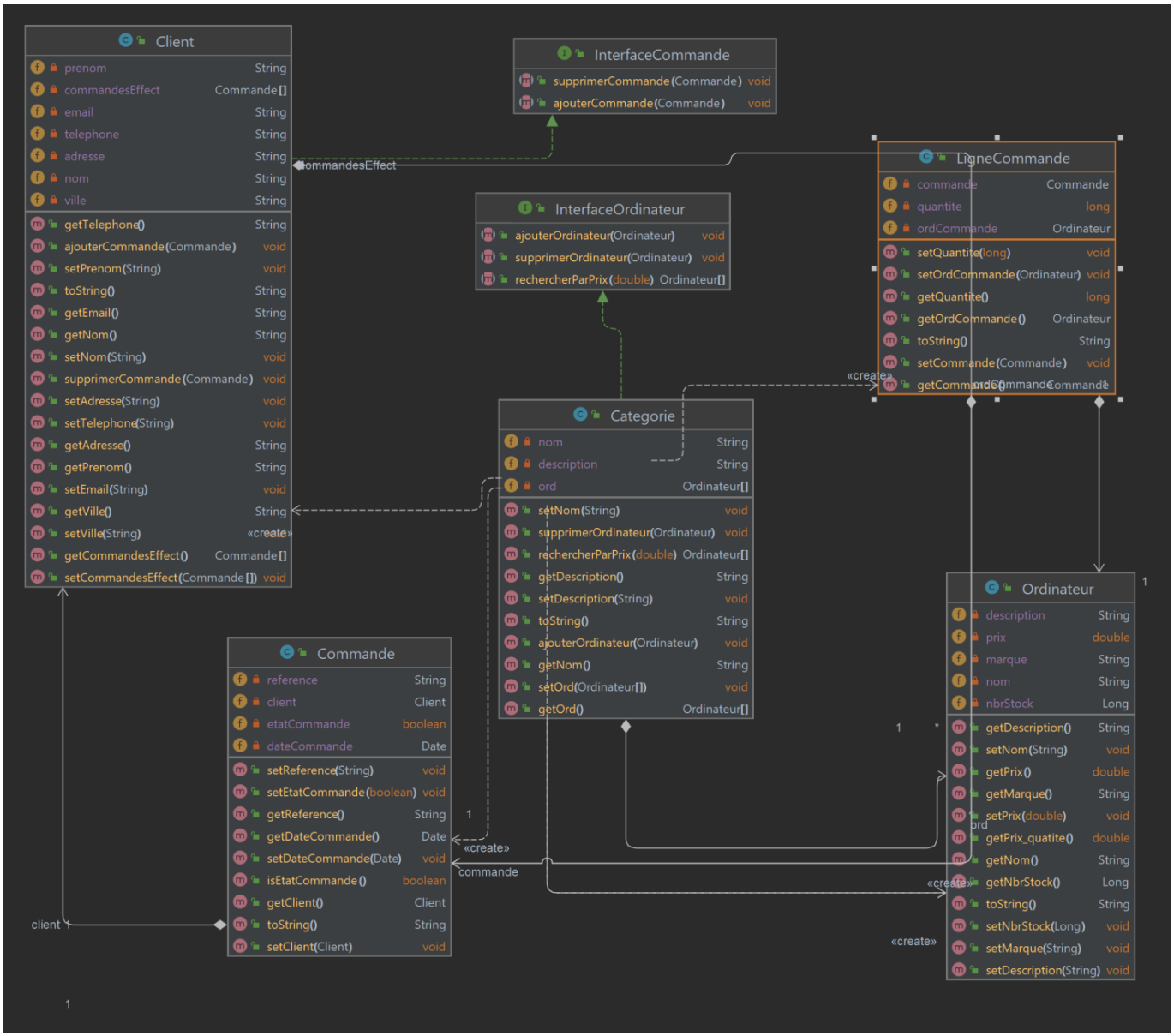


Diagramme de classes



Exécution

```
----------------------------------------------------------------------------------------------------------------------------------------------------------------
**Commande**

reference = mlk9854
*client*
nom = ETOULLALI prenom = ayoub  adresse = lot 6543 address 1    email = ayoub@gmail.com ville = ERRACHIDIA  telephone=+212 6 58 71 20 11    comman
dateCommande = Mon Nov 27 00:00:00 WEST 3922
etatCommande = true
----------------------------------------------------------------------------------------------------------------------------------------------------------------

⇒ ajouterOrdinateur :
Categorie{
nom=''
, description=''
, ord=[Ordinateur{nom='acer', marque='Mklj5625', prix=5000.0, description='bonne marque', nbrStock=546215462}, Ordinateur{nom='dell', marque='lkj7
}


⇒ rechercherParPrix :
Ordinateur{nom='hp', marque='iuyt965', prix=10000.0, description='meilleur marque', nbrStock=58745}

⇒ supprimerOrdinateur :
Categorie{
nom=''
, description=''
, ord=[Ordinateur{nom='acer', marque='Mklj5625', prix=5000.0, description='bonne marque', nbrStock=546215462}, Ordinateur{nom='dell', marque='lk
}

Process finished with exit code 0
```

EXERCICE 4

L'objectif de cet exercice est de manipuler une collection d'objets de type produit en utilisant les listes et les interfaces.

- Créez une classe **Produit** avec les attributs id, nom, marque, prix, description, et nombre en stock.
- Créer une Interface **IMetierProduit** qui va déclarer les méthodes pour gérer nos objets Produit. Cette interface contient les méthodes suivantes :
  - public Produit add(Produit p) : qui permet d'ajouter un produit à la liste.
  - public List<Produit> getAll() : qui retourne les produits sous forme d'une liste.
  - public list<Produit> findByNom(String motCle) : qui retourne une liste de produits dont le nom contient le mot clé passé en paramètre.
  - public Produit findById(long id) : qui retourne un produit par id.
  - public void delete(long id) : qui supprime un produit par nom.

- Créer une classe **MetierProduitImpl** qui implémente l'interface **IMetierProduit**.
- Ecrire une classe **Application** contenant la méthode main qui propose à l'utilisateur dans une boucle while le menu suivant :
  1. Afficher la liste des produits.
  2. Rechercher des produits par mot clé.
  3. Ajouter un nouveau produit dans la liste.
  4. Récupérer et afficher un produit par ID.
  5. Supprimer un produit par id.
6. Quitter ce programme.

Exercice 4

Projet

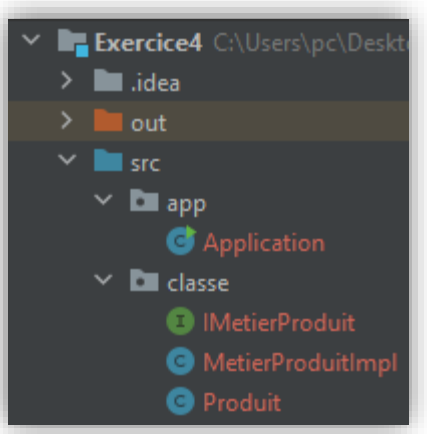
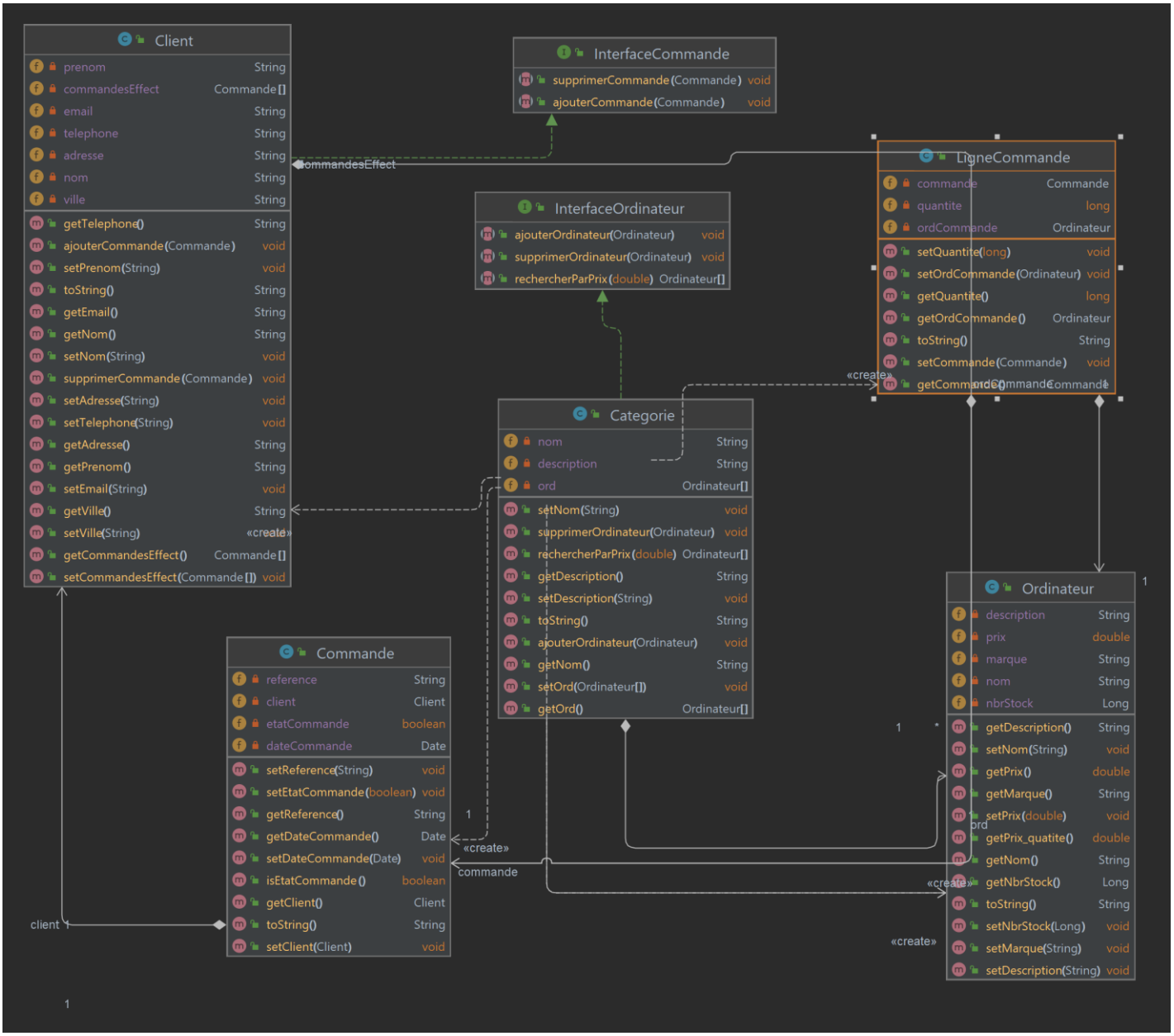


Diagramme de classes



Exercice 4

Exécution

```
-----------------------------------------
1. Afficher la liste des produits.
2. Rechercher des produits par mot clé.
3. Ajouter un nouveau produit dans la liste.
4. Récupérer et afficher un produit par ID.
5. Supprimer un produit par id.
6. Quitter ce programme
-----------------------------------------


choissez un nbr selon votre besoin :
```

```
1. Afficher la liste des produits.
 choissez un nbr selon votre besoin :
 1
 Produit{id=1, nom='produit 1', marque='marque 1', prix=1500.0, description='description 1', nbrStock=15}
 Produit{id=2, nom='produit 2', marque='marque 2', prix=500.0, description='description 2', nbrStock=100}
 Produit{id=3, nom='produit 3', marque='marque 3', prix=2000.0, description='description 3', nbrStock=44}
```

```
2. Rechercher des produits par mot clé.
 choissez un nbr selon votre besoin :
 2
 Entrer un nom:
 produit 1

 Produit{id=1, nom='produit 1', marque='marque 1', prix=1500.0, description='description 1', nbrStock=15}
```

```
3. Ajouter un nouveau produit dans la liste.
 choissez un nbr selon votre besoin :
 3
 setId:
 5
 setNom:
 nom5

 setMarque:
 marque5

 setPrix:
 54
 setDescription:
 des5

 setNbrStock:
 8
 Produit{id=1, nom='produit 1', marque='marque 1', prix=1500.0, description='description 1', nbrStock=15}
 Produit{id=2, nom='produit 2', marque='marque 2', prix=500.0, description='description 2', nbrStock=100}
 Produit{id=3, nom='produit 3', marque='marque 3', prix=2000.0, description='description 3', nbrStock=44}
 Produit{id=5, nom='nom5', marque='marque5', prix=54.0, description='des5', nbrStock=8}
```

Exercice 4

```
4. Récupérer et afficher un produit par ID.

 choissez un nbr selon votre besoin :
 4
 Entrer id:
 3
 Produit{id=3, nom='produit 3', marque='marque 3', prix=2000.0, description='description 3', nbrStock=44}
```

```
5. Supprimer un produit par id.

 choissez un nbr selon votre besoin :
 5
 Entrer id:
 3
 Produit{id=1, nom='produit 1', marque='marque 1', prix=1500.0, description='description 1', nbrStock=15}
 Produit{id=2, nom='produit 2', marque='marque 2', prix=500.0, description='description 2', nbrStock=100}
 Produit{id=5, nom='nom5', marque='marque5', prix=54.0, description='des5', nbrStock=8}
```

```
6. Quitter ce programme

 choissez un nbr selon votre besoin :
 6
 ~~By..
```

```
choissez un nbr selon votre besoin :
10
cette choix est incorrecte !!
```

CONCLUSION :

J'ai appris, au travers de la réalisation des exercices, les fondements du langage Java parce que la maîtrise de ces notions est indispensable pour produire des applications ou des bibliothèques convenables. Néanmoins, pour pleinement profiter des nombreuses autres possibilités offertes par Java, j'ai fait dès maintenant se pencher sur les nombreuses facettes de java.