**RÉPUBLIQUE DÉMOCRATIQUE DU CONGO**

**MINISTÈRE DE L’ENSEIGNEMENT SUPÉRIEUR ET**

**UNIVERSITAIRE**



Projet final : UBuyIRent

**Fait par** : KASEKA MWAVINGI TRESOR

MAKUTA NKYEBO HERVE

Promotion : L3 Informatique

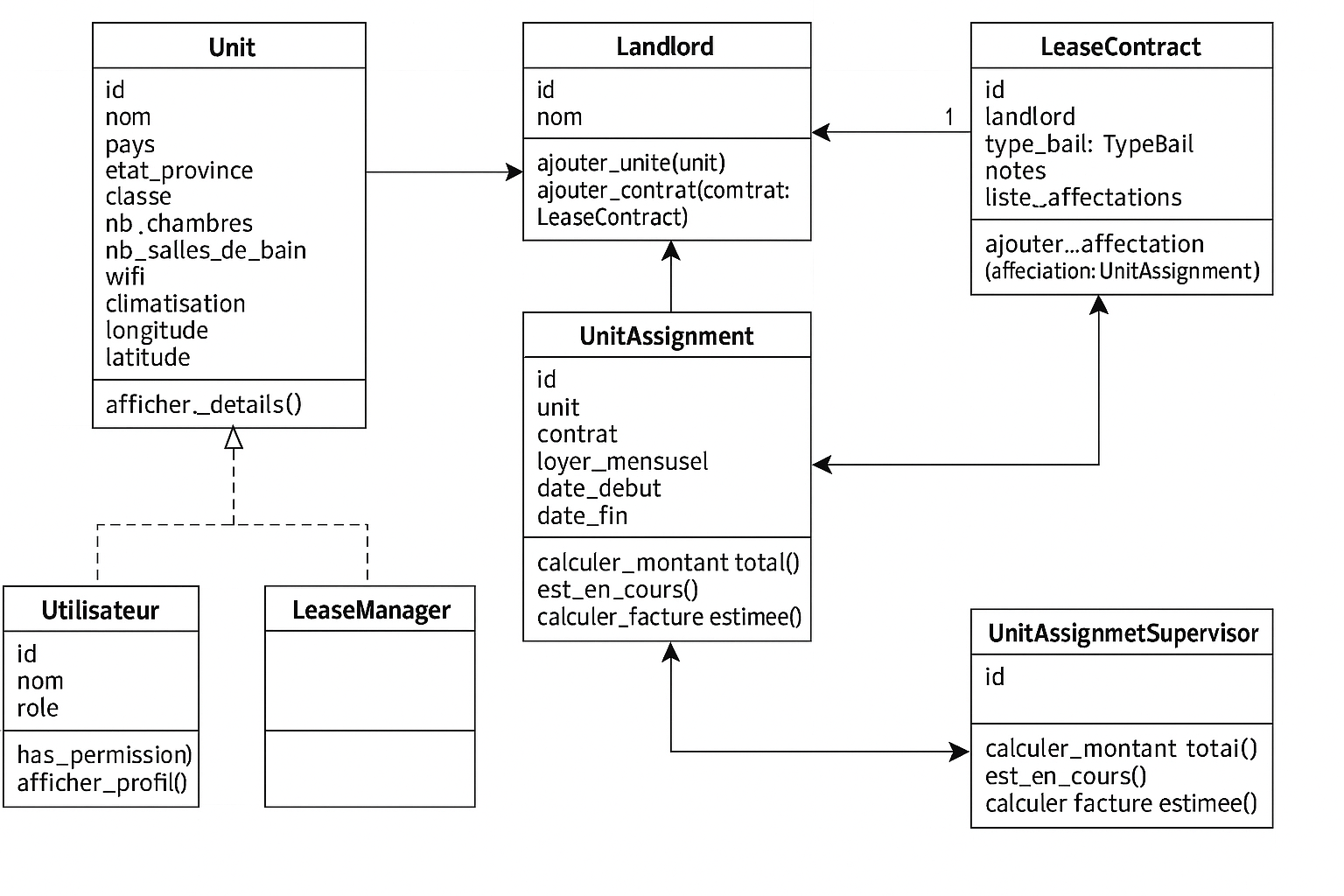
ANNÉE ACADÉMIQUE : 2024 - 2025

# 1. Introduction au problème métier

L’entreprise fictive UBuyIRent gère actuellement ses contrats de location à l’aide d’un simple fichier Excel. Ce mode de gestion entraîne des erreurs fréquentes, des oublis de facturation, et des difficultés de contrôle. Le but du projet est de concevoir un système orienté objet permettant de modéliser correctement les entités métier (unités, contrats, utilisateurs…) et d’automatiser les tâches de gestion locative.

# 2. Architecture de classes UML

Le diagramme ci-dessous représente les classes principales du système ainsi que leurs relations. On retrouve une séparation claire entre les objets métier (`Unit`, `LeaseContract`, `Landlord`...) et la hiérarchie des rôles utilisateurs.

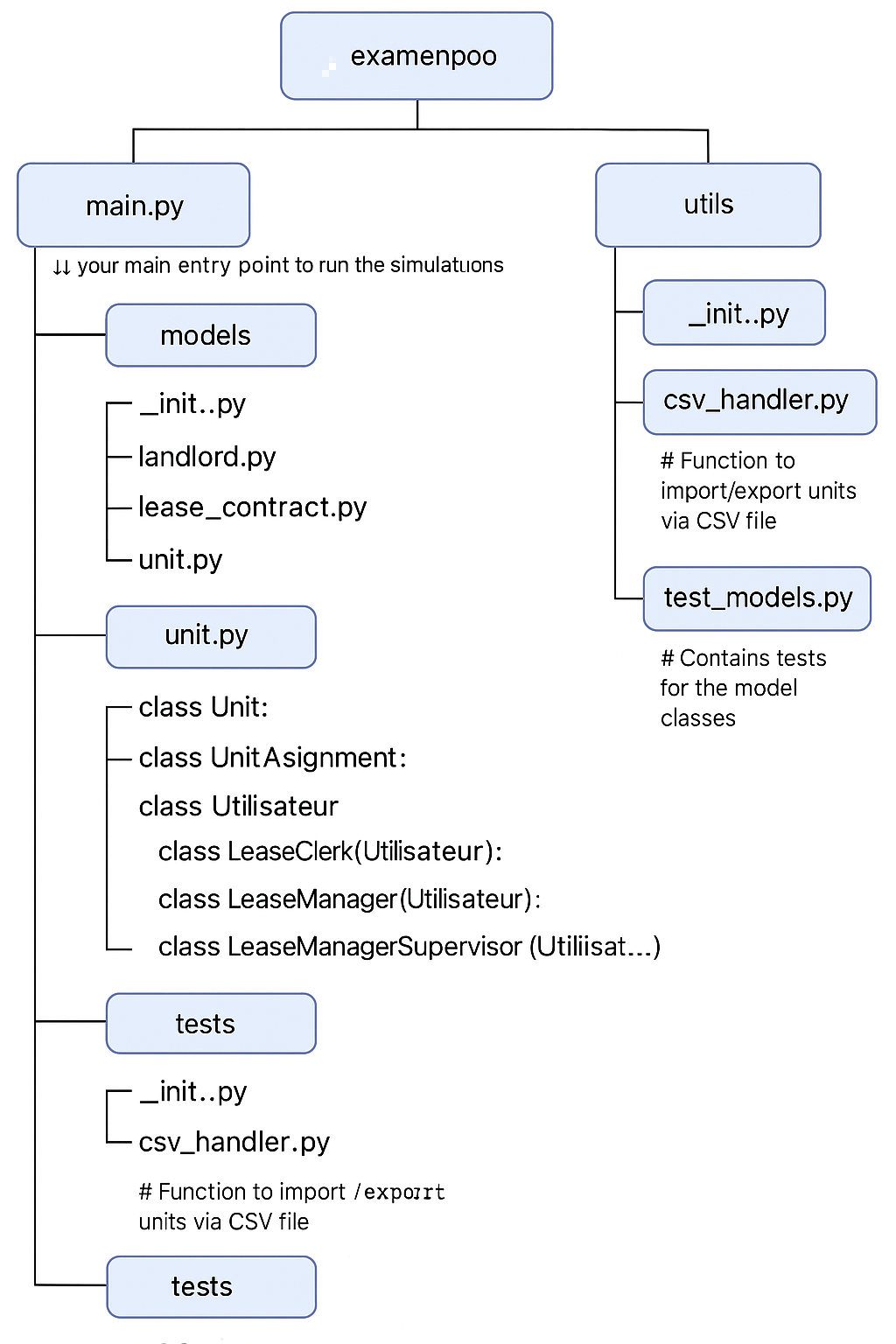


# 3. Justification des choix POO / SOLID

Le projet applique les principes de la programmation orientée objet :  
- Encapsulation : chaque entité possède ses propres attributs et méthodes.  
- Héritage : les rôles utilisateurs héritent de la classe `Utilisateur`.  
- Polymorphisme : les permissions sont redéfinies selon le rôle dans `has\_permission()`.  
- Composition : un `LeaseContract` contient une liste de `UnitAssignment`.

Les principes SOLID sont également respectés :  
- S (Single Responsibility) : chaque classe a une seule responsabilité.  
- O (Open/Closed) : les rôles peuvent être étendus sans modifier `Utilisateur`.  
- L (Liskov) : les sous-classes de `Utilisateur` respectent le contrat de base.  
- I (Interface Segregation) : chaque rôle a ses propres méthodes pertinentes.  
- D (Dependency Inversion) : le système est modulaire et facilement testable.

# 4. captures

Voici la structure de notre projet   


**models/landlord.py**  
  
class Landlord:

def \_\_init\_\_(self, id, nom):

self.id = id

self.nom = nom

self.liste\_unites = []

self.liste\_contrats = []

def ajouter\_unite(self, unit):

if unit.est\_valide():

self.liste\_unites.append(unit)

def ajouter\_contrat(self, contrat):

self.liste\_contrats.append(contrat)

**models/lease\_contract.py**

from enum import Enum

class TypeBail(Enum):

SHORT\_TERM = "Short-Term"

LONG\_TERM = "Long-Term"

STUDENT = "Student"

class LeaseContract:

def \_\_init\_\_(self, id, landlord, type\_bail, notes):

self.id = id

self.landlord = landlord

self.type\_bail = type\_bail

self.notes = notes

self.liste\_affectations = []

def ajouter\_affectation(self, affectation):

if affectation.est\_en\_cours():

self.liste\_affectations.append(affectation)

**models/unit.py**

class Unit:

def \_\_init\_\_(self, id, nom, pays, etat\_province, classe, nb\_chambres, nb\_salles\_de\_bain, wifi, climatisation):

self.id = id

self.nom = nom

self.pays = pays

self.etat\_province = etat\_province

self.classe = classe

self.nb\_chambres = nb\_chambres

self.nb\_salles\_de\_bain = nb\_salles\_de\_bain

self.wifi = wifi

self.climatisation = climatisation

def afficher\_details(self):

return (f"Unit {self.nom}: {self.nb\_chambres} chambres, "

f"{self.nb\_salles\_de\_bain} SDB - Wifi: {self.wifi}, Clim: {self.climatisation}")

def est\_valide(self):

return self.nb\_chambres > 0 and self.nb\_salles\_de\_bain > 0

def generer\_lien\_google\_maps(self):

return f[https://www.google.com/maps/search/?api=1&query={self.nom}+{self.etat\_province}+{self.pays}](https://www.google.com/maps/search/?api=1&query=%7bself.nom%7d+%7bself.etat_province%7d+%7bself.pays%7d)

**models/unit\_assignment.py**

from datetime import date, timedelta

class UnitAssignment:

def \_\_init\_\_(self, id, unit, contrat, loyer\_mensuel, date\_debut, date\_fin):

self.id = id

self.unit = unit

self.contrat = contrat

self.loyer\_mensuel = loyer\_mensuel

self.date\_debut = date\_debut

self.date\_fin = date\_fin

self.date\_prochaine\_facture = date\_debut

self.factures\_precedentes = []

def calculer\_montant\_total(self):

duree = (self.date\_fin - self.date\_debut).days // 30

return duree \* self.loyer\_mensuel

def est\_en\_cours(self):

aujourd\_hui = date.today()

return self.date\_debut <= aujourd\_hui <= self.date\_fin

def calculer\_facture\_estimee(self):

return self.loyer\_mensuel

**models/user.py**

class Utilisateur:

def \_\_init\_\_(self, id, nom, role):

self.id = id

self.nom = nom

self.role = role

def has\_permission(self, action):

raise NotImplementedError("This method should be overridden in subclasses")

def afficher\_profil(self):

return f"Utilisateur: {self.nom}, Rôle: {self.role}"

class LeaseClerk(Utilisateur):

def has\_permission(self, action):

return action in ['create\_unit', 'create\_assignment']

class LeaseManager(Utilisateur):

def has\_permission(self, action):

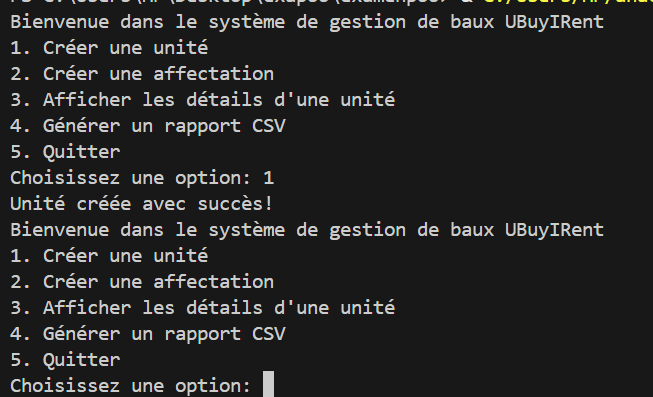
return action in ['create\_unit', 'create\_assignment', 'create\_contract', 'edit\_unit']

class LeaseManagerSupervisor(Utilisateur):

def has\_permission(self, action):

return True # accès complet

et voici le résultat finale



# 5. Jeu de données commenté

Voici un exemple de contenu du fichier `rapport\_units.csv` avec explication :

ID,Nom,Pays,Province,Classe,Chambres,SallesDeBain,WiFi,Clim  
U001,Appart A,France,Île-de-France,Standard,2,1,Oui,Oui  
U002,Studio B,Canada,Ontario,Économique,1,1,Non,Oui

- Chaque ligne représente une unité disponible à la location.  
- Les colonnes indiquent les caractéristiques techniques : localisation, confort, équipements.  
- Ces données sont utilisées pour tester le système et générer des objets `Unit` automatiquement.

# 6. Table des permissions par rôle

|  |  |  |  |
| --- | --- | --- | --- |
| Action | LeaseClerk | LeaseManager | LeaseManagerSupervisor |
| Créer unité | ✔️ | ✔️ | ✔️ |
| Modifier unité | ✖️ | ✔️ | ✔️ |
| Créer affectation | ✔️ | ✔️ | ✔️ |
| Modifier affectation | ✔️ (limité) | ✔️ | ✔️ |
| Créer contrat | ✖️ | ✔️ | ✔️ |
| Supprimer contrat | ✖️ | ✖️ | ✔️ |

# Lien GitHub

[Herve-makuta/POO\_project](https://github.com/Herve-makuta/POO_project)