### **📌 Étapes détaillées pour réaliser le diagramme de classes du projet "Application Mobile de Marché Agricole"**

*(Basé sur les règles des documents UML fournis et en respectant la structure du projet)*

## **🎯 Objectif du diagramme de classes**

Le diagramme de classes permet de **modéliser la structure interne** de l’application en représentant les **entités principales**, leurs **attributs**, leurs **méthodes**, ainsi que les **relations** entre elles.

## **🚀 Étape 1 : Identification des classes principales**

Les **classes principales** sont définies en fonction des fonctionnalités du projet.

### **🔹 Classes essentielles du système**

1️⃣ **Utilisateur** *(Classe parent commune à Agriculteur et Acheteur)*

* id\_user : int
* nom : string
* email : string
* mot\_de\_passe : string
* téléphone : string
* adresse : string
* s\_inscrire()
* se\_connecter()
* mettre\_a\_jour\_profil()

2️⃣ **Agriculteur** *(Hérite de Utilisateur)*

* publier\_produit()
* modifier\_produit()
* supprimer\_produit()
* gérer\_commandes()
* confirmer\_livraison()

3️⃣ **Acheteur** *(Hérite de Utilisateur)*

* rechercher\_produit()
* passer\_commande()
* suivre\_commande()
* payer\_commande()
* évaluer\_produit()

4️⃣ **Produit** *(Représente les articles mis en vente par les agriculteurs)*

* id\_produit : int
* nom : string
* description : text
* prix : float
* quantité\_disponible : int
* catégorie : string
* image : string
* date\_ajout : DateTime
* ajouter\_produit()
* modifier\_produit()
* supprimer\_produit()

5️⃣ **Commande** *(Gère les achats des acheteurs)*

* id\_commande : int
* id\_acheteur : int
* id\_produit : int
* quantité : int
* montant\_total : float
* statut : string (En attente, Confirmée, Livrée)
* date\_commande : DateTime
* valider\_commande()
* annuler\_commande()
* confirmer\_livraison()

6️⃣ **Paiement** *(Gère les transactions entre acheteurs et agriculteurs)*

* id\_paiement : int
* id\_commande : int
* montant : float
* mode\_paiement : string (Mobile Money, Espèces)
* statut : string (Validé, En attente, Échoué)
* effectuer\_paiement()
* rembourser()

7️⃣ **Messagerie** *(Permet la communication entre acheteurs et agriculteurs)*

* id\_message : int
* id\_expediteur : int
* id\_destinataire : int
* contenu : text
* date\_envoi : DateTime
* envoyer\_message()
* supprimer\_message()

8️⃣ **Avis & Notation** *(Permet aux acheteurs d’évaluer les produits après achat)*

* id\_avis : int
* id\_acheteur : int
* id\_produit : int
* note : int (1 à 5)
* commentaire : text
* date\_avis : DateTime
* laisser\_avis()
* modifier\_avis()
* supprimer\_avis()

## **🚀 Étape 2 : Définition des relations entre les classes**

Les relations doivent être définies en utilisant **les bonnes pratiques UML**.

1️⃣ **Relation d’héritage ("généralisation/spécialisation")**✅ Utilisateur est une **classe parent** de Agriculteur et Acheteur.  
📝 *Notation UML : flèche pleine avec un triangle vers la classe parente.*

2️⃣ **Relations entre entités (associations et cardinalités)**📌 Un agriculteur (1) peut publier plusieurs produits (0..\*) → **Association 1:N**📌 Un acheteur (1) peut passer plusieurs commandes (0..\*) → **Association 1:N**📌 Une commande (1) peut concerner plusieurs produits (1..\*) → **Association M:N**📌 Un paiement (1) est associé à une seule commande (1) → **Association 1:1**📌 Un acheteur (1) peut laisser plusieurs avis (0..\*) sur les produits → **Association 1:N**📌 Un message est échangé entre deux utilisateurs → **Association réflexive N:N**

## **🚀 Étape 3 : Dessiner le diagramme de classes UML**

📌 **Outil recommandé** : Draw.io, Lucidchart, StarUML, Visual Paradigm

### **💡 Instructions pour réaliser le diagramme UML :**

1️⃣ **Créer les classes**

* Dessiner **chaque classe** sous forme de **rectangle UML**.
* Ajouter les **attributs** et **méthodes** avec leurs types.

2️⃣ **Ajouter les relations entre les classes**

* **Héritage :** Flèche avec un **triangle vide** de la sous-classe vers la classe parente.
* **Association :** Ligne simple entre les classes avec **multiplicités (1..*, 0..*)**.
* **Composition (forte dépendance) :** Losange plein côté classe "composite".
* **Agrégation (faible dépendance) :** Losange vide côté classe "composite".

3️⃣ **Vérifier la clarté et lisibilité**✅ **Éviter les croisements inutiles de lignes**.  
✅ **Respecter les conventions UML**.  
✅ **Utiliser les bons symboles pour les relations**.

## **🚀 Exemple de représentation UML attendue (schéma simplifié en texte)**