

## Rapport du Projet de programmation orientée objets

### 1. Introduction

À l'ère de la digitalisation, les services de restauration doivent impérativement moderniser leurs processus pour répondre aux attentes des clients et améliorer leur efficacité. Dans ce cadre, la société **RaPizz** souhaite automatiser la gestion de ses activités de fabrication et de livraison de pizzas à domicile. Le projet vise à développer un système informatique complet qui couvre l'ensemble du parcours client, depuis la prise de commande jusqu'à la livraison, en intégrant la gestion des clients, du catalogue de pizzas, des livreurs ainsi que des règles commerciales spécifiques.

Nous avons abordé le développement de ce projet de manière méthodique, en commençant par une phase de réflexion sur la structure et les besoins du système. Cette réflexion nous a conduits à modéliser le système à l'aide d'un diagramme UML, afin de définir clairement les entités principales — telles que le client, la pizza, la commande et le livreur — ainsi que leurs interactions et responsabilités.

Cette modélisation a ensuite été traduite en code Java, avec la prise en compte des règles métiers essentielles : l'influence de la taille des pizzas sur leur prix, la livraison gratuite en cas de retard, le paiement par solde prépayé, ainsi que la mise en place d'un programme de fidélité

~~théorie et pratique pour concevoir une solution réaliste et adaptée aux besoins de RaPizz, récompensant les clients après l'achat de deux pizzas. Ce projet nous a permis de combiner~~

### 2. Fonctionnalités développées

Le projet est construit autour de plusieurs classes principales, chacune correspondant à un concept du domaine (client, pizza, commande, livreur, etc.).

Voici les fonctionnalités développées :

**Gestion des Clients** - Recherche par téléphone et gestion du solde & fidélité via BaseClients.

- 
- Vérification d'existence avant commande.

**Catalogue des Pizzas** - Menu de pizzas avec nom, ingrédients, prix de base.

- Trois tailles (Naine, Humaine, Ogresse) modifiant le prix.

**Prise de Commande** -Saisie interactive : choix pizza, taille, quantité.

-Confirmation de fin de commande et calcul du total.

**Paiement** -Paiement validé si solde suffisant, sinon annulation.

-Affichage du montant payé et de la date. -

**Livraison** Livraison simulée avec un temps aléatoire. -

Si >30min, commande offerte (prix = 0 €).

**Fidélisation** -1 pizza gratuite toutes les 10 achetées.

-Message de félicitations ou progression affichée.

**Gestion des Erreurs**-Messages clairs : client introuvable, pizza absente, solde insuffisant.

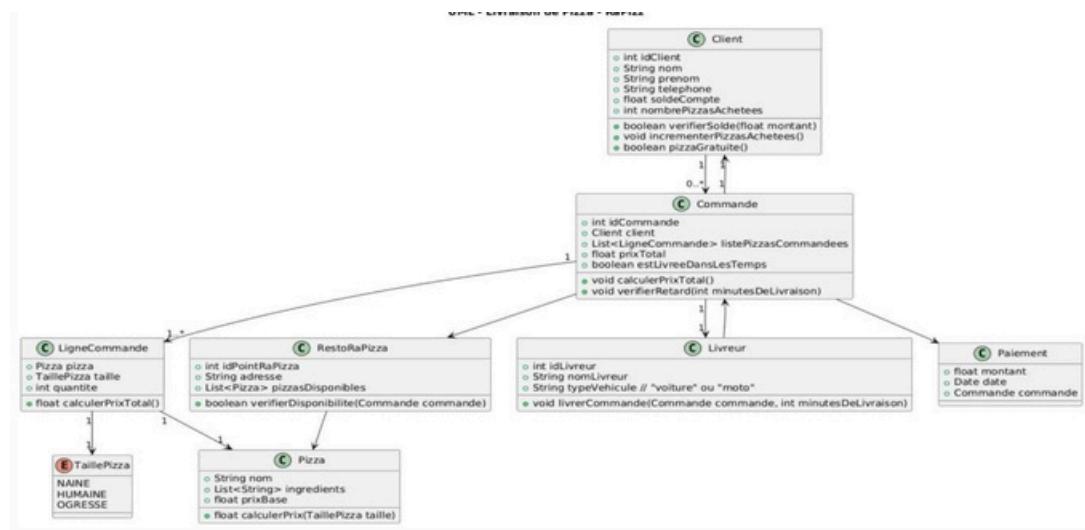
-Saisie sécurisée et guidée.



### 3. Diagramme de classes UML

Pour réaliser le diagramme de classes UML, nous avons utilisé le site **Mermaid Live Editor**, qui permet de créer des diagrammes à partir d'un code simple et facile à écrire.

Après avoir produit une première version du diagramme, nous l'avons soumis à la vérification et aux corrections de notre professeur du TD pour validation. Grâce à ses retours, nous avons pu affiner et améliorer notre modèle pour arriver au diagramme final présenté ci-dessous.



## 4. Conclusion

En conclusion, au cours du développement, nous avons rencontré quelques obstacles, notamment pour bien gérer la question posée au client concernant l'ajout d'autres pizzas à sa commande. Cela a nécessité plusieurs tests et ajustements afin d'assurer une interaction claire et fonctionnelle. Par ailleurs, la gestion du solde client a été améliorée : initialement, rien ne bloquait la commande en cas de solde insuffisant. Nous avons corrigé ce point en ajoutant une vérification stricte avant le paiement, ce qui garantit désormais la fiabilité du processus.

Globalement, le projet s'est déroulé de manière fluide, et nous sommes satisfaits du résultat final, qui répond aux besoins de RaPizz. Nous espérons que notre application offrira une expérience simple et agréable pour passer commande.

