Contexte

MiamMia, un restaurant de plage au charme méditerranéen, réputé pour ses plats exquis et son ambiance chaleureuse.

Mais avec la popularité grandissante du restaurant, la gestion des réservations et des commandes devient un véritable casse-tête pour l'équipe. Entre les clients qui affluent en masse, les réservations oubliées, et les erreurs de commande en pleine heure de pointe, le stress est à son comble !

C'est là que vous intervenez ! La direction de **MiamMia** a décidé de moderniser son service en intégrant un **chatbot intelligent** capable de gérer les **réservations**. Grâce à ce chatbot, les clients pourront réserver une table en un clin d'œil, passer leurs commandes sans erreur et même recevoir des recommandations personnalisées sur les plats du jour. De plus, le chatbot permettra au restaurant d'analyser les tendances des commandes et de mieux gérer son activité.

Votre mission ? Développer ce chatbot révolutionnaire en Java pour simplifier la vie des employés et offrir une expérience fluide et agréable aux clients. Êtes-vous prêt à relever ce défi ?

Exemple d'Utilisation du Chatbot

Scénario

Un client, Luca, souhaite réserver une table pour dîner avec sa famille au restaurant Serena. Il se connecte au chatbot du restaurant via WhatsApp ou le site web.

Conversation type

Luca : Bonjour ! Je voudrais réserver une table pour ce soir.

Chatbot MiamMia: Bonjour Luca ! Combien de personnes serez-vous ?

Luca : Nous serons 4 personnes.

Chatbot MiamMia: Parfait ! À quelle heure souhaitez-vous réserver ?

Luca : 20h.

Chatbot MiamMia: Très bien, votre table pour 4 personnes est réservée pour 20h. Souhaitez-vous consulter notre menu en ligne ou ajouter une commande à l'avance ?

Luca : Oui, je voudrais voir le menu.

Chatbot MiamMia: Voici notre menu du jour : 🍕 Pizza Margherita, 🍝 Pasta Carbonara, 🍹 Cocktail maison. Voulez-vous précommander un plat ?

Luca : Oui, 2 Pizza Margherita et 2 Pasta Carbonara.

Chatbot MiamMia: Merci ! Votre réservation est confirmée et votre commande sera prête à votre arrivée. Bon appétit !

Prérequis

Avant de commencer, assurez-vous d'avoir les connaissances suivantes :

- Programmation en Java (classes, objets, héritage, polymorphisme)
- Gestion des entrées/sorties en Java
- Structures de données (listes, maps, fichiers JSON ou SQLite pour le stockage des conversations et des réservations)
- Notions de bases sur les expressions régulières et le traitement du langage naturel (NLP)

Outils recommandés

- **IDE** : IntelliJ IDEA / Eclipse / VS Code
- **JDK** : Java 11+
- **Frameworks/Bibliothèques** : Gson ou Jackson pour JSON, SQLite JDBC pour la base de données

Prérequis technique

- Un **IDE** (IntelliJ, Eclipse, VSCode)
- Installer un **JDK** si besoin
    - Disponible sur le site d'Oracle.
    - Noter le chemin ou vous l'installez
    - Ajouter le à vos *variables d'environnement* sous le nom JDK_HOME
    - Redémarrez votre IDE
- Créer un nouveau projet JAVA (de préférence avec Maven)
- Ajoutez la dépendence ***json-simple*** à votre projet, vous en aurez besoin

Fonctionnalités du Chatbot

Version de base

1. Accueillir l'utilisateur et lui demander ses besoins (réservation, commande, information).
2. Comprendre des phrases simples et proposer des créneaux de réservation.

3. Enregistrer les réservations et les commandes dans une base de données.
4. Générer un récapitulatif des commandes et réservations.

Version avancée

5. Intégrer une API NLP (ex : OpenAI, Wit.ai, etc.).
6. Supporter des commandes avancées (/help, /exit, /history, /stats).
7. Gérer un système de recommandations basé sur l'historique des clients.
8. Générer des statistiques sur les réservations et la fréquentation du restaurant.

Objectifs

Le programme à 3 objectifs pricipaux :

1. Comprendre des phrases simples et proposer des créneaux de réservation

- Le chatbot doit pouvoir interpréter les phrases des utilisateurs pour détecter une intention de réservation.
    - Exemple de phrases :
        - "Je veux réserver une table pour 4 à 20h."
        - "Avez-vous une disponibilité demain à 19h ?"
- Fonctionnalités attendues :
    - Proposer les créneaux disponibles pour un jour donné.
    - Valider les informations fournies par l'utilisateur (nombre de personnes, heure, etc.).
    - Gérer les erreurs dans les demandes de réservation (ex. créneau non disponible, heure invalide).

2. Enregistrer les commandes des clients et calculer l'addition

- Les commandes sont prises par table, avec un fichier JSON par table (nommé par le numéro de la table).
- Lorsque l'utilisateur demande à voir ou ajouter une commande :
    - Créer ou mettre à jour un fichier JSON pour la table en question.
    - Ajouter les plats, boissons ou menus commandés par les clients dans le fichier.
    - Calculer automatiquement le total des consommations en fonction des prix des produits.
- Exemples de commandes :
    - "Nous voulons 2 pizzas et 3 boissons."
    - "Ajoutez un dessert pour la table 4."
- Fonctionnalités attendues :
    - Récupérer et afficher un récapitulatif des commandes pour une table.
    - Calculer le prix total à payer et archiver les consommations une fois la commande réglée.

3. Ajouter les réservations à venir et archiver les anciennes

- Les réservations sont prises en ligne
- A chaque fois que l'utilisateur demande une réservation :

        - Enregistrer les reservations dans un fichier global (nommé *current.json*)
        - Archiver les reservations passées dans un fichier *archive.json* (y compris celles se trouvant dans le fichier *current.json*)
        - Supprimer les fichiers JSON contenant les réservations passées

Règle de gestion

Le restaurant a quelques petites règles qu'il faut suivre pour ce projet :

- Les dates sont en français au format "*dd/MM/yyyy HH:mm*"
- Les réservations à venir doivent être enregistrés dans un fichier *current.json*, celles passées dans un fichier *archive.json*, les 2 fichiers doivent être dans un dossier *reservation*
- Les codes des menus sont pensé tel que :
    - "Pi" = Pizza
    - "Pa" = Pastas
    - "S" = Secondi (Plat principal en italien)
    - "B" = Bibite (Boisson en italien)
    - "D" = Dessert
    - Un menu SB signifie donc "*Secondi* + *Bibite*", PiBD = "*Pizza* + *Bibite* + *Dessert*"

Produits

Le restaurant donne la liste des produits avec leurs prix et particularités en JSON (voir *products.json*).

Tous les **produits** ont en commun d'avoir un ***nom*** et un ***prix***

- Il y a pour le moment 4 types de produits :
    - Plats (*Dish*)
        - à un type (*PIZZA*, *PASTA*, *SECONDI*, *CONTORNI*)
    - Boissons (*Drink*)
        - peut être alcoolisé ou non
    - Desserts (*Dessert*)
        - à un type (*GELATO*, *CLASSICO*)
    - Menu (*Meal*)
        - un code
        - un plat

        - une boisson
        - un dessert (optionnel)
        - un supplément (0 par défaut)

Contraintes technique

Le programme doit être réaliser en Java. Le but de l'exercice est d'utiliser la **POO** (**P**rogrammation **O**rientée **O**bjet), il faudra donc créer des classes et les utiliser de manière cohérente. Utiliser l'**héritage** et le **polymorphisme** est fortement recommandé.

Le ***main*** de l'application doit être le plus simple possible, il doit refleter clairement le cheminement d'un utilisateur.

Pour rendre votre code plus lisible n'hésitez pas à utiliser une ou plusieurs *classes utilitaire* (nommé **ToolBox** par exemple). Ces classes peuvent contenir des **constantes** ou des **méthode** indépendente décorélés de la partie fonctionnelle de l'appli. Par exemple récupérer le contenu d'un fichier ou formatter une date sont des fonctions assez générique pour être séparé des classes fonctionnelles de l'application.

Etapes (conseillées)

- Afficher un message à l'utilisateur
- Pouvoir lire ce que l'utilisateur saisie
- Controller ce que l'utilisateur saisi et **gérer les erreurs** (s'il saisit n'importe quoi)
- Gestion des reservations
    - Lire un fichier JSON
    - Exploiter et modifier des objets JSON avec la dépendance *json-simple*
    - Ecrire un fichier JSON
    - Gérer le format de date

Conseils

- Prenez votre temps, un pas après l'autre, une fonctionnalité après l'autre
- Tester votre code TRES régulièrement
- Mettez des logs dans votre code
- Mettez des commentaires pour comprendre ce que vous avez fait

- Lisez vos erreurs et retrouvez l'origine avant d'aller sur internet
- Utilisez **intelligement** les ressources à votre disposition (**ne copiez collez pas du code de chatGPT que vous ne comprenez pas**)