INFORME 3 LABORATORIO PARADIGMAS DE PROGRAMACION: CHATBOTS SIMPLIFICADO

Nombre: Ka Hao Zeng Zhong

Sección: B-2

Fecha: 11/12/2023

Tabla de contenidos

Introducción

En el mundo digital actual, los chatbots se han convertido en herramientas esenciales, facilitando desde respuestas rápidas hasta transacciones en línea. Sin embargo, detrás de esa aparente simplicidad, la creación y gestión de estos chatbots es un desafío. El propósito de este laboratorio es explorar cómo pueden ser utilizados de manera eficiente y sencilla. El presente informe corresponde al laboratorio número 3 del curso de “Paradigmas de Programación” el cual trata el paradigma orientado a objetos, donde se utiliza el lenguaje Java. El informe tiene la siguiente estructura: Descripción breve de problema, descripción del paradigma, análisis del problema, diseño de la solución, aspectos de implementación, instrucciones de uso, resultados y autoevaluación, conclusión y, por último, las referencias.

Descripción breve del problema

Se necesita un sistema para crear, desplegar y administrar chatbots estructurados que interactúan mediante opciones predefinidas o palabras clave. Estos chatbots deben poder enlazarse entre sí, guiando al usuario de uno a otro según sus elecciones. La problemática central radica en el desarrollo de una plataforma que no solo facilite la creación de estos chatbots, sino que también los gestione de manera eficiente, manteniendo la fluidez y coherencia en las interacciones. Aunque los chatbots son herramientas fundamentales, su desarrollo presenta diferentes desafíos, particularmente en la creación de una experiencia de usuario intuitiva y una gestión de diálogo coherente.

Usando la programación orientada a objetos y el lenguaje Java, se busca simplificar este proceso. El sistema permitirá a los usuarios configurar un entorno para múltiples chatbots, crear e interactuar con ellos y, al final, recibir un resumen de la conversación. La interacción puede ser a través de comandos o, opcionalmente, una interfaz gráfica similar a las plataformas de ecommerce. Este sistema pretende proporcionar una solución integral para la gestión de chatbots, ofreciendo una plataforma que permita una personalización extensa y una interconexión fluida entre diferentes chatbots.

Descripción breve del paradigma

El paradigma de programación orientado a objetos (POO) es un enfoque de programación que organiza el diseño de software en torno a datos, o "objetos", en lugar de funciones y lógica. Un objeto puede ser definido como una instancia de una clase, que es una plantilla que describe las características y comportamientos (atributos y métodos) asociados con ese tipo de objeto. En POO, los objetos interactúan entre sí para realizar operaciones y manipular datos. Este paradigma se centra en cuatro principios fundamentales:

1. **Encapsulamiento**: Cada objeto mantiene su estado privado dentro de la clase, exponiendo solo los métodos necesarios para el uso de otros objetos.
2. **Abstracción:** Se enfoca en ocultar la complejidad interna y mostrar solo las características esenciales del objeto.
3. **Herencia:** Permite crear nuevas clases basadas en clases existentes, facilitando la reutilización de código y la creación de jerarquías de clases.
4. **Polimorfismo:** La capacidad de un objeto para tomar muchas formas, lo que permite a los objetos ser tratados como instancias de sus clases padre, facilitando la extensibilidad y el mantenimiento del código.

El POO es ampliamente utilizado por su capacidad para modelar entidades del mundo real de manera intuitiva, mejorar la organización del código y facilitar la escalabilidad y el mantenimiento de los sistemas de software.

Análisis del problema

El problema planteado busca construir un sistema para la creación, despliegue y administración de chatbots estructurados. Estos chatbots son singulares porque se basan en interacciones predefinidas, no en análisis avanzados de lenguaje natural. Los requisitos principales para este sistema son crear un espacio unificado que albergue todos los chatbots, permitiendo una gestión y monitoreo centralizado además de facilitar la creación y modificación de chatbots según necesidades específicas, lo que implica una plataforma flexible y adaptable.

También debe existir la posibilidad de conectar diferentes chatbots para crear flujos de diálogo interconectados y dinámicos con los cuales asegurar que los chatbots puedan interactuar con los usuarios a través de opciones predefinidas o palabras clave, adaptándose a diferentes estilos de comunicación. Luego, ofrece una recapitulación o resumen de las interacciones realizadas, lo cual es crucial para análisis y mejoras futuras. Por último, debe proporcionar una interfaz de línea de comandos esencial, con la posibilidad de integrar interfaces gráficas según sea necesario.

Además de estos requisitos principales, es crucial que el sistema sea intuitivo y accesible para los usuarios finales y administradores del sistema. Esto implica una interfaz de usuario clara, documentación completa y facilidades para la depuración y el mantenimiento. Asimismo, el sistema debe ser escalable para soportar una creciente cantidad de chatbots y usuarios, y debe asegurar la privacidad y seguridad de los datos manejados.

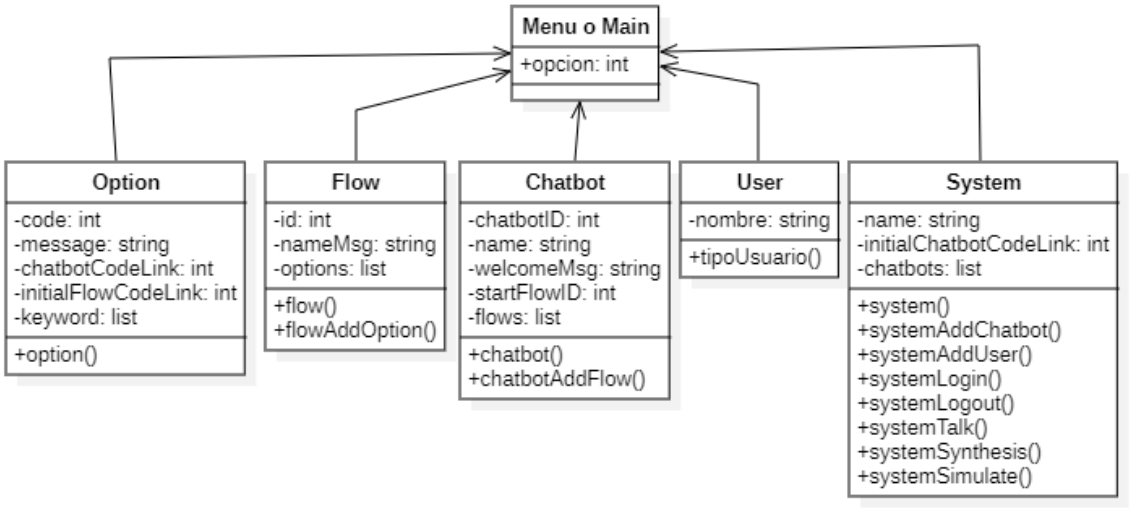


Figura 1: Diagrama UML antes de la implementación de la solución

Diseño de la solución

Para la solución se definieron ciertas clases no solicitadas además de las obligatorias.

- **Option:** Representa una opción dentro de un flujo de un chatbot. Contiene detalles como código, mensaje, enlace a chatbot, enlace a flujo inicial, y palabras clave asociadas.
- **Flow:** Representa un flujo en un chatbot. Un flujo consta de varias opciones y tiene un identificador único y un mensaje asociado.
- **Chatbot:** Representa un chatbot individual. Cada chatbot tiene un identificador único, un nombre, un mensaje de bienvenida, un flujo de inicio y una lista de flujos asociados.
- **System:** Representa un sistema que gestiona múltiples chatbots. Incluye funcionalidades para agregar chatbots, registrar usuarios, manejar sesiones de usuario y procesar mensajes a través de chatbots y flujos activos. También mantiene un historial de chat.
- **AdminMenu:** Proporciona un menú de administrador que permite realizar acciones como crear opciones, flujos, chatbots, sistemas, y mostrar listas de elementos guardados.
- **UserMenu:** Presenta un menú de usuario común que permite realizar acciones como registrarse, iniciar sesión, interactuar con chatbots y ver el historial de chat.
- **User:** Es una clase que controla el flujo del programa basado en el tipo de usuario (administrador o usuario común). Dirige a los usuarios al menú correspondiente según su elección.

- **Main:** Esta es la clase principal que ejecuta el programa. Inicializa datos predeterminados como opciones, flujos, chatbots, y sistemas. También contiene un menú principal que dirige al usuario a diferentes menús según sea un administrador o un usuario común.

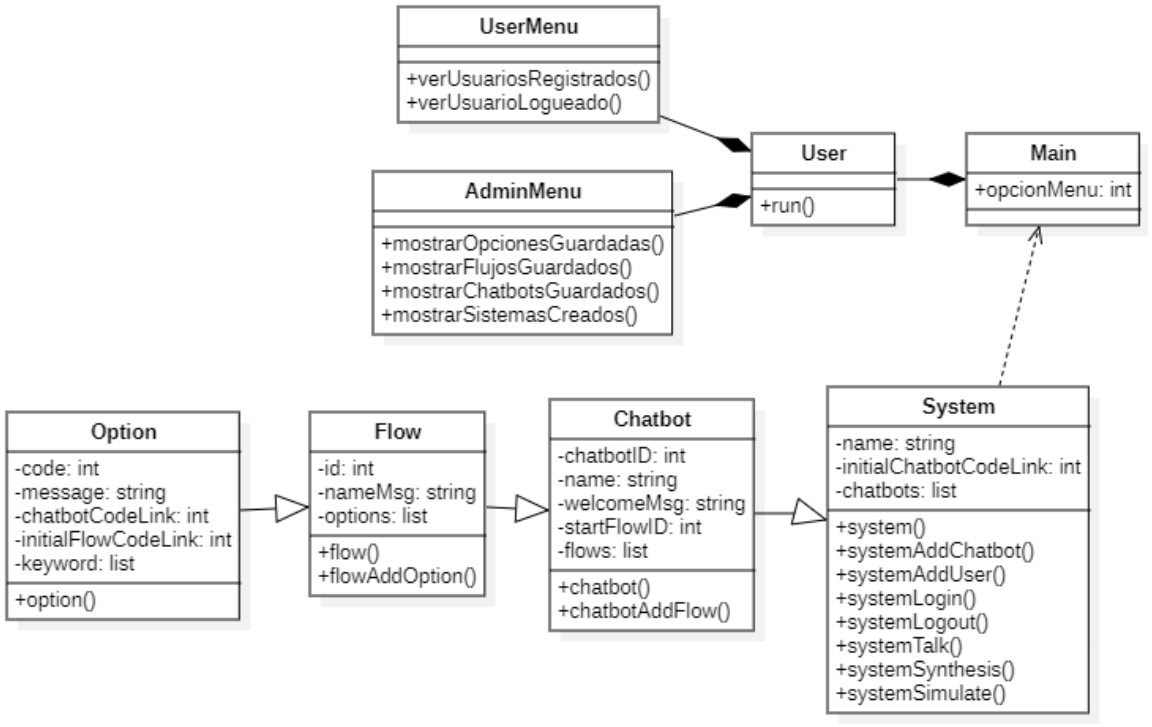


Figura 2: Diagrama UML después de la implementación de la solución

Aspectos de implementación

Para este proyecto fue necesario utilizar un IDE de Java. Existen distintos IDEs como por ejemplo Apache Neatbeans o Eclipse IDE, entre otros, pero para este proyecto en particular se utilizó InteliJ IDEA en su versión 2023.2.5. Igualmente se tenía instalado el JDK (Java Development Kit) en su versión 11 para lograr compilar el programa. Se utilizaron solo librerías estándar de Java, se compiló en el sistema operativo Windows 11 y se integró el proyecto con la herramienta Gradle.

Instrucciones de uso

Instrucciones específicas al anexo del documento.

Resultados y autoevaluación

En cuanto a los resultados obtenidos:

- 0: No realizado
- 0.25: Implementación con problemas mayores
- 0.5: Implementación con funcionamiento irregular
- 0.75: Implementación con problemas menores
- 1: Implementación completa sin problemas

Requerimientos Funcionales	Evaluación
Option	1
Flow	1
FlowAddOption	0,75
Chatbot	1
ChatbotAddFlow	0,75
System	1
SystemAddChatbot	0,75
SystemAddUser	1
SystemLogin	1
SystemLogout	1
SystemTalk	0.5
SystemSynthesis	1
SystemSimulate	0

Al finalizar cada uno de los requerimientos funcionales, éstos fueron probados mediante el uso de variados ejemplos para comprobar su funcionalidad, esto también es evaluado en la tabla anterior ya que, si una prueba daba un resultado inesperado o directamente erróneo, se hace un descuento de puntaje.

Cabe recalcar que en la interacción con el menú se espera que el usuario entregue los argumentos de manera correcta, es decir que por ejemplo si se espera un entero en alguna función no puede ir una cadena. Si esto llega a pasar hay casos en los que el programa se falla o se cae dependiendo de la gravedad del error.

Anadir también que en la mayoría de los requerimientos funcionales se usó como ejemplo las opciones, flujos, chatbots y sistemas ya creados correspondientes al script de pruebas del laboratorio pasado. Por último, no está de más recordar que en todas las funciones de sistema se debe empezar escribiendo el nombre de este mismo teniendo cuidado con las mayúsculas y minúsculas.

Conclusiones

Finalmente, luego de haber construido el programa se puede concluir que se obtuvo aprendizaje sobre un paradigma nuevo además de obtener una experiencia con un resultado parcialmente positivo, ya que se logró conseguir que el programa interactuara con el usuario mediante un menú por consola y que los requerimientos funcionales construidos funcionaran, pero no se lograron construir todos.

En cuanto al POO comparado con los dos paradigmas anteriormente utilizado, es claro que éste es mucho más intuitivo y simple de manejar, al principio podrá ser difícil de entender su composición y el manejo de todos sus conceptos, pero cuando ya se le toma el ritmo, el uso de variables, funciones, clases, atributos y métodos realmente permite una experiencia mucho más amena. Las complicaciones del proyecto fueron que algunas funciones auxiliares para abarcar todos los casos no resultaban bien y por ende, retornaban valores no esperados.

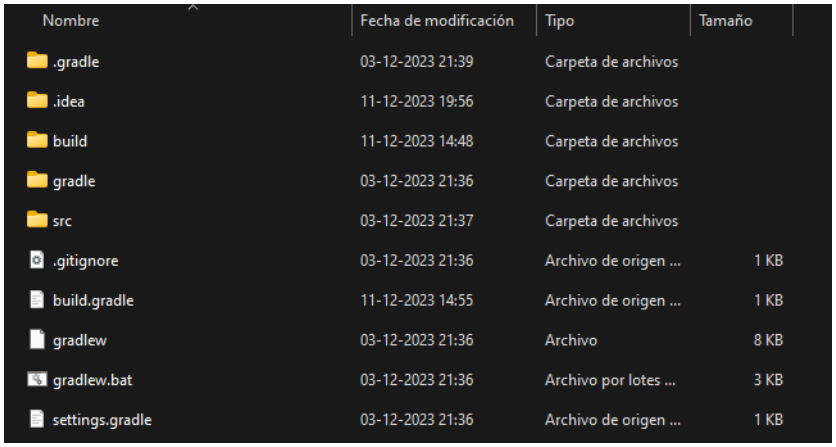
Referencias

1. Martínez, G. (2023). "Proyecto Semestral de Laboratorio". Paradigmas de Programación. Enunciado de Proyecto Online. Recuperado de: https://docs.google.com/document/d/1Psn6YqXfWA99n3AA-rLsvAJsc2-gqj6ot7j90ucLcog/edit

Anexos

Instrucciones seguidas para correr programa de java con gradle:

1. Se debe tener instalado JDK en su versión 11
2. Se buscará la carpeta que contenga el archivo llamado "gradlew.bat"



Nombre	Fecha de modificación	Tipo	Tamaño
.gradle	03-12-2023 21:39	Carpeta de archivos	
.idea	11-12-2023 19:56	Carpeta de archivos	
build	11-12-2023 14:48	Carpeta de archivos	
gradle	03-12-2023 21:36	Carpeta de archivos	
src	03-12-2023 21:37	Carpeta de archivos	
.gitignore	03-12-2023 21:36	Archivo de origen ...	1 KB
build.gradle	11-12-2023 14:55	Archivo de origen ...	1 KB
gradlew	03-12-2023 21:36	Archivo	8 KB
gradlew.bat	03-12-2023 21:36	Archivo por lotes ...	3 KB
settings.gradle	03-12-2023 21:36	Archivo de origen ...	1 KB

3. Se echará a correr desde la consola con los siguientes comandos

```
PS C:\Users\kahao\IdeaProjects\Laboratorio3_20642718_ZengZhong\lab3> .\gradlew.bat build
```

```
PS C:\Users\kahao\IdeaProjects\Laboratorio3_20642718_ZengZhong\lab3> .\gradlew.bat run
```

4. Debería aparecer el siguiente menú

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\kahao\IdeaProjects\Laboratorio3_20642718_ZengZhong\lab3> .\gradlew.bat build

BUILD SUCCESSFUL in 1s
5 actionable tasks: 5 executed
PS C:\Users\kahao\IdeaProjects\Laboratorio3_20642718_ZengZhong\lab3> .\gradlew.bat run

> Task :run
Usuario 'user1' registrado exitosamente en el sistema.
Usuario 'user2' registrado exitosamente en el sistema.
Usuario 'user3' registrado exitosamente en el sistema.
Usuario 'user2' ha iniciado sesion exitosamente.
Como quiere ejecutar el programa?
1) Administrador
2) Usuario Comun
3) Salir
Elija la opcion (1, 2 o 3):
<=========----> 75% EXECUTING [28s]
```

5. De aquí en adelante es intuitivo lo que se debe hacer. Por ejemplo, si se descomentan los systemTalkRec en main y se quiere consultar al historial de chat del sistema “Chatbots Paradigmas” daría como resultado las siguientes imágenes.

```
Usuario: user2igmas
Chatbot: Inicial
Flujo: flujo 1
Respuesta: 1) Viajar
-----------------------------------
-----------------------------------
Usuario: user2
Chatbot: Agencia Viajes
Flujo: Flujo 1 Chatbot1
┬íD├│nde te Gustar├¡a ir?
Respuesta: 1) New York, USA
-----------------------------------
-----------------------------------
Usuario: user2
Chatbot: Agencia Viajes
Flujo: Flujo 2 Chatbot1
┬íQu├® atractivos te gustar├¡a visitar?
Respuesta: 2) Museos
-----------------------------------
-----------------------------------
Usuario: user2
Chatbot: Agencia Viajes
Flujo: Flujo 2 Chatbot1
┬íQu├® atractivos te gustar├¡a visitar?
Respuesta: 1) Central Park
-----------------------------------
-----------------------------------
Usuario: user2
Chatbot: Agencia Viajes
Flujo: Flujo 2 Chatbot1
┬íQu├® atractivos te gustar├¡a visitar?
Respuesta: 3) Ning├╠n otro atractivo
-----------------------------------
-----------------------------------
Usuario: user2
Chatbot: Agencia Viajes
Flujo: Flujo 3 Chatbot1
┬íVas solo o acompa├ ado?
Respuesta: 5) En realidad quiero otro destino
-----------------------------------
### Menu Usuario ###
1. Registrar Usuario
2. Iniciar Sesion
3. Cerrar Sesion
4. Ver usuarios registrados
5. Ver usuario logueado
6. Interactuar con un chatbot
7. Ver el historial del chat
8. Volver al Menu Principal
Seleccione una opcion:
<==========----> 75% EXECUTING [4m 17s]
> :run
```