Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática

Laboratorio N°3

Sistema de Chatbots en Base al Paradigma de Programación Orientado a Objetos

Rodrigo González García

Profesor: Roberto González Ibáñez

Ramo: Paradigmas de la programación

Sección: 13310-0

Tabla de Contenidos.

1. Introducción

En este informe se abordará el proceso de desarrollo del segundo laboratorio de paradigmas de la programación, el cual consiste en crear un sistema de gestión de Chatbots de ITR (Respuesta de Interacción de texto).
Se comenzará con la descripción del problema a resolver, una descripción del paradigma a utilizar (En este caso siendo el Paradigma Orientado a Objetos), seguido del análisis del problema, indagando en la representación de los elementos del sistema y como se trabajaran con estos, posteriormente se analizará el diseño de la solución, presentando el enfoque que se tomó para el desarrollo del problema y cómo se crearon los predicados que componen el programa, se continuará con los resultados obtenidos, junto con una autoevaluación finalizando con las conclusiones de este laboratorio

2. Descripción del Problema.

La problemática por afrontar es la creación de un sistema de creación y gestión de Chatbots de diversos temas.
El sistema debe tener la capacidad de crear y almacenar varios Chatbots, los cuales dentro de ellos deben contener flujos que representan el tema y la dirección que toma la interacción entre el usuario y el sistema, estos Flow a su vez deben poder almacenar distintas opciones, las cuales permiten, a la hora de seleccionarlas, el redireccionamiento a otro flujo o Chatbot, permitiendo cambiar de tema (como por ejemplo, un sistema que representa un centro comercial, donde los distintos Chatbots representen tiendas y los flujos los distintos productos de la tienda).

3. Descripción del Paradigma.

La programación orientada a objetos utiliza clases como su unidad de abstracción, ósea, para la construcción de programas y desarrollo de soluciones, busca transformar objetos de la realidad en clases, tomando solo los elementos útiles para la solución.
Una clase consiste en atributos, los cuales corresponden a las características útiles de un objeto representado en el programa, y métodos, estos siendo equivalente a las funciones en otros paradigmas, o sea, son los procesos que pueden ser llevados a cabo con la clase (Java, Classes, s.f.).
A la hora de crear una instancia de una clase se crea un objeto, el cual tiene su propia dirección de memoria y permite la modificación de sus datos y el uso de sus métodos.
Algunas herramientas de gran utilidad presentes en este paradigma son:
Herencia: Permite utilizar una clase para crear a otra, haciendo que esta herede los métodos y atributos de la clase superior (Java, Inheritance, s.f.)
Polimorfismo: La capacidad de utilizar algo de varias maneras, para explicarlo de mejor manera, hay 2 tipos de polimorfismo de gran importancia, la sobreescritura y sobrecarga, el primero permitiendo redefinir un método de la misma firma (combinación de nombre, aridad y tipos de datos), haciendo que el mismo método haga cosas distintas al ser heredado, y el segundo correspondiendo a la capacidad de generar un método del mismo nombre, pero distinta aridad o distintos tipos de dato de entrada entro de la misma clase o en clases heredadas. (Java, Polymorphism, s.f.)
Es bastante claro que estas 2 características representan una de las mayores ventajas de este paradigma de programación, la capacidad de reutilizar código de manera rápida y efectiva

4. Análisis del Problema

Para comenzar el análisis, vale la pena ver cuáles componentes dan forma al sistema, estos siendo, en orden de mayor a menor nivel: System, Chatbot, Flow, Option, con User y su historial siendo elementos relacionados con el sistema, pero se trabajará con ellos de manera más independiente.
Se sabe que el system debe poseer un nombre, el ID del Chatbot inicial y una lista de Chatbots, a su vez un Chatbot está compuesto por su id, el cual debe ser único dentro del sistema, su nombre, mensaje de bienvenida, el ID de su Flow inicial y su lista de Flows, estos están compuestos por su ID (el cual también debe ser único dentro de su respectivo Chatbot), su nombre y mensaje y una lista de sus respectivos option, para finalizar el análisis de la estructura del sistema los option se componen de un código, el mensaje de la opción, el ID del Chatbot y del Flow al cual se dirigirá la interacción del usuario y una lista de keywords característica de cada opción.
Teniendo en cuenta lo mencionado anteriormente, se representarán cada componente del sistema como una clase.
Cabe destacar que esta vez, los usuarios se deben dividir en usuarios normales y administradores, cada uno con sus propias capacidades, esto se profundizara más adelante.
Debido a que el usuario debe interactuar con el sistema, uno de los elementos más importantes que se deben guardar son las ID del Chatbot y el Flow que están siendo usados actualmente, los cuales deben ser guardados en el sistema y en el Chatbot que se están utilizando, pero esto crea un problema, a la hora de empezar desde cero una interacción (como debe suceder al momento de cerrar sesión con un usuario y abrir sesión con otro) también se debe tener a mano todos los valores iniciales del sistema, es por esto que se deberán guardar los ID que se vayan a cambiar a la hora de interactuar con el sistema.
También cada usuario debe tener vinculado un historial de su interacción con el sistema, lo cual se puede lograr de una manera práctica al tener en cuenta que cada usuario debe estar registrado en el sistema para poder interactuar con él, por esto, el usuario deberá tener un historial base a la hora de añadirse al sistema, lo cual se puede lograr representando al usuario como una clase que posea su historial como atributo, además de su rol dentro del sistema (si el usuario es admin o no).
También se pide la implementación de un menú para permitir que el usuario se comunique con el sistema, lo cual se puede lograr de manera fácil con una clase que utilice a todas las clases anteriormente mencionadas.
El diagrama de análisis se presenta en los anexos (Figura.1)

5. *Diseño de la Solución*

Para comenzar, se utilizará una técnica de diseño Bottom-up, la cual consiste en crear las partes independientes del sistema de manera detallada para luego enlazarlas, formando componentes más grandes (Panigrahi, 2023), además, las clases que se piden implementar y sus métodos requeridos serán implementados mediante interfaces.

Empezando por la clase Option, esta posee como atributos un ID (como int, esto no se especificará de ahora en adelante), un mensaje, el ID del Chatbot al que está vinculado, el flujo a utilizar de este flow y una lista de las keywords disponibles, además, solo se implementan los constructores y selectores de esta clase.

La clase Flow presenta 3 atributos, su ID, su nombre y mensaje dentro de un string y la lista de opciones, los métodos que posee esta clase son su constructor y selectores entre los más generales, además posee el modificado flowAddOption que permite añadir una serie de opciones al Flow, finalmente, también se implementa el método checkidrepeat, que revisa si en una lista está presente un Option que posea un id repetido, para evitar su duplicidad a la hora de añadir Options o crear un Flow.

Continuando con la clase Chatbot, sus atributos son su ID, su nombre y un mensaje como String, un ID del Flow que utiliza, además del ID del Flow inicial, y una lista de Flows, además, los métodos que posee esta clase son los constructores, los selectores, los modificadores chatbotAddFlow, que permite añadir un Flow al Chatbot, resetchatbot, que permite volver a poner el ID del flow actual como el Flow inicial, y setflowactual, que cambia el Flow actual por el ID entregado, finalmente, también tiene su equivalente de checkrepeatid.

Llegando al penultimo nivel de la estructura, la clase System posee un nombre, el ID del Chatbot actual, un espacio para un usuario activo, una lista de usuarios, una lista de chatbots y el chatbot inicial, se implementan los métodos básicos constructor y selectores.

Entre los modificadores están presentes systemAddChatbot, que permite añadir un chatbot al sistema, evitando que se repitan sus ID (utilizando su propia versión de checkrepeatid), systemAddUser, que añade un usuario al sistema (puede ser o admin o un usuario normal, esta diferencia se implementa dentro del menú), systemLogin, que permite que un usuario inicie sesión, systemLogout, que reinicia los chatbots del sistema y cierra la sesión del usuario.

Finalmente, se implementa systemtalk, que permite que el usuario interactúe con el sistema y esta se añade a su historial, esto en base a un mensaje que puede ser un id del option que se busca utilizar o una de sus keywords, y systemsynthesis, que permite que el usuario vea su propio historial.

Adicionalmente, se implementan las clases User y su clase heredada Admin, las cuales poseen los atributos de su nombre, su historial y un booleano que indica si este es o no un administrador y como métodos tienen su constructor, selector y un modificador para añadir al historial.

Como ultima clase se implementa la clase Menú Principal, la cual permite al usuario interactuar con el sistema, esta clase posee un sistema creado a la hora de instanciarse, un Scanner para leer los inputs del usuario y un booleano que indica si se debe seguir corriendo el programa, entre otras variables utilizadas para poder utilizar los métodos de otras clases, solo posee su constructor y el método run, que permite mostrar por consola un menú que, dependiendo del tipo de usuario, permite que este ejecute distintas acciones, si el sistema no tiene usuario, entonces permite añadir usuarios y administradores, además de poder iniciar sesión, si el usuario es un administrador, este puede añadir Chatbpts, Flows, y Options, además de ver la estructura del sistema, finalmente, si el usuario es un usuario normal, entonces este puede interactuar con los Chatbots y ver su historial, ambos usuarios normales y administradores pueden cerrar sesión.

Cabe destacar que, por temas de tiempo, no se pudo implementar el chequeo de los tipos de datos insertados al utilizar el menú, es por esto que el programa parara de funcionar si es que no se le entrega el tipo de dato correcto.

El diagrama de diseño se presenta en el anexo (Figura.2) para entender de manera más visual la implementación

6. *Aspectos de la Implementación*

El sistema operativo utilizado es: Windows 10 Home.
La versión de java JDK utilizada: build 11.0.21+9.
El IDE utilizado es: IntelliJ Build #IU-232.10227.8.
No se utilizaron librerías adicionales además de las default de Java.

7. Instrucciones de Uso

El menú funciona en base de opciones donde se debe ingresar el numero de la opción, como se ejemplifica en la siguiente imagen:

```
Que desea hacer?(ingrese el numero de la opcion):

1)Añadir Usuario
2)Añadir Admin
3)Log-in
4)Salir
```

Imagen.1 Ejemplo de menú Principal.

En este menú se puede ingresar un usuario o administrador ingresando el nombre de este e iniciar sesión, donde se mostrarán los usuarios disponibles y su rol, se debe ingresar el nombre.

```
Con que usuario desea iniciar sesion?
Nombre:  Roland  Es administrador?: No
Nombre:  Carmen  Es administrador?: Si
Carmen
```

Imagen.2 Ejemplo inicio de sesión de Admin.

El administrador puede añadir Chatbots,Flows y Options (en el caso de estos últimos dos se debe indicar su pertenencia indicando la opción del Chatbot al que se desea añadir), en el caso de añadir, se debe ingresar el tipo de dato pedido, si no el programa parara de funcionar.

```
Que desea hacer?

1)Añadir Chatbot
2)Añadir Flow
3)Añadir Option
4)Visualizar Chatbots
5)Visualizar Chatbots con Flows y Options
6)Log-out
```

Imagen.3 Ejemplo menú Admin.

Además, se pueden ver los Chatbots con sus respectivos Flows y Options

En el caso de se un usuario normal, se puede interactuar con el Chatbot, seleccionando la opción que se desea utilizar o ingresando un keyword de este

```
A donde quieres ir?
1) 1) Mall
2) 2) Restaurante
1
```

Imagen4.Ejemplo de interacción.

También se puede ver el historial del usuario activo utilizando su respectiva opción

En el caso de cerrar sesión, se volverá al menú de adición de usuarios, donde se puede salir del programa ingresando 4.

8. Resultados y Autoevaluación

Como resultado, no se alcanzaron a implementar la totalidad de métodos funcionales requeridos, pero los que sí fueron implementados funcionan correctamente, pero sin la implementación de respuesta a errores, permitiendo que en ciertos casos el programa pare de funcionar.

La funcionalidad no implementada fue systemsimulate.

Como autoevaluación, a pesar de lograr implementar la mayoría de los requisitos, no se dividió correctamente el tiempo, evitando un desarrollo completamente óptimo de este laboratorio

9. Conclusiones

Durante el desarrollo de este laboratorio, la mayor limitante fue el tiempo, pero utilizar un Paradigma más parecido a lo que usualmente se utiliza al principio de esta carrera facilito mujo la ejecución de este laboratorio.

En comparación al laboratorio anterior, se logró un desarrollo óptimo de este, logrando un mayor nivel de logro.

Además, gracias a este laboratorio se logró entender la gran versatilidad de este paradigma, gracias a las posibilidades de reutilización de código y su fácil entendimiento, demostrando el porqué es tan popular

10. Bibliografia

Java. (s.f.). *Classes*. Obtenido de Oracle: https://docs.oracle.com/javase/tutorial/java/javaOO/classes.html

Java. (s.f.). *Inheritance*. Obtenido de Oracle: https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

Java. (s.f.). *Polymorphism*. Obtenido de Oracle: https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html

Panigrahi, K. K. (06 de 09 de 2023). *Difference between Bottom-Up Model and Top-Down Model*. Obtenido de Tutorialspoint: https://www.tutorialspoint.com/difference-between-bottom-up-model-and-top-down-model

11. Anexos:

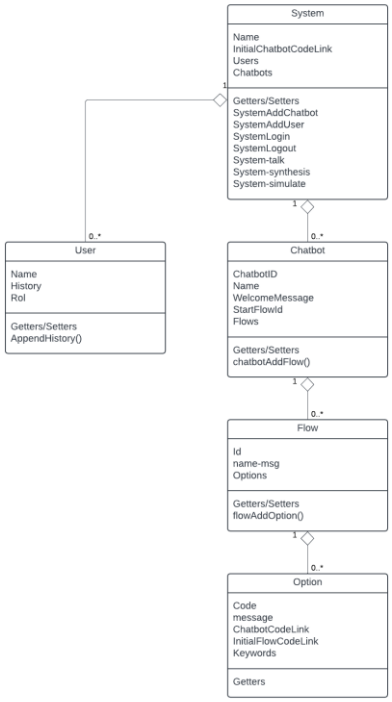

Figura 1. Diagrama de análisis

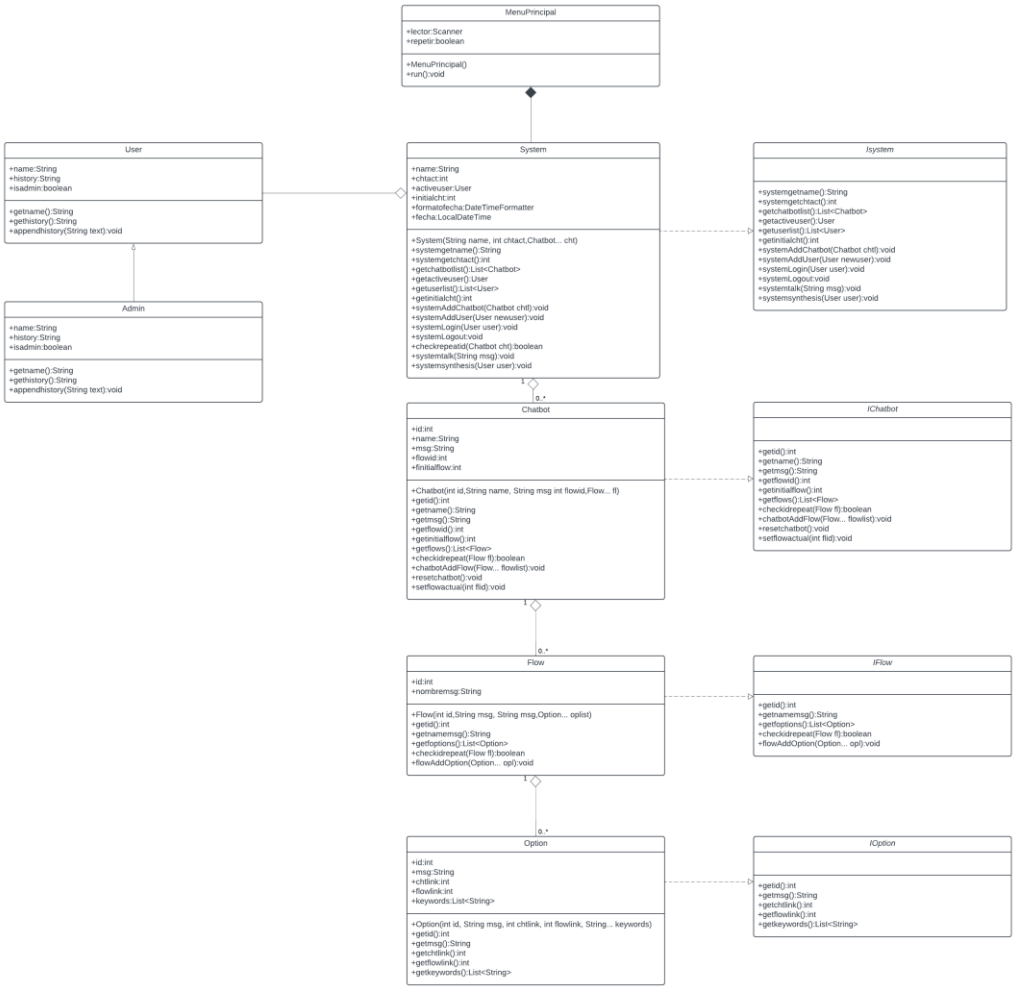

Figura.2 Diagrama de Diseño