DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

UNIVERSIDAD DE SANTIAGO DE CHILE FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

INFORME DE LABORATORIO 3:

Sistema de Chatbots con programación orientada a objetos

Nombre: Alvaro Muñoz Araya

Profesor: Víctor Flores

Asignatura: Paradigmas de Programación

Fecha: 11 de Diciembre de 2023

Contenido

1. INTRODUCCIÓN

En este informe se abordará el laboratorio 3 del curso de Paradigmas de Programación. En este laboratorio trabajaremos con el Paradigma Orientado a Objetos usando el lenguaje de programación Java para crear un sistema contenedor de Chatbots. Mas adelante podrá encontrar respuesta a las problemáticas abordadas y una breve explicación de el proceso realizado para este laboratorio.

1.1 DESCRIPCIÓN DEL PROBLEMA

Como problemática se pide realizar un programa a través del lenguaje de programación Java, que permita la creación/simulación/conversación/edición de Chatbots en base a interacción de texto. Esto quiere decir que se busca que el programa realizado tenga la capacidad de crear un Chatbot dentro de un sistema, con el cual se pueda conversar de manera simple a partir de opciones predefinidas por el usuario o el propio programa.

Es necesario mencionar que, al programar en el paradigma orientado a objetos, debemos hacer uso de clases para representar cada uno de los elementos mencionados.

1.2 DESCRIPCIÓN DEL PARADIGMA

El paradigma orientado a objetos es un enfoque de programación que se basa en el concepto de "objetos", los cuales combinan datos y comportamiento (métodos) en una única unidad. En base a esto surgen las siguientes definiciones:

Abstracción: Permite identificar las características esenciales de un objeto y ocultar los detalles no relevantes para centrarse en lo que realmente importa.

Encapsulamiento: Consiste en el empaquetamiento de datos (atributos) y funciones (métodos) que operan en esos datos en un solo componente, protegiendo la integridad de los datos al ocultar su estado interno y permitiendo el acceso controlado mediante interfaces.

Herencia: Permite crear nuevas clases basadas en clases previamente definidas, heredando sus atributos y métodos. Esto promueve la reutilización de código y la creación de jerarquías de clases.

Polimorfismo: Permite que objetos de distintas clases sean tratados de manera uniforme si pertenecen a la misma jerarquía, lo que significa que un objeto puede presentar diferentes formas o comportamientos según el contexto en el que se utilice.

Clases: Las clases definen las propiedades y comportamientos de los objetos. Son estructuras que encapsulan datos y funciones (métodos) relacionados. Por ejemplo, una clase "Coche" podría definir qué características y acciones tienen todos los coches, como su color, modelo y la capacidad de acelerar o frenar.

Atributos: Los atributos son las características o datos asociados a un objeto. En el ejemplo del coche, los atributos podrían ser el color, el modelo, el año, etc. Son variables que pertenecen a una instancia específica de una clase.

Métodos: Los métodos son funciones que están asociadas a una clase y describen el comportamiento de los objetos de esa clase. Son acciones que un objeto puede realizar. En el ejemplo del coche, podrían ser métodos como "acelerar", "frenar" o "cambiar marcha".

Objetos: Los objetos son instancias específicas de una clase. Se crean a partir de una clase y representan entidades individuales que poseen sus propios valores para los atributos definidos en la clase. Si tienes la clase "Coche", un objeto sería un coche en particular con un color, modelo y año específicos.

1.3 OBJETIVOS

El principal objetivo de este laboratorio es aprender sobre la programación orientada a objetos, conocer sus características e implementar una solución a un problema especifico utilizando Java como un lenguaje enfocado en este tipo de programación.

2. DESARROLLO

2.1 ANÁLISIS DEL PROBLEMA

El sistema de Chatbots no trabaja en base a la inteligencia artificial, si no mas bien corresponde a una implementación de interacción en base a texto, por lo que para poder conversar con un chatbot, se debe programar la lógica de conversación en base a palabras y frases predefinidas dentro del programa o por el propio usuario.

Por otro lado, para poder entender mejore el problema, debemos explicar algunos conceptos importantes a tener en cuenta, los cuales han sido utilizados durante todo el trabajo y como parte inherente del propio paradigma utilizado:

Chatbot: Un chatbot es una entidad capaz de contener Flows, cada Chatbot tiene su nombre, id, un mensaje de bienvenida, y un Flow de inicio. El chatbot es el encargado de almacenar toda esta información que será útil para conversar con el mismo.

System: Un sistema puede tener muchos Chatbots, también contiene un id, usuarios registrados y un solo usuario logeado. El usuario logeado es capaz de modificar el sistema agregando y/o eliminando Chatbots.

Flow: El Flow es el encargado de almacenar las opciones, este es necesario para navegar entre las distintas instancias de conversación que ofrece un chatbot, es decir permite mostrar un cumulo de opciones asociadas al mismo Flow. Cada Flow contiene un id, un mensaje y una lista de opciones.

Option: Una opción es una cadena de caracteres que se asocian a un Flow determinado, así como también a un chatbot. Cada opción tiene su propia id, mensaje, un código que asocia al Flow y al chatbot y además una lista de palabras claves que se usan para determinar una similitud con el mensaje ingresado por el usuario.

Usuario: El usuario representa al ente que conversa con un chatbot dentro del sistema, cada usuario tiene un nombre y una contraseña, así como un historial y un parámetro que determina si un usuario es administrador del sistema o no.

2.2 DISEÑO DE LA SOLUCIÓN

2.2.1.1 CLASE Identificador

- Representación: Int id, String message.
  Esta es una clase abstracta de la cual heredan las clases de Option, Flow y Chatbot

2.2.1.2 CLASE Option

- Representación: Int id es el identificador de una opción, String message es el contenido de la opción, int chatbotCodeLink que representa el id al cual está asociada la opción, int initialFlowCodeLink que representa el Flow al que está asociada la opción, List keywords es una lista de palabras claves que se asocian a la opción.

2.2.1.3 CLASE Flow

- Representación: Int id es el identificador de un Flow, String message es el contenido del Flow, List options es un listado de opciones enlazadas a un Flow

2.2.1.4 CLASE Chatbot

- Representación: Int id es el identificador de un Chatbot, String message es el mensaje de bienvenida, String name corresponde al nombre del Chatbot, int StartFlowId es el Flow inicial que esta enlazado al Chatbot, List flow es una lista de Flows que contiene el Chatbot

2.2.1.5 CLASE User

- Representación: Bolean admin representa si el usuario es admin o no, String username es el nombre de usuario, String password es la contraseña, ChatHistory historial es el historial del usuario

2.2.1.6 CLASE ChatHistory

- Representación: User usuario es el usuario asociado al historial, List messages es una lista de mensajes con diversa información.

2.2.1.7 CLASE Message

- Representación: String mensaje representa el mensaje ingresado por el usuario, Lista fecha representa la fecha y hora actual en la que se ingresó el mensaje, int chatbotID representa el id del Chatbot asociado al mensaje enviado, int flowID representa el id del Flow asociado al mensaje enviado, int opcionID representa la id de la opción escogida de la lista de opciones.

2.2.1.8 CLASE System

- Representación: String name representa el nombre del sistema, int initialChatbotCodeLink representa el id inicial del Chatbot del sistema, List chatbots es una lista de Chatbots que están en el sistema, List registeredUsers es la lista de los usuarios registrados en el sistema, List activeUser es una lista que contiene 1 o 0 usuarios y representa el usuario que esta logeado en el sistema.

2.2.2 FUNCIONALIDADES OBLIGATORIAS

TDA Option – constructor: Es el método constructor de la clase Option, crea una opción nueva a partir de los parametros int id X String message X int chatbotCodeLink X ArrayList<String> keywords X int initialFlowCodeLink

TDA Flow – constructor: Es el método constructor de la clase Flow, crea un Flow nuevo a partir de los parámetros int id X String message X ArrayList<Option > options, también existe el constructor vacio, que crea un Flow con atributos por defecto (No recibe ningún parametro)

flowAddOption: Método que recibe como parámetro una Opción, la cual añade al flujo verificando si su id no se encuentra dentro de la lista de Opciones.

TDA chatbot – constructor: Es el método constructor de la clase Chatbot, crea un Chatbot nuevo a partir de los parámetros int chatbotID X String name X String welcomeMessage X int startFlowId X ArrayList<Flow > Flows, también existe el constructor vacio, que crea un Chatbot con atributos por defecto (No recibe ningún parametro)

chatbotAddFlow: Método que recibe como parámetro un Flow, el cual añade a la lista de Flows que contiene el Chatbot, verifica si la id del Flow se encuentra en la lista.

TDA Usuario – constructor: Es el método constructor de la clase User, crea un User nuevo a partir de los parámetros String username X String userPassword X boolean admin, existe también el constructor vacío.

TDA system – constructor: Es el método constructor de la clase System, crea un System nuevo a partir de los parámetros String name X int initialChatbotCodeLink X ArrayList<Chatbot > chatbots X ArrayList<User > activeUser

systemAddChatbot: Método que añade un Chatbot a la lista de Chatbots del sistema, verificando que el id del Chatbot no se repita, recibe el parámetro Chatbot.

systemAddUser: Método que añade un User a la lista de usuarios del sistema, verificando que el nombre del usuario no se encuentre en la lista de usuarios registrados, recibe el paramtro User.

systemLogin: Método que permite añadir un User a la lista de usuarios logeados, en esta lista se verifica que no exista ningún usuario para poder ingresar uno. Recibe como parámetro User.

systemLogout: Método que permite deslogear al usuario del sistema, simplemente quita al usuario que se encuentra dentro de la lista de usuarios logeados. No recibe ningún parámetro.

system-talk: Método que permite interactuar con un Chatbot, busca el ultimo mensaje ingresado por el usuario en la lista de mensajes guardadas en el historial del usuario logeado, de este ultimo mensaje consigue el id del Flow, Chatbot y Option, seguido a ello, a partir del mensaje ingresado por el usuario, guarda el siguiente mensaje comparándolo con las opciones del Flow. Recibe como parámetro String

system-synthesis: Método que formatea los mensajes del historial del usuario para poder imprimirlos por consola. Recibe como parámetro User.

2.3 ASPECTOS DE IMPLEMENTACIÓN

2.3.1 COMPILADOR / IDE

El proyecto está hecho en Java y se necesita un IDE como, por ejemplo, Netbeans, inteliJ o Eclipse. Durante el desarrollo de este trabajo se utilizó específicamente el IDE Eclipse en su versión 4.29.0. La versión de Java utilizada fue la 11.0.21. No se utilizaron librerías externas, solo librerías estándar de Java, además el Código ha sido compilado y ejecutado en la versión de Windows 10.

2.3.2 ESTRUCTURA DEL CÓDIGO

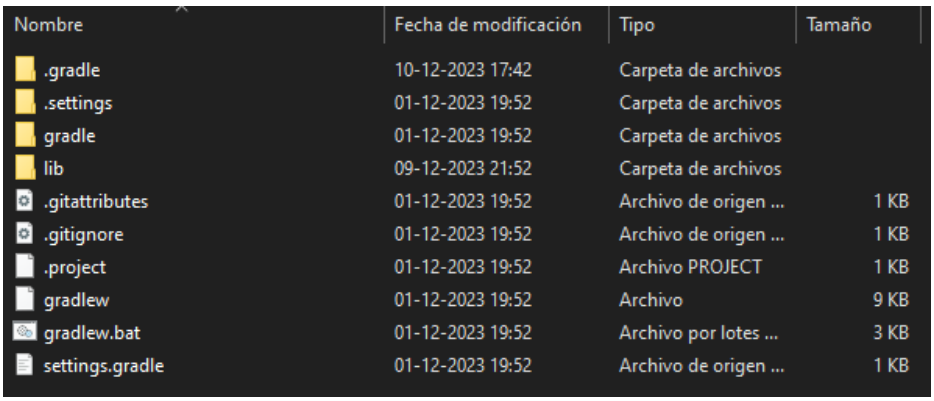
La ruta especifica para encontrar todos los archivos **.java** pertenecientes a cada uno de los TDA's requeridos es:
lab3_207242381_MunozAraya\lab3Paradigmas\lib\src\main\java\lab3Paradigmas

La ruta para encontrar la pagina **index.html** asociada a la documentación en JavaDoc es:
lab3_207242381_MunozAraya\lab3Paradigmas\doc

2.4 INSTRUCCIONES DE USO

Para poder ejecutar el Código debemos asegurarnos de tener instalado JDK 11 y gradle (teniendo incluido y activado el Gradle Wrapper)

Es importante mencionar que para ejecutar el programa correctamente debemos asegurarnos de abrir la consola en la ubicación correcta, para este caso debemos dirigirnos a la ubicación **\lab3Paradigmas** la cual contiene los siguientes archivos:



Nombre	Fecha de modificación	Tipo	Tamaño
.gradle	10-12-2023 17:42	Carpeta de archivos	
.settings	01-12-2023 19:52	Carpeta de archivos	
gradle	01-12-2023 19:52	Carpeta de archivos	
lib	09-12-2023 21:52	Carpeta de archivos	
.gitattributes	01-12-2023 19:52	Archivo de origen ...	1 KB
.gitignore	01-12-2023 19:52	Archivo de origen ...	1 KB
.project	01-12-2023 19:52	Archivo PROJECT	1 KB
gradlew	01-12-2023 19:52	Archivo	9 KB
gradlew.bat	01-12-2023 19:52	Archivo por lotes ...	3 KB
settings.gradle	01-12-2023 19:52	Archivo de origen ...	1 KB

figura 1: Carpeta \lab3Paradigmas

Desde esta ubicación **en la consola** se ejecuta el comando:

```
gradlew.bat run
```

Gradle al momento de ejecutarse muestra una barra de progreso que puede causar incomodidad a la hora de visualizar el programa, es por ello que es altamente recomendable ejecutar el programa usando el comando:

```
gradlew run --console=plain
```

Es importante seguir los pasos correctamente a la hora de entrar en el menú del programa (Solo al momento de ejecutar un Chatbot se utilizan palabras claves y/o números para hacer la interacción más simple, el menú propio del programa requiere ingresar las entradas de manera correcta).

Otro punto importante de mencionar es que ya existe un Chatbot creado en el sistema, el cual tiene Flows asociados con sus respectivas opciones. Este Chatbot tiene id = 0 y el usuario puede interactuar con él accediendo al menú y seleccionando la opción de ejecutar un Chatbot, luego de ello simplemente se debe ingresar el número 0 correspondiente al id del Chatbot para empezar su ejecución.

Finalmente se señala la existencia de un usuario administrador, el cual tiene como nombre de usuario: admin y contraseña: admin

Este usuario como cualquier otro usuario creado con la capacidad de administración, tiene la capacidad de añadir Chatbots con sus respectivos Flows y Options al sistema ya creado. Es importante que, al momento de ingresar el usuario y la contraseña, esto se haga de manera correcta y siguiendo las instrucciones que se indican en la consola.

2.4.1 RESULTADOS ESPERADOS

El resultado esperado es que el programa sea capaz de crear un ambiente contenedor de Chatbots, dándole al usuario la libertad de poder ejecutar un Chatbot y mantener una conversación, por otra parte, también se espera que el usuario administrador sea capaz de modificar el sistema, agregando o eliminando Chatbots, Flujos y/o opciones.

2.4.2 POSIBLES ERRORES

Como posibles errores se encuentran los casos donde el usuario no ingrese adecuadamente los parámetros requeridos o donde algún Chatbot, Flow u Option quede de forma Nulo dentro del sistema.

2.5 RESULTADOS Y AUTOEVALUACIÓN

Los resultados obtenidos no son los esperados, ya que falta realizar la implementación del método systemSimulate, que permite simular la conversación de un usuario con un Chatbot. Sin embargo, el programa es funcional a la hora de que un usuario pueda interactuar con el sistema y crear Chatbots/flujos/opciones.

Se sugiere revisar los diagramas de clases creados antes y después de realizar el trabajo. *(Revisar bibliografía figura 2 y 3).*

La autoevaluación realizada es:

Requerimientos Funcionales	Evaluación
TDAs	1
option	1
flow	1
flow-add-option	1

chatbot	1
chatbot-add-flow	1
system	1
system-add-chatbot	1
system-add-user	1
system-login	1
system-logout	1
system-talk	1
system-synthesis	1
system-simulate	0

3. CONCLUSIÓN

Para finalizar, se dice que a grandes se tuvo éxito en la elaboración del trabajo, donde se implementaron los conceptos asociados al paradigma de la programación orientada a objetos, abordando el problema inicial de la manera requerida y utilizando el lenguaje de programación Java como fue solicitado.

En relación con lo anterior no se descartan problemas de ejecución además de los ya mencionados y se menciona nuevamente la no implementación del método SystemSimulate el cual por falta de tiempo no fue agregado al programa.

Por último, también se tuvo éxito a la hora de entender y manejar Java gracias a los conceptos vistos en catedra y clases de ejercicios, por lo que la escritura del código fue más sencilla.

4. BIBLIOGRAFIA Y REFERENCIAS

Paradigma Orientado a Objetos

https://uvirtual.usach.cl/moodle/course/view.php?id=10036&section=20

Programación Orientada a Objetos en Videojuegos

https://gameprogrammingpatterns.com/contents.html

Tutorial de uso de Gradle

https://gameprogrammingpatterns.com/contents.html

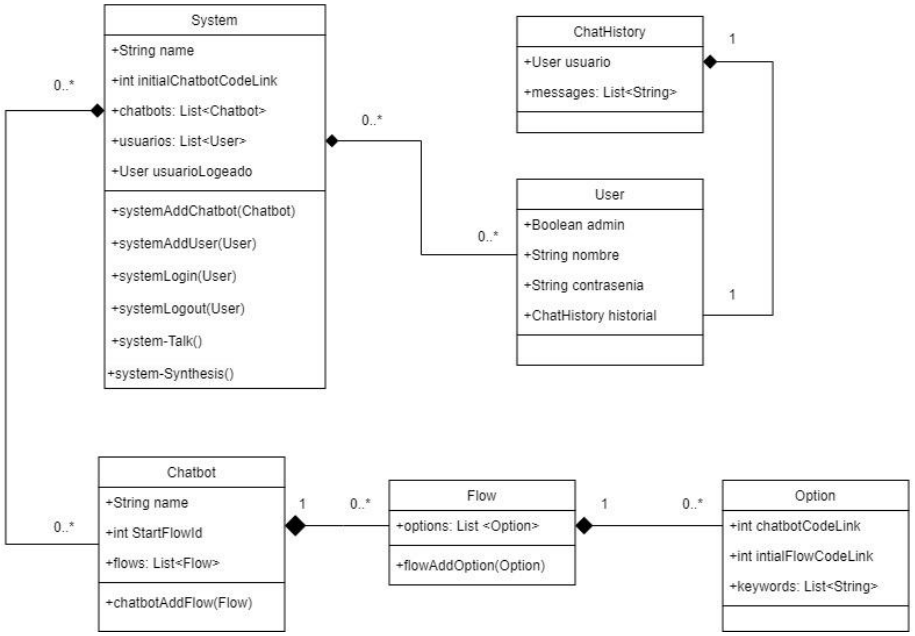


Figura 2: Diagrama de clases **antes** de realizar el trabajo

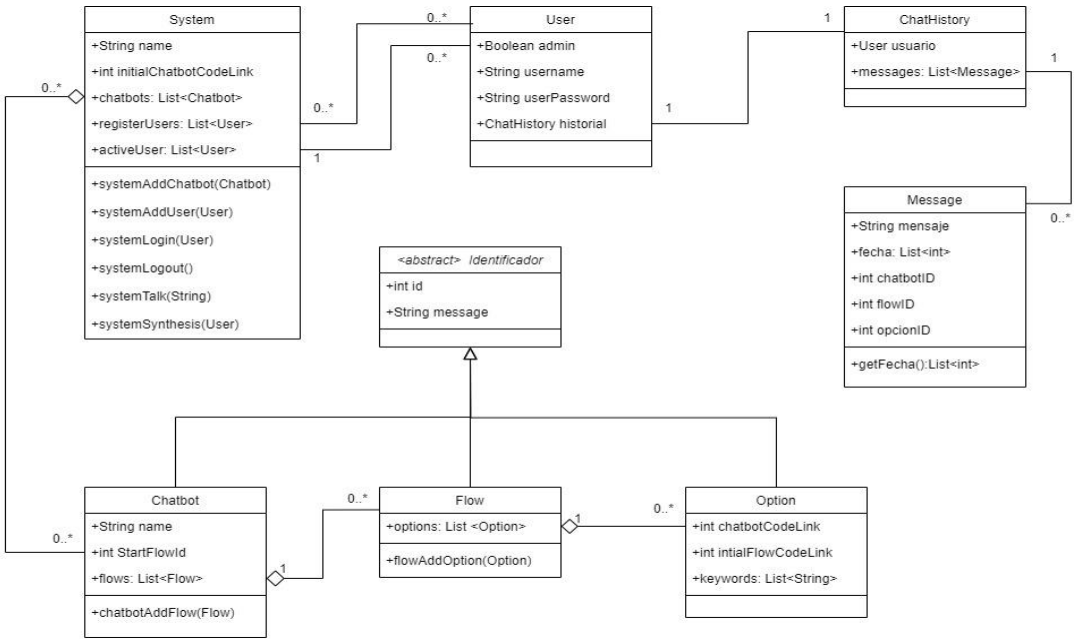


Figura 3: Diagrama de clases despues de realizar el trabajo