

**Paradigmas de la Programación**

Informe Laboratorio nro. 3

Paradigma Orientado a Objeto

Profesor: Gonzalo Martínez.

Alumna: Karina Bustamante H.

16 de diciembre de 2023

Contenido

[Introducción 2](#_Toc152970365)

[Descripción del paradigma 3](#_Toc152970366)

[Descripción del problema 4](#_Toc152970367)

[Análisis del problema 5](#_Toc152970368)

[Diseño de la solución 5](#_Toc152970369)

[Consideraciones de implementación 5](#_Toc152970370)

[Diagramas UML previa implementación 6](#_Toc152970371)

[Diagramas UML posterior a implementación 6](#_Toc152970372)

[Instrucciones de uso 6](#_Toc152970373)

[Resultados y evaluación 7](#_Toc152970374)

[Conclusiones 7](#_Toc152970375)

[Referencias 7](#_Toc152970376)

# Introducción

En el presente trabajo, abordaremos el desarrollo del laboratorio Nº3 de la asignatura de Paradigmas de la Programación, donde el objetivo es implementar algoritmos bajo el paradigma orientado a objetos para resolver el siguiente problema planteado:

" **Sistema para la creación, despliegue y administración de chatbots simplificado.**"

Para llevar a cabo esta implementación, utilizaremos el lenguaje de programación Java, con el cual, buscaremos brindar una solución integral a los requerimientos funcionales establecidos, de modo que la solución propuesta permita al usuario realizar diversas acciones sobre este sistema de administración de chatbot. Estas acciones incluyen la creación e identificación de un chatbot, la adición de preguntas y opciones, entre otras funcionalidades.

El paradigma orientado a objetos se presenta como el enfoque idóneo para abordar este tipo de problemas. Nos brinda la capacidad de modelar y organizar el sistema de administración de chatbot y sus interacciones mediante el uso de objetos y clases. Aprovecharemos conceptos claves como la encapsulación, la herencia y el polimorfismo para construir una solución modular, eficiente y escalable.

En las secciones siguientes, describiremos en detalle el enunciado del problema, la metodología utilizada, la estructura del programa implementado y los resultados obtenidos. También analizaremos posibles mejoras y compartiremos las lecciones aprendidas durante la realización de este laboratorio.

# Descripción del paradigma

El paradigma orientado a objetos (POO) es un enfoque de programación que se basa en la organización y manipulación de objetos como elementos centrales. Un objeto es una entidad que combina datos (propiedades o atributos) y comportamiento (métodos o funciones) relacionados.

En el paradigma orientado a objetos, los objetos son instancias de clases, que actúan como plantillas o moldes para la creación de objetos. Una clase define las propiedades y los métodos que los objetos de esa clase tendrán. Las propiedades representan el estado o los datos que posee un objeto, mientras que los métodos encapsulan el comportamiento o las acciones que el objeto puede realizar.

La programación orientada a objetos se basa en cuatro conceptos fundamentales:

***Abstracción***: Permite representar las características esenciales de un objeto del mundo real en un modelo simplificado dentro del programa. La abstracción se logra mediante la identificación de las propiedades y el comportamiento relevantes para el problema en cuestión.

***Encapsulación***: Consiste en agrupar datos y métodos relacionados en un objeto y ocultar los detalles internos del objeto. Esto se logra definiendo la visibilidad de los miembros de la clase (públicos, privados, protegidos) para controlar el acceso desde el exterior.

***Herencia***: Permite la creación de nuevas clases (subclases) basadas en clases existentes (superclases). La herencia permite la reutilización de código y establece una relación "es-un" entre las clases, donde una subclase hereda las propiedades y métodos de su superclase.

***Polimorfismo***: Permite que objetos de diferentes clases respondan de manera diferente a un mismo mensaje o método. El polimorfismo permite tratar a objetos de distintas clases de manera uniforme, lo que facilita la flexibilidad y la extensibilidad del código.

El paradigma orientado a objetos se utiliza ampliamente en el desarrollo de software debido a su capacidad para modelar de manera eficiente problemas complejos, promover la reutilización de código y mejorar la modularidad y mantenibilidad del sistema.

# Descripción del problema

En el contexto de esta simulación, nos enfrentamos al desafío de desarrollar chatbots utilizando el lenguaje de programación Java, con un enfoque específico en la categoría ITR (Interacción a Través del Lenguaje Natural). Esta categoría implica que los chatbots estarán diseñados según los principios fundamentales del paradigma orientado a objetos.

En este sentido, se aprovecharán las características claves del paradigma orientado a objetos, como la encapsulación, la herencia y el polimorfismo, para modelar de manera eficaz la estructura y comportamiento de los chatbots. La encapsulación permitirá la organización interna de los componentes del chatbot, la herencia facilitará la reutilización de código al definir relaciones jerárquicas entre clases, y el polimorfismo posibilitará adaptar las respuestas según el contexto de interacción.

Dentro de la categoría ITR, las opciones de interacción se diseñarán de manera que el usuario pueda seleccionar preguntas de una lista predefinida. Además, se considerará la posibilidad de incorporar palabras claves y alternativas simplificadas mediante números o letras, adaptando la flexibilidad del paradigma a la variedad de formas de interacción del usuario.

Por lo anterior, el desarrollo de los chatbots no solo se centrará en la funcionalidad específica de la ITR, sino que también se beneficiará de las ventajas que brinda el paradigma orientado a objetos para lograr una solución modular, flexible y eficiente.

# Análisis del problema y Diseño de la solución

En este apartado buscamos analizar algunos de los problemas de la implementación de un sistema simplificado de chatbot. Comenzaremos con el “*inicio de sesión*”, el sistema debe identificar el tipo de usuario que operará el sistema de chatbot, para ello, crearemos un algoritmo que permita identificar al tipo de usuario conectado mediante un campo booleano isAdmin (true / false), con esta información lograremos desplegar el menú correspondiente al tipo de usuario.

Para la creación de un chatbot, se propone la implementación de un algoritmo que mediante las opciones de un menú y posteriores solicitudes de entradas de información por parte del usuario se obtengan los datos necesarios para crear las Opciones, Flujos y finalmente el Chatbot.

También, es requerido verificar la unicidad de opciones, flujos y chatbots que se agreguen al sistema, para ello implementaremos un ciclo que chequee cada lista antes de agregarlas a sus contenedores. Es decir, validará la unicidad de las opciones al momento de agregarlas a un flujo, la unicidad de los flujos al momento de agregarlos a un chatbot y la unicidad de los chatbots al agregarlos al sistema.

La modificación de un chatbot, deberá verificar antes su existencia, para ello se pretende realizar un algoritmo que despliegue los chatbots existentes en el sistema, de esta forma, el usuario pueda identificar el id del chatbot que desea modificar, lo seleccione y realice los cambios.

Para la visualización de los objetos creados, se implementará un algoritmo que permita imprimir por consola la información obtenida de la acción de recorrer el sistema en busca de los objetos consultados.

# Aspectos de la implementación

Para la implementación del sistema simplificado de chatbot, diseñado bajo el paradigma orientado a objeto y codificado en el lenguaje de programación Java, destacaremos a continuación, algunos aspectos importantes.

***Diseño de clases:***

* *Chatbot*, representa al chatbot en sí.
* *Interfaz de usuario*, maneja la interacción con el usuario, mostrando mensajes y recibiendo la entrada.
* *Base de datos*, variables globales que almacenaran la información en forma de listas.

***Implementación de clases***:

* *Chatbot*, métodos para iniciar la conversación, procesar la entrada del usuario y generar una respuesta mediante la implementación de una lógica especifica.
* *Interfaz de usuario*, métodos para despliegue de los menús al usuario y captura de las respuestas.
* *Base de datos*, almacena información que el chatbot requiere para generar respuestas.

***Manejo de Datos***:

* Entrada/salida:

- Se utiliza la clase “Scanner” para obtener la entrada del usuario desde la consola.

- Se muestran las respuestas del chatbot en la consola.

# Diagramas UML previa implementación

Diagrama

Descripción generada automáticamente

# Diagramas UML posterior a implementación

# Instrucciones de uso

Para operar esta simulación de sistema simplificado de chatbot es necesario tener un usuario registrado, es decir, deberá iniciar sesión en el sistema con su usuario, si no existe como usuario deberá antes crear el usuario mediante la opción “Registrar Usuario” dependiendo del tipo de usuario con que se conectará al sistema. Posterior a ello, deberá volver al menú inicio e iniciar sesión con su usuario recién creado, se asignarán los menús correspondientes al tipo de usuario y podrá interactuar con el sistema simplificado de chatbots a través de diferentes menús, en donde deberá seleccionar la opción deseada mediante el numero de la opción para avanzar en su recorrido.

# Resultados y evaluación

A continuación, se muestra la autoevaluación de los requerimientos funcionales requeridos por el laboratorio, y el grado de logro alcanzado para cada uno de ellos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RF Nº | Requerimiento funcional | Grado de alcance | Tipo o cantidad de pruebas | Éxitos | Fracasos | Razones de fallo |
| 1 | RF1- Clases y estructuras | 1 |  |  |  |  |
| 2 | RF2 – Menú Interactivo por consola | 1 | Impresión por consola |  |  |  |
| 3 | TDA Option-Constructor | 1 | 2 | 2 |  |  |
| 4 | TDA Flow-Constructor | 1 | 2 | 2 |  |  |
| 5 | TDA Flow-flowAddOption | 0,75 - 0 | 3 | 2 | 1 | Falta implementar chequeo unicidad |
| 6 | TDA Chatbot-Constructor | 1 | 3 | 1 | 2 | Faltaba agregar toString |
| 7 | TDA Chatbot-chatbotAddFlow | 0,75 0 | 3 | 2 | 1 | Falta implementar chequeo unicidad |
| 8 | TDA Usuario - Constructor | 1 | 4 | 2 | 2 | Unicidad y tipo de usuario |
| 9 | TDA System - Constructor | 1 | 1 | 1 |  |  |
| 10 | TDA System - systemAddChatbot | 0 |  |  |  |  |
| 11 | TDA System - systemAddUser | 1 | Impresión por consola | 2 |  |  |
| 12 | TDA System - systemLogin | 1 | Impresión por consola | 4 | 2 | Tipo de usuario para mostrar menú correspondiente |
| 13 | TDA System - systemLogout | 0.75 - 1 | Impresión por consola | 2 |  |  |
| 14 | systemTalk | 0 |  |  |  |  |
| 15 | system-synthesis | 0 |  |  |  |  |
| 16 | system-simulate | 0 |  |  |  |  |

Tabla 1.- Muestra los requerimientos funcionales y el nivel de logro.

# Conclusiones

Es desarrollo de este proyecto nos permitió abordar la creación de un sistema simplificado de chatbot, implementado en lenguaje de programación Java y bajo la mirada del Paradigma Orientado a Objeto, con el cual se pudo generar respuestas estructuradas a través de acciones específicas. El chatbot, categorizado como ITR (Iteración a Través del Lenguaje Natural), permitirá a los usuarios interactuar de manera eficiente utilizando palabras clave, números o letras. El análisis del problema ha proporcionado una visión clara de los objetivos, requisitos, alcance y limitaciones del proyecto, guiando así la estrategia de implementación para lograr un sistema de chatbot adaptado a las necesidades identificadas. En cuanto a la experiencia obtenida de desarrollar este proyecto en los tres paradigmas de programación que aprendimos, el POO nos permitió modelar entidades a través de objetos, facilitó la reutilización del código mediante la encapsulación y la herencia. La estructura creada para las entidades de “Usuario”, “Chatbot”, “Interfaz de Usuario” facilito la comprensión del sistema. En el paradigma funcional, que fomenta el uso de estructuras inmutables y facilita la gestión de estados permitiendo pasar funciones como argumentos y devolverlas como resultados, me fue más complejo incorporar este conocimiento para el desarrollo e implementación del proyecto. Y desde la mirada del paradigma Lógico que se centra en lo que quiere lograr en lugar del como hacerlo, mediante reglas y relaciones lógicas, fue menos complejo que el paradigma funcional, implementar el proyecto de chatbot simplificado.

# Referencias

* Campus Virtual, (2023). Paradigmas de Programación. <https://uvirtual.usach.cl/moodle/>
* Clases Grabadas, (2023). Paradigmas de Programación.

<https://drive.google.com/drive/folders/1EcdB9yz_FtxTJInpoJEUXiEJcGvmBKeC?usp=sharing>