

Paradigmas de la Programación

Informe Laboratorio nro. 3

Paradigma Orientado a Objeto

*Profesor: Gonzalo Martínez.*

*Alumna: Karina Bustamante H.*

12 de diciembre de 2023

# 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:

1. **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.
2. **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.
3. **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.
4. **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, aprovechando la flexibilidad y modularidad inherentes al paradigma orientado a objetos. Además, se considerará la posibilidad de incorporar sinónimos y alternativas simplificadas mediante números o letras, adaptando la flexibilidad del paradigma a la variedad de formas de interacción del usuario.

Por tanto, 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

acá deberías analizar cada requerimiento y pensar una posible solución

# Diseño de la solución

falta profundizar en el diseño de la solución cómo solucionaste el login, cuál fue tu representación del tda, sus capas, cómo abordaste el tda con estas capas, cada requerimiento como lo diseñaste (no todos, pero los más significativos)

# Consideraciones de implementación

**Diagramas UML previa implementación**

a. System: Representa el sistema con sus atributos y comportamientos.

b. Chatbot: Representa el chatbot con sus atributos y comportamientos.

c. Flow: Representa el flujo con sus atributos y comportamientos.

d. Option: Representa cada option con sus atributos y comportamientos.

e. User: Representa a cada usuario y su comportamiento asociado.

# Diagramas UML posterior a implementación

# Instrucciones de uso

# 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 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |
| 13 |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |

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

# 

# Conclusiones

falto comparar la experiencia de este paradigma con el paradigma anterior (funcional) y otros experiencias previas que has tenido (por ejemplo, python, c, etc).

Próximo laboratorio: la conclusión de orientado a objeto debes compararlo con tu experiencia acá en prolog y luego en scheme

# Referencias