UNIVERSIDAD DE SANTIAGO DE CHILE FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INFORMÁTICA

**INFORME CHATBOT**

**Programación en Java**

|  |  |
| --- | --- |
| Nombre Alumno: | Francisco Guajardo |
| Profesores: | Daniel Gacitúa |
| Ayudantes: | Javier Vasquez |
| Fecha de Entrega: | 10 – 08 - 2018 |

Tabla de contenido

[Introducción 2](#_Toc521708051)

[Marco teórico 3](#_Toc521708052)

[Descripción problema 4](#_Toc521708053)

[Paradigma orientado a objetos 5](#_Toc521708054)

[Análisis del problema 6](#_Toc521708055)

[Diseño y aspectos de implementación de la solución 7](#_Toc521708056)

[Resultados 9](#_Toc521708057)

[Conclusiones 9](#_Toc521708058)

[Referencias 10](#_Toc521708059)

## **Introducción**

En la actualidad muchas empresas y compañías de bienes y servicios implementan en algún momento la entidad chatbot.

El chatbot es una entidad conversacional artificial. Son básicamente programas computacionales que son capaces de mantener o generar una conversación fluida con un usuario.

En este laboratorio se nos encomendó la tarea de realizar un chatbot que logre los estándares de este, basando nuestro algoritmo en el paradigma orientado a objetos usando el lenguaje Java.

Este chatbot esta diseñado para mantener una conversación corta con un usuario que desea comprar una película, juego o manga de la tienda X, indicando su precio y pidiendo forma de pago al usuario.

Este programa funciona en cualquier pc que posea instalado JDK 8 o superior, ya que asi esta planteado el manejo de este lenguaje.

## **Marco teórico**

* **Lenguaje de programación:** Un lenguaje de programación es un vocabulario, un conjunto de reglas gramaticales que tiene como objetivo dar instrucciones al computador para que este realice tareas especificas.
* **Java:** Es un lenguaje de programación y una plataforma informática comercializada por Oracle.
* **Paradigma de programación:** Es un estilo de programar, un camino a seguir, una filosofía para diseñar soluciones a problemas relacionados con la programación.
* **Recursión:** Es el proceso de definir una solución a un problema en términos de ella misma. Teniendo elementos mínimos como condición de parada y llamada recursiva. Existen diferentes tipos de recursión, como la recursión natural, recursión arbórea y recursión de cola.
* **Algoritmo:** Un algoritmo es un procedimiento esquemático que comprende un conjunto de pasos secuenciales ordenados, para realizar una actividad especifica
* **TDA:** Un tipo de dato abstracto consiste en la representación de un objeto, proceso o cualquier elemento real o imaginario para que sea operado por la maquina. Para formar un TDA tenemos que tener en consideración su arquitectura de seis elementos los cuales son Operadores, modificadores, selectores, funciones de pertenencia, constructores y destructores. Como mínimo de esa arquitectura se requiere una función constructora, función selectora y funciones para operar el TDA.
* **Clase:** Dentro de este paradigma, este elemento sirve para poder crear objetos, (tipo plantilla) con atributos y métodos que ayudan a poder desarrollar correctamente el programa.
* **Objeto:** es una unidad dentro de un programa de computadores que consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
* **Diagrama**: Representación gráfica de las variaciones de un fenómeno o de las relaciones que tienen los elementos o las partes de un conjunto.

## **Descripción problema**

El problema que nos fue planteado es el de crear un chatbot basado en el paradigma orientado a objetos utilizando Java.

Para este problema había que diseñar un contexto o temática general para el chatbot, el cual se usaría para guiar y definir las actitudes que este va a tomar con las respuestas que el usuario le entregue, además este contexto se deberá́ de mantener durante todos los laboratorios.

El chatbot debería ser capaz de poder responder al usuario de forma empática para lograr una mayor conexión con el, este a su vez deberá́ de contar con líneas de mensajes protocolares y de estilo libre.

Un ejemplo de conversación podría ser el siguiente:

• Chatbot: ¿Cual de nuestros servicios desea contratar?

• Usuario: cable

• Chatbot: Registrado, ¿Qué hay de internet?

• Usuario: no

• Chatbot: Entendido, ¿y telefonía?

• Usuario: si

• Chatbot: Perfecto, ya he registrado sus servicios

(ejemplo rescatado del enunciado del problema desde USACH Virtual)

Al ser un lenguaje orientado a objetos se nos solicita el manejo de las características del usuario, para su posterior análisis.

La forma de programar, en comparación con los laboratorios anteriores es la implementación de un nuevo lenguaje, Java y otro paradigma, que contrasta en muchas cosas con los anteriormente vistos.

## **Paradigma orientado a objetos**

El paradigma esta organizado alrededor de **objetos** en lugar de acciones y **datos** en lugar de lógica.

La programación orientada a objetos (POO) ofrece elementos que facilitan el desarrollo de soluciones en aspectos tales como modularización, reutilización de piezas de código (no copiar y pegar) y encapsulamiento (ocultar el estado de los objetos).

Existe un acuerdo acerca de qué características contempla la "orientación a objetos". A continuación, se dará una breve definición de algunas de las mas importantes.

**Abstracción**: denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real.

**Encapsulamiento**: significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema.

**Modularidad**: se denomina modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas, cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

**Polimorfismo**: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando.

**Herencia**: las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes.

## **Análisis del problema**

El desafío es implementar nuestro ya conocido chatbot en base a Java con las herramientas que nos entrega JDK 8 sin la adición de nuevas librerías.

Para el desarrollo de las entidades que serán involucradas en el programa se exige el manejo de estas mediante el uso de clases, para así tener estas entidades encapsuladas en objetos que serán mas fáciles de manejar durante el programa.

En si el problema es exactamente el mismo que los anteriores laboratorios, en este nos facilitan el uso de variables y clases, por lo que la dificultad no debería de ser mayor a los anteriores, pero, para lograr un uso optimo y adecuado de estos elementos se requiere poder comprender el paradigma en el cual se esta trabajando y sobretodo la programación orientada a objetos.

La seed le otorgara a nuestra entidad chatbot un estado de animo pseudo-aleatorio que cambiara los parámetros y la instanciación del objeto de nuestra IA, entregando un abanico de posibles respuestas distinto dependiendo de la seed ingresada por el usuario.

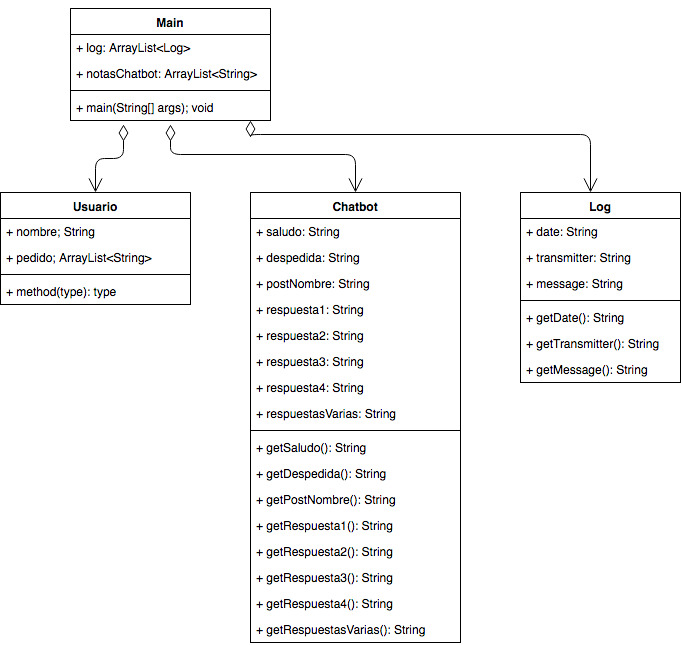
En el problema la conversación fluye por medio de la consola (o el medio por donde se este ejecutando el programa) con el usuario ingresando datos por pantalla. Para definir las funcionalidades del chatbot se crea un método por el cual se diferenciarán las funciones con el dialogo normal, esta característica es agregando un ”!” antes de la palabra (ej. !beginDialog). De las funciones que se tienen que implementar, un par de estas se implementan de forma similar, o utilizando casi las mismas librerías como son !saveLog y !loadLog, ambas requieren manejo de archivos, el resto solo interactúa con los objetos presentes dentro del paquete.

El desafío es crear la interacción bot-user sin el uso de estas funciones, buscando frases claves para el buen uso de las respuestas que el bot nos puede ofrecer.

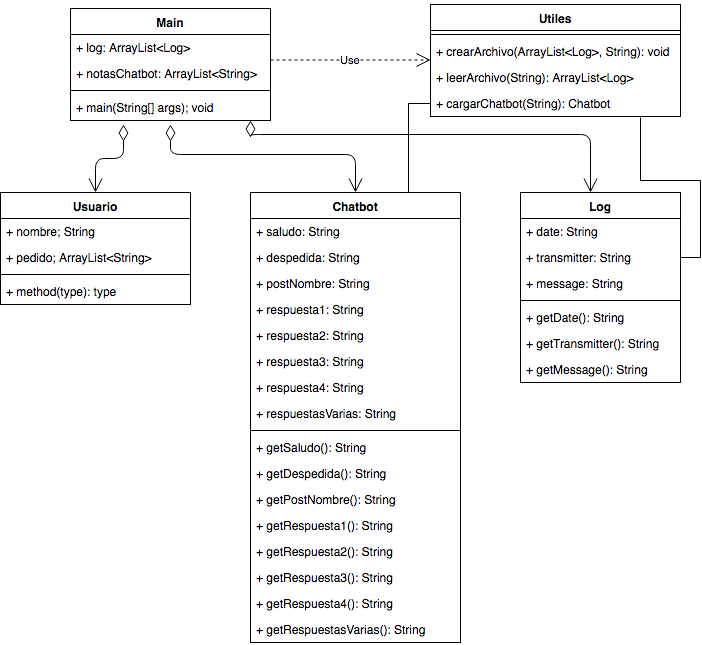
## **Diseño y aspectos de implementación de la solución**

Primero se entregarán los diagramas de clases, el inicial y el final, se notarán pequeños cambios en ambos:

**Primer diagrama:**



**Segundo Diagrama:**



Como se puede apreciar, el diseño consta de la clase principal Main, donde se verán todas las interacciones usuario-bot, donde se almacena el Log de la conversación que corresponderá a un arraylist de objetos Log que incluirán lo necesario para su análisis.

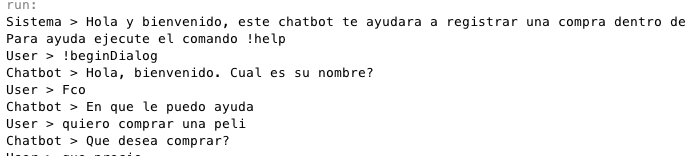
La clase Utiles tendrá métodos Static que usará Main para el manejo de archivos, será como una librería de funciones.

Tanto la clase Chatbot como Log posee los métodos para obtener sus datos, los get(), asi el manejo de objetos se facilita un montón y le quita peso a la clase Main.

Se usaron las librerías incluidas en JDK 8 sin la adición de extras.

## **Resultados**

Los resultados, de la misma forma que en los laboratorios anteriores, no son los deseados puesto que se logra una conversación muy limitada entre bot y usuario, limitándose a un par de preguntas y respuestas.



La conversación lo logra ser natural pero las funciones funcionan bien, limitando lo que deben de limitar y logrando lo esperado.

De los casos prueba que se hicieron, casi el 50% terminaron en error, por la mala escritura del archivo para cargar o porque el chatbot no terminaba ideas y se perdía el hilo de la conversación, con un par de ejemplos se logro lo deseado.

La no obtención de resultados esperados solo se justifica con la poca investigación realizada por parte del alumno, estando los contenidos no se logro lo deseado.

## **Conclusiones**

Como el paradigma orientado a objetos no es sencillo de ver y tiene muchos elementos abstractos, costo generar la idea de los diagramas y código en general, logrando un chatbot limitado, pero “eficiente” en lo especifico.

Las limitaciones se debieron a las pocas posibilidades de respuesta creadas para el chatbot, siendo posible responder solo a 4 preguntas dichas por el usuario, falta imaginación para lograr lo cometido.

En este laboratorio, al igual que el anterior, no termino conforme con el resultado final, siento que se pudo haber implementado de mejor forma, pero la comprensión del paradigma y como dirigir la programación fue la que termino retrasando el desarrollo correcto de este laboratorio.

Comparando este paradigma con el anteriormente tratado, me resulto mas cómodo puesto que ya se veía la existencia de variables editables, (de lo cual siempre me quejare de los anteriores) pero a su vez se solicitaba algo un poco mas complejo que los anteriores puesto que el alcance de la POO es mucho mayor.

## **Referencias**

1. <http://rockeromaniaco.blogspot.es/1408916012/paradigmas-de-programacion/>
2. PPTs de la clase, <https://www.udesantiagovirtual.cl/wp/>
3. <http://cs.lmu.edu/~ray/notes/paradigms/>
4. <https://users.dcc.uchile.cl/~bebustos/apuntes/cc30a/TDA/>
5. <https://es.wikibooks.org/wiki/Programaci%C3%B3n_Orientada_a_Objetos/Caracter%C3%ADsticas_de_la_POO>
6. <https://medium.com/@richardeng/a-simple-explanation-of-oop-46a156581214>