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

LABORATORIO 3

PARADIGMAS DE PROGRAMACIÓN

Catalina Morales Rojas

Profesor: Roberto González
Fecha de Entrega: 10 de agosto de 2018

Santiago de Chile

1 - 2018

TABLA DE CONTENIDOS

TABLA DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

Los chatbots corresponden a programas que permiten mantener una conversación con el usuario, incorporan en su implementación sistemas de inteligencia artificial, lo cual les permite aprender a lo largo de la conversación gustos y preferencias. Las conversaciones que se establecen pueden ser a través de voz o texto. Chatbots conocidos son Siri o cortana, que cuentan principalmente con funciones de búsqueda. El área de servicios es la que cuenta con una mayor implementación de chatbots en la actualidad debido a que ahorran tiempo a los usuarios e incluso adhieren publicidad en su uso.
El objetivo del informe es presentar la manera en que fue planteado, pensado, diseñado y aplicado el laboratorio 3 del curso de Paradigmas de programación, el cual se encuentra implementado en el lenguaje de programación java, bajo el paradigma orientado a objetos.

El paradigma orientado a objetos o POO, toma a los objetos como entidades que tienen un estado, atributos y métodos específicos. Su concepto clave es la reutilización de código con mecanismos como la herencia, polimorfismo, asociación, entre otros.

Los objetivos del laboratorio son:

- Aplicar los conceptos aprendidos en cátedra y en la investigación personal en la creación del chatbot, haciendo uso de buenas prácticas de programación.
- Realizar una correcta implementación del paradigma orientado a objetos, sin faltar a sus estándares.
- Comprender la estructura de las clases y objetos.
- Apreciar las ventajas y desventajas del paradigma orientado a objetos para el desarrollo del laboratorio con respectos a los utilizados anteriormente.
- Modelar el análisis y diseño de la solución mediante el uso de diagramas UML.

CAPÍTULO 2. MARCO TEÓRICO

2.1. PARADIGMA ORIENTADO A OBJETOS

El paradigma orientado a objetos busca acelerar la productividad mediante la reutilización de código y que el pensar al programar sea cercano al mundo real. En la POO, un tipo de dato abstracto puede ser representado por medio de una clase, la cual relaciona atributos (características) y métodos (comportamiento), por lo que puede ser llamada como un plano o molde del objeto. El paradigma cuenta con una serie de conceptos, entre los que se encuentran:

1. Abstracción: Este proceso consiste en dejar de lado los detalles para obtener solo lo fundamental, lo cual permitirá representar un ente a través de sus principales atributos.
2. Herencia: En lugar de definir nuevamente los datos abstractos, se heredan los atributos y métodos, además de la posibilidad de agregar nuevos si es necesario y redefinir métodos.
3. Polimorfismo: Este proceso se realiza cuando existen diferentes comportamientos para distintos objetos, pero que pueden compartir el nombre, o dicho de otra forma, según el mismo nombre se realizará el comportamiento correspondiente al objeto.
4. Encapsulación: Busca que los detalles de la implementación no sean visibles para los usuarios.
5. Garbage collector: corresponde a una técnica que libera la memoria de objetos que no se están utilizando sin la intervención del programador.

CAPÍTULO 3. DESCRIPCIÓN DEL PROBLEMA

Se solicita crear un chatbot en el lenguaje Java, bajo el paradigma orientado a objetos, el cual se encontrará orientado a un contexto o temática. El programa debe contar con las siguientes clases:

1. **Chatbot**: Estructura con diversos parámetros que refleja la personalidad del chatbot y un Seed que se encarga de generar la variabilidad en las respuestas.
2. **Log**: Es el registro histórico de los mensajes enviados por el usuario y el chatbot, debe indicar emisor del mensaje, fecha en que se genera y ser almacenado para futuras referencias.
3. **Usuario**: Contiene la información útil que puede reunir el chatbot sobre el usuario a medida que avanza la conversación.

Además, el programa debe contar con funcionalidades como: !beginDialog, **!endDialog**, **!Rate, !saveLog y !loadLog** y con diagramas UML tanto a nivel de análisis como diseño.

3.1 ANÁLISIS

Para implementar la solución, es necesario tener claro cuál es la representación del problema y definir el TDA. Es importante destacar, que para este paradigma la solución debe ser representada mediante el uso de clases, tal como lo describe el enunciado, en donde cada una tendrá sus atributos y métodos. A continuación, se presenta el diagrama UML con respecto al análisis inicial con que se abordó el problema.

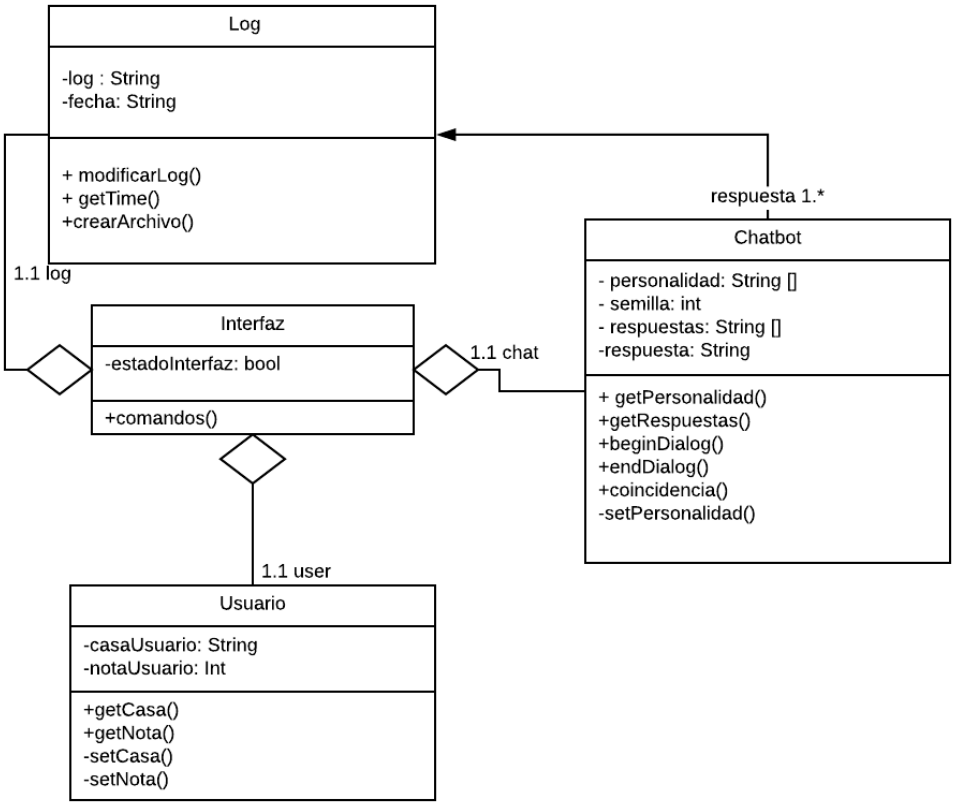

Figura 1.- Diagrama inicial de análisis del problema.

Lo primero que se debe realizar es definir la representación mediante las clases que tendrá el laboratorio, para este caso, en la clase Chatbot es donde se encuentran todas las posibles respuestas y personalidades con las que interactuará el usuario, además de la semilla que permite variar las respuestas entregadas, por lo que como base deben existir métodos que permitan obtener la respuesta en base a lo ingresado por el usuario.

La clase Log será la encargada de modificar, almacenar y cargar el desarrollo de la conversación (cuando se requiera cargar una conversación) durante la ejecución del programa, además de agregar datos relevantes tales como la hora y el emisor del mensaje.

La clase Usuario almacena los datos importantes que son ingresados por el usuario durante las interacciones que tiene con el chatbot, tales como la casa a la cual pertenece, el nombre del usuario y la nota que el mismo asigna con respecto a su comportamiento.

Analizando el problema mediante el uso del TDA de seis capas, el programa debe contar con constructores en alguna de sus clases, en este caso será en chatbot y log. En la clase chatbot el constructor se encargará de definir la personalidad con que interactuará el usuario y la lista de

palabras claves con que se generará el vínculo entre lo ingresado por el usuario y la respuesta del Chatbot. En el caso de log, al momento de instanciar un objeto se generará un string vacío.
Además, en todas las clases se deben incluir métodos setter y getter, los cuales se encargarán de realizar el proceso de encapsulación de los atributos (para el TDA corresponde a los modificadores y selectores).
Finalmente se debe contar con otros métodos, que permitan entregar la hora y los números "random" que se deben utilizar para generar el identificador de una conversación o para obtener una de las posibles respuestas.
Se generan las clases solicitadas en el enunciado junto con la clase interfaz, la cual será encargada de manejar la interacción entre lo que solicita el usuario y lo que debe retornar el chatbot.

CAPÍTULO 4. DISEÑO DE LA SOLUCIÓN

El chatbot implementado es un guía para un estudiante nuevo en la escuela ficticia Hogwarts de magia y hechicería, al cual se le podrán realizar una serie de preguntas.
La representación del TDA chatbot en este caso corresponde a dos ArrayList (implementados en la clase chatbot), el primero contiene una lista con todas las posibles respuestas a las frases ingresadas por el usuario:

respuestas (chatbot, [" frase1", "frase2", "…", "fraseN"]).

El segundo corresponde a una lista con las "palabras claves", tal como se muestra a continuación:

palabrasClave (["Plabra1", "Palabra2", "…"," PalabraN"]).

Cada una de las palabras claves consta con tres posibles respuestas y tres posibles personalidades (chatbot formal, simpático y enojado), la cual es asignada al inicio del programa. El log en este caso será manejado como un string, por lo que, al momento de generar la primera interacción, este corresponderá a un string vacío ("").
El principal método que permite el funcionamiento del programa corresponde a coincidencia (que se encuentra en la clase chatbot), en donde se compara la frase ingresada por el usuario con la lista de palabras claves, cuando encuentra el valor del índice en que se encuentra la respuesta, este se ingresa a una fórmula, la cual depende del valor de la semilla y la personalidad que está interactuando con el usuario y finalmente entrega la respuesta.
Los métodos BeginDialog y endDialog tienen un comportamiento similar, ya que ambos mediante coincidencia buscan con una palabra clave, un mensaje de bienvenida y despedida, que depende de la hora en que se realice la conversación.
Cada clase se encarga de realizar tareas específicas, pero que necesitan al resto de las clases para funcionar de manera correcta:

1. Chatbot: Cuenta con una "base de datos" con las respuestas que tiene el programa, sus atributos corresponden a características que representan a un chatbot tales como su personalidad, la semilla con la que elige la variabilidad de las respuestas, las palabras clave con que se asocian las preguntas entregadas, entre otros. Entre sus métodos cuenta principalmente con selectores que permiten responder al usuario de manera correcta (coincidencia, beginDialog y endDialog).

2. Log: Sus atributos corresponden a un string con la conversación, el cual a lo largo que avanza el dialogo se debe modificar, por lo cual debe contar también con métodos que permitan esto. También es el encargado de crear archivos con las conversaciones que se tienen y de cargar archivos de conversaciones ya finalizadas.
3. Usuario: Almacena datos relevantes del usuario, como por ejemplo su casa y la lista de palabras más buscadas a lo largo del desarrollo de la conversación, las cuales son referenciadas por el chatbot en caso de que el usuario no sepa que preguntar.
4. Interfaz: Es la encargada de interactuar con el usuario, recibiendo los comandos. Regula la conversación, por lo cual entre sus atributos se encuentra el estado en que está la conversación, así como también su valides. Corresponde a la clase que tiene la mayor cantidad de relaciones, ya que maneja de alguna manera la ejecución del programa.

   Evaluaciones: Esta clase contiene las notas que son colocadas por el usuario, por lo que entre sus atributos se encuentras las calificaciones (tanto del usuario como del chatbot) y cuenta con métodos que permiten analizar esta información, como lo son el cálculo del promedio y la desviación estándar.

Como se puede ver en el diagrama, se cuenta con relaciones tales como asociación, agregación y composición. Los acoplamientos que se dieron en el programa fueron los siguientes:

- La asociación se da entre las clases chatbot e interfaz, ya que interfaz necesita en algún momento atributos de la clase chatbot para realizar métodos.
- La agregación se produce entre las clases log y chatbot, debido a que al momento de encontrar una coincidencia entre lo ingresado por el usuario y lo contenido en la clase chatbot, esto debe ser agregado al log, por lo cual se modifican atributos de una clase dentro de la otra, pero ambas clases pueden existir de manera independiente.
- Por último, la composición se da entre las clases log, evaluaciones y usuario con la clase interfaz. Corresponde a composición debido a que las tres primeras clases son instanciadas en la clase interfaz, por lo cual el tiempo de vida de interfaz limita al resto.

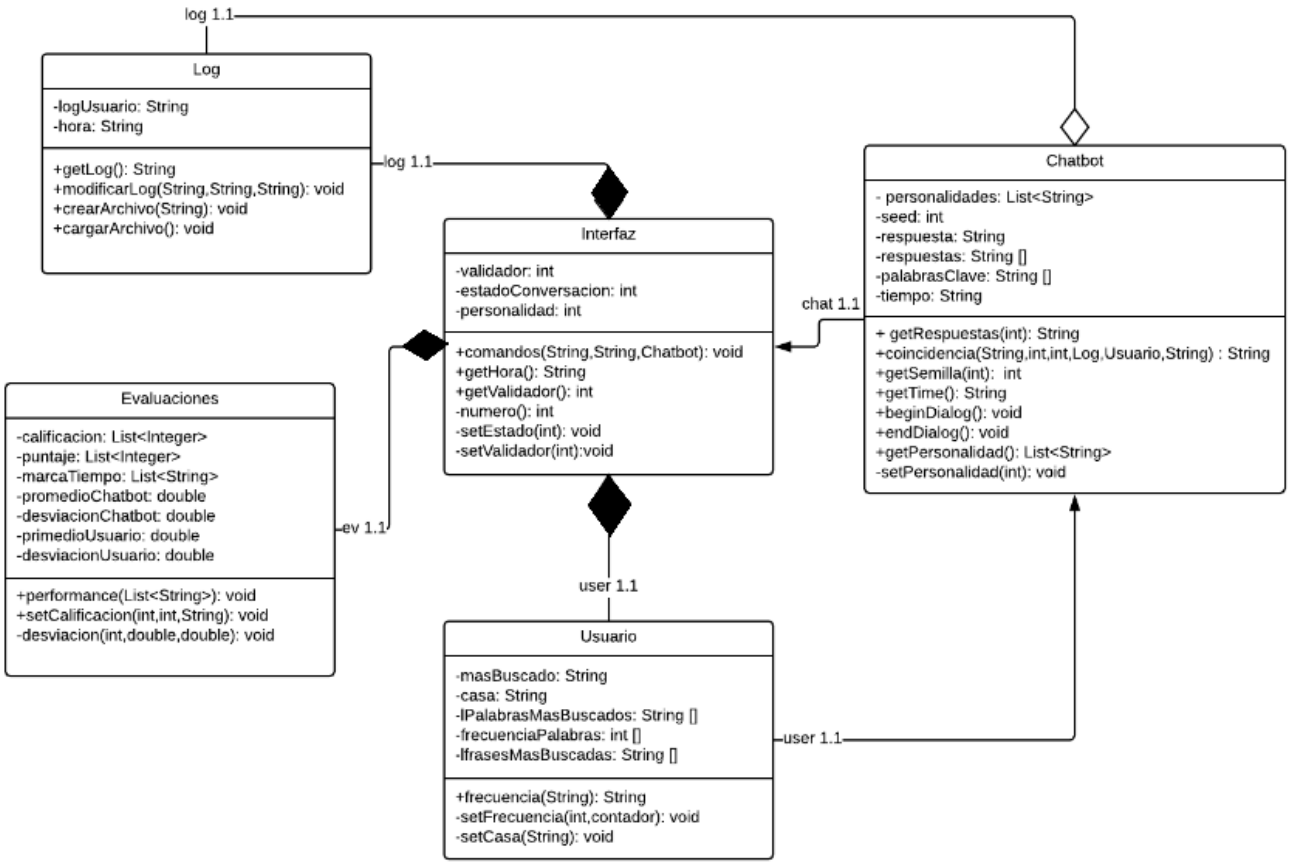

Figura 1.- Diagrama final del desarrollo del problema.

CAPÍTULO 5. IMPLEMENTACIÓN

La implementación del programa cuenta con algunos cambios con respecto a la idea original, ya que se agrega la clase Evaluaciones, debido a que de lo contrario se generaría un mayor grado de acoplamiento entre el resto de las clases para lograr asignar las notas al chatbot y al usuario. Además las relaciones de acoplamiento cambiaron en un grado leve, esto se debe a que a medida que se implementaba el programa surgian problemas sobre como asociar los elementos.

5.1 PROGRAMA UTILIZADO

Para ejecutar el código se utilizó el sofware java jdk, el cual permite compilar el programa.

CAPÍTULO 6. RESULTADOS

El resultado del laboratorio corresponde a un chatbot que puede responder consultas del usuario, guardar y/o cargar una conversación y asignar una nota al programa. Debido a que el chatbot es un sistema de consulta, la memoria funciona ya que almacena la información de la lista de palabras más buscadas, para sugerirlas cuando el usuario ingrese una consulta desconocida.

Al iniciar el programa, HogwartsBot entrega un mensaje de bienvenida indicando que es lo que realiza el chatbot. Si el usuario no ha iniciado una conversación con !beginDialog no podrá interactuar con el chatbot y se entregará un mensaje de error, lo mismo pasa si se intenta iniciar una nueva conversación mientras se encuentra una en curso o terminar una conversación que aún no se ha iniciado. Para !rate sucede algo similar, ya que mientras no se termine la conversación, no será posible evaluar el desempeño del chatbot.

```
Air-de-Catalina:lab3-paradigmas cata$ java HogwartsBot
[                                                                                                                    ]


                *******Bienvenido a Hogwarts! HOGWARTSBOT te guiará por todos los rincones del Castillo, disfruta tu estadía*******
          (-O-O-)/



<15:07:00:17:07:2018> <Usuario> !beginDialog
<15:07:00:17:07:2018> <HOGWARTSBOT> Buenas tardes, bienvenido a Hogwarts, hoy sere tu guia en el castillo,  En que puedo ayudarte?
<15:07:04:17:07:2018> <Usuario> quiero informacion sobre ravenclaw
<15:07:04:17:07:2018> <HOGWARTSBOT> Para entrar a la sala comun de ravenclaw, se debe resolver un acertijo que no siempre es el mismo, formulado por un ald
aba de bronce con forma de aguila en una puerta sin picaporte. Que mas quieres saber?
<15:07:21:17:07:2018> <Usuario> que son los dementores ?
<15:07:21:17:07:2018> <HOGWARTSBOT> Los Dementores estan entre las criaturas mas nauseabundas del mundo. Infestan los lugares mas oscuros y mas sucios. Qui
eres saber algo mas?
<15:07:37:17:07:2018> <Usuario> !saveLog
<15:07:37:17:07:2018> <HOGWARTSBOT> Se ha creado el archivo
<15:07:46:17:07:2018> <Usuario> !endDialog
<15:07:46:17:07:2018> <HOGWARTSBOT> Hogwarts es la mejor escuela del mundo magico, disfruta mucho, que pases una buena tarde!
<15:07:55:17:07:2018> <Usuario> !exit
Air-de-Catalina:lab3-paradigmas cata$
```

Figura 3.- Ejemplo conversación con el chatbot.

En el caso de ocurrir errores a lo largo del programa, estos son manejados utilizando principalmente try/catch para las excepciones, así se genera un mensaje informando lo sucedido al usuario y solicitando ingresar nuevamente los datos.

El resultado de avance de las funcionalidades obligatorias fue el requerido en el enunciado para: !beginDialog, !endDialog, !saveLog, !loadLog y !rate. Y en el caso de las funciones extras, solo fue implementada !chatbotPerformance.

```
Air-de-Catalina:lab3-paradigmas cata$ java HogwartsBot


                *******Bienvenido a Hogwarts! HOGWARTSBOT te guiará por todos los rincones del Castillo, disfruta tu estadía*******
          (-O-O-)/



<15:28:10:17:07:2018> <Usuario> !loadLog aasasdasd
No fue posible abrir el archivo: aasasdasd
```

Figura 4.- Ejemplo caso de error

CAPÍTULO 7. CONCLUSIONES

Debido a que el paradigma orientado a objetos permite representar de manera sencilla la realidad, la abstracción para desarrollar el problema fue más sencilla que en los paradigmas anteriores. La POO permite agrupar elementos que cuentan con comportamientos similares, por lo cual es posible hacer uso de la reutilización del código.

En comparación a los paradigmas utilizados con anterioridad (lógico y funcional), se cuenta con una mayor cantidad de herramientas para desarrollar el programa, como lo son el uso de los ciclos, por lo cual no es necesario en este caso hacer uso de la recursión.
Otra de las diferencias, es la posibilidad de modificar los valores de los elementos o atributos de cada una de las clases, lo que permite generar la memoria dentro del chatbot (esto no fue posible en el laboratorio anterior). Por último, la extensión del código es bastante mayor con respecto a los programas anteriores.

Dentro de los problemas que presenta el uso del paradigma orientado a objetos se encuentra el acoplamiento, ya que resulta casi imposible no vincular las clases, debido a que deben interactuar para que el programa funcione.
Por otro lado, el diagrama UML fue una gran ayuda para establecer las bases del programa y modificar sólo algunas variantes, no toda la representación como ocurrió en el caso de la implementación con los otros paradigmas.

Con respecto a los objetivos planteados al inicio, se logró cumplir con la aplicación de los conceptos aprendidos en cátedra y en la investigación personal para la creación del chatbot, haciendo uso de buenas prácticas de programación, comprender la estructura del paradigma lógico y apreciar las ventajas y desventajas del paradigma para el desarrollo del laboratorio, además de la representación y modelamiento del análisis y desarrollo del problema utilizando los diagramas UML.

BIBLIOGRAFÍA

- Anónimo, 2009 "Paradigma orientado a objetos" (Recuperado 19/05/2018) https://kevinldp.wordpress.com/4-paradigma-orientado-a-objetos/