Informe Laboratorio 3
Paradigma de Programación Orientada a Objetos

Estudiante: Benjamín Zúñiga J.

Asignatura: Paradigmas de Programación

Profesor: Roberto González I.

Diciembre 2023

Tabla de Contenido

Introducción

En la actualidad el masivo aumento del uso de las tecnologías digitales en la mayoría de los ámbitos de la vida cotidiana ha llevado a que el uso de un chatbot puede ser algo muy útil para cualquiera que quiera automatizar sus procesos, debido a que este una vez creado e implementado este en una web, aplicación, etc. Permite interactuar con usuarios a tiempo completo y entregar respuestas a estos de manera casi instantánea, a parte que tampoco significa un costo extra a quien esté buscando implementarlo para el uso particular que le quiera dar, todo esto ha provocado que en el último tiempo la "fama" e interés sobre los chatbots ha aumentado de gran manera. Ahora en lo que respecta a este informe, se presentaran distintos apartados tales como: una descripción del problema, una descripción del paradigma, un análisis del problema, diseño de la solución, aspectos de implementación, instrucciones de uso, resultados y conclusión, apartados que son importantes para entender cómo se llegó a la solución alcanzada del problema que se presenta para este trabajo, el cual es crear un sistema de chatbots a través del paradigma de programación orientada a objetos, el cual al ser parte del paradigma de programación imperativo se enfoca más en "Como" resolver por sobre el "Qué y más concretamente será una implementación por medio del lenguaje de programación Java que implementa el paradigma de programación orientada a objetos de manera más "pura" que otros lenguajes que implementan este paradigma.

Descripción del problema

Para efectos de este trabajo tal como fue mencionado en el apartado anterior el problema que se busca solucionar es la creación de un sistema de chatbots ITR (Respuesta de Interacción a Texto) que permita a un usuario interactuar con este mediante texto y luego ser derivado a un chatbot especialista en el tema que quiera el usuario, a su vez cada uno de estos chatbots deben contener flujos, los cuales van a permitir "desplazarse" por el sistema y al mismo tiempo estos flujos deben contener opciones que finalmente son las que permiten interactuar al usuario que pueda llegar a utilizar este sistema, además se debe crear un menú que funciones por la consola/terminal de un computador o una interfaz gráfica que permita a un usuario usar todos los requerimientos que posee este trabajo, ahora en cuanto al "como" se debe tratar de lograr la creación de este sistema de chatbots es mediante el paradigma de programación orientada a objetos, por lo cual inicialmente se necesita representar muchas de las partes del sistema mencionadas anteriormente como clases y que a la vez estas funcionen implementando interfaces y/o heredando de alguna superclase .

Descripción del paradigma

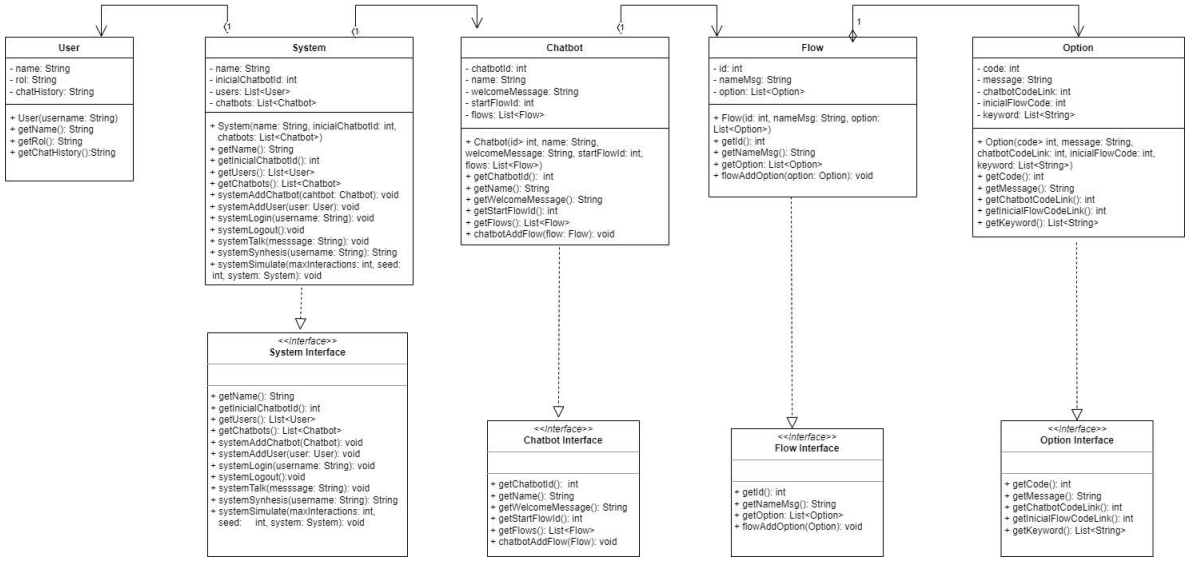
Tal como fue mencionado anteriormente el paradigma sobre el cual se va a basar la solución propuesta próximamente es en el paradigma de programación orientada a objetos, el cual tiene como unidad de abstracción básica los objetos, los cuales pertenecen a una clase, y poseen atributos y métodos, donde objeto lo podemos entender como una instanciación de una clase, clase como "categorías de objetos" (Pavón, J. 2007, Fundamentos de la

Programación Orientada a Objetos), atributos como “valores o características de un objeto” (Pavón, J. 2007, Fundamentos de la Programación Orientada a Objetos) y métodos como “acciones que puede realizar un objeto” (Pavón, J. 2007, Fundamentos de la Programación Orientada a Objetos). También es necesario entender el concepto de interfaz o “interface”, el cual es una especie de “contrato” que puede ser implementado por una clase, en el cual se enuncian los atributos y/o métodos que debe tener la clase que lo implemente, con sus respectivos parámetros de entrada y tipos de datos. Por otra parte puede ser de beneficio entender algunas de las relaciones que existen en este paradigma y que significan, donde destaca herencia, la cual se puede entender como “el mecanismo de implementación mediante el cual elementos más específicos incorporan la estructura y comportamiento de elementos más generales” (Ponce de León, P y Cachero, C. s.f. Programación orientada a objetos) y agregación que se puede comprender como una relación en la cual una clase u objeto posee otro pero no es responsable de su tiempo de vida. Dentro de las bondades del paradigma podemos encontrar que a diferencia de otros paradigmas la forma de abstraer y representar cosas es más entendible humanamente hablando y dentro de las limitaciones, se encuentra que al pertenecer al paradigma imperativo, no basta con saber el “qué” resolver, ya que también es muy importante el “cómo”, por lo cual siempre va a ser necesario diseñar un algoritmo o método que pueda solucionar el problema identificado.

Análisis del problema

Al profundizar en el problema, cabe mencionar que la creación de este sistema de chatbots tiene algunos requisitos y/o restricciones específicas principalmente relacionados a los parámetros de entrada de cada método, es decir, el tipo de dato que ingresa al método también el cómo se deben implementar los TDAs que representaran las “partes” que componen este sistema de chatbots, tales como los chatbots, flujos, opciones, usuarios y el sistema, ya que a la hora de construir estos, se pide expresamente que cada uno sea una clase e implementen interfaces, a excepción de los usuarios , que se permite que sean implementados mediante herencia de algún tipo de superclase, debido a que existen dos tipos de usuarios los cuales pueden ser tanto comunes, como administradores, lo cual afecta a cuantas funcionalidades va a poder acceder mediante el menú posteriormente, aparte de las restricciones respecto a los parámetros de entrada de cada método, los constructores de los chatbots, flujos y opciones manejan un id que sirve como verificador para evitar que alguna de estas estructuras esté repetida dentro de otra y evitar errores, además es necesario implementar métodos que permitan registrar usuarios, en el sistema una única vez, es decir, no poder registrar dos veces el mismo usuario, o registrar un usuario común y administrador que se llamen igual, luego que solo alguno de los usuarios registrados de pueda “loguear” y que solo exista un usuario logueado a la vez, y solo si es que existe un usuario logueado pueda interactuar con el sistema, guardando su interacción en un historial propio de ese usuario, también implementar un predicado de “deslogueo” el cual permite quitar del estado logueado al usuario que lo este y por ultimo implementar un predicado que permita un simulación pseudoaleatoria de conversación con el sistema, donde el máximo de interacciones del sistema esté dado por un número que el usuario entregue. A continuación se adjunta el diagrama de análisis UML que

representa de manera gráfica lo mencionado anteriormente, incluyendo las clases, métodos y relaciones que se creen inicialmente como necesarias y suficientes para solucionar el problema.



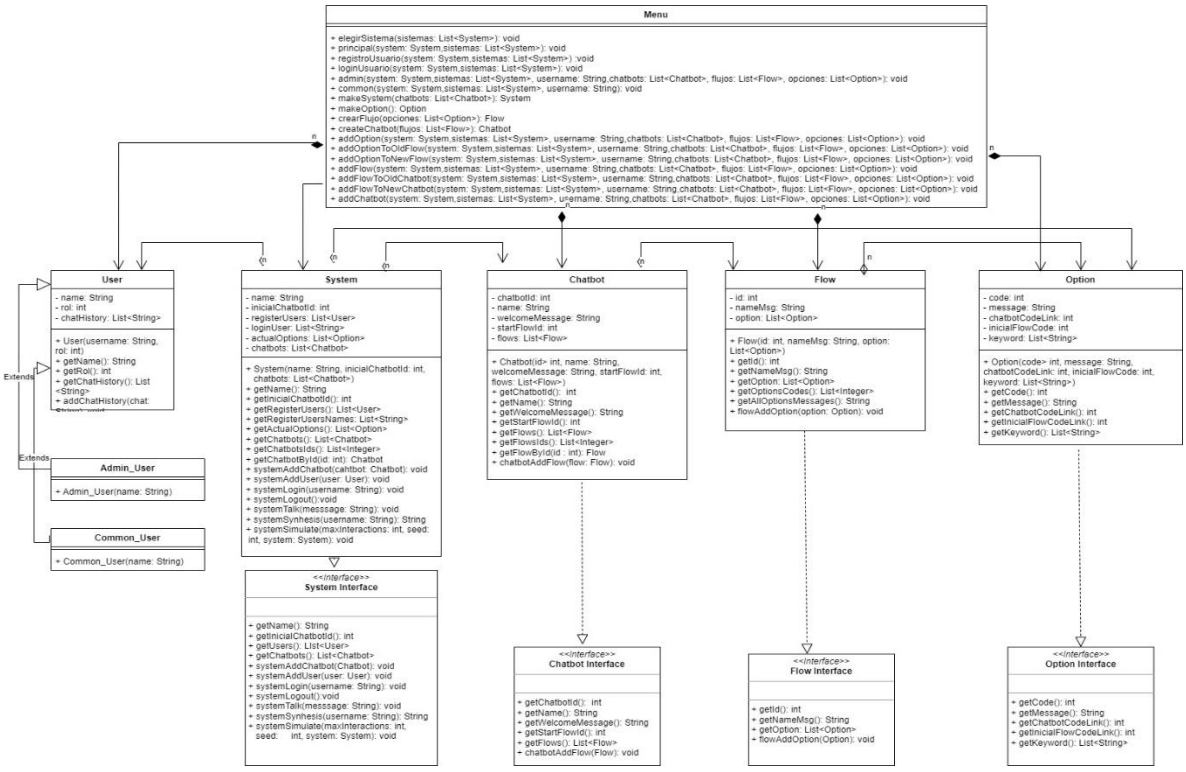
(*Diagrama 1: Diagrama de Análisis UML*)

Diseño de la solución

Respecto al diseño de solución, es importante comenzar mencionando que el principal enfoque fue la funcionalidad de tanto los TDAs como los propios métodos, es decir, lo principal fue lograr que cada predicado y TDA cumpliera su objetivo “como fuera” por sobre otros aspectos, como la optimización, entre otros, siempre respetando los requisitos específicos de cada método y TDA, , en este punto clave aclarar que una clase es una implementación de un TDA, por lo que para efectos prácticos desde ahora se referirá a cada TDA como clase.

En relación a la descomposición del problema lo primero realizado fue intentar graficar un modelo del sistema para comprenderlo visualmente y entender en qué lugar del modelo va cada estructura y cuales su función, cada estructura fue representada mediante una clase distinta, en este caso las diseñados y empleadas fueron la clase Option, Flow, Chatbot, System, User, siendo tanto Option como User los TDAs más “simples” se podría decir como clase debido a que contienen estructuras y tipos de datos nativos del lenguaje, mientras que Flow, Chatbot y System contienen otras clases, como se puede ver en la imagen 1 en la sección “Anexos” y en un diagrama de diseño UML al final de esta sección, que representa de manera gráfica lo mencionado anteriormente, incluyendo las clases, métodos y relaciones que fueron efectivamente implementadas para llegar a la solución.

Respecto a algoritmos o técnicas usados para alunas solucionar problemas particulares, no se usaron ningunas en particular.



(Diagrama 2: Diagrama de diseño UML)

Aspectos de implementación

Respecto a la implementación esta fue desarrollada en el ide IntelliJ en su versión Community Edition 2023 2,5 , el cual es para programar en el lenguaje Java, en el desarrollo de este fue utilizada la clase java.util y la estructura de la implementación del proyecto es la creada por el ide de manera automática, donde lo más importante que se debe mencionar es que el proyecto está integrado con gradle y dentro de la carpeta "src" está la carpeta "main" y dentro de esta está la carpeta "java", que contiene la carpeta "Codigo_fuente" donde está el código de la solción, y esta carpeta consta de 13 archivos, contando el main, 4 interfaces, las de Option, Flow, Chatbot y System y 8 archivos de clase, dentro de los cuales están Option, Flow, Chatbot, System, Common_User, Admin_User, User y Menu. Por último cabe recalcar que el proyecto fue realizado usando JDK versión 11.0.2.

Instrucciones de uso

Relacionado a las instrucciones de uso es importante dejar en claro que se asumió que el usuario va a ingresar cosas coherentes y respetar los parámetros de entrada método a la hora de utilizar el menú, para ejecutar el programa se debe abrir la terminal del computador e ingresar la dirección de la carpeta general del proyecto y después ejecutar el comando ".\gradlew.bat run", que iniciará el programa y permitirá interactuar al usuario mediante la terminal. (Ejemplo de ejecución del programa en imagen 2 en la sección "Anexos")

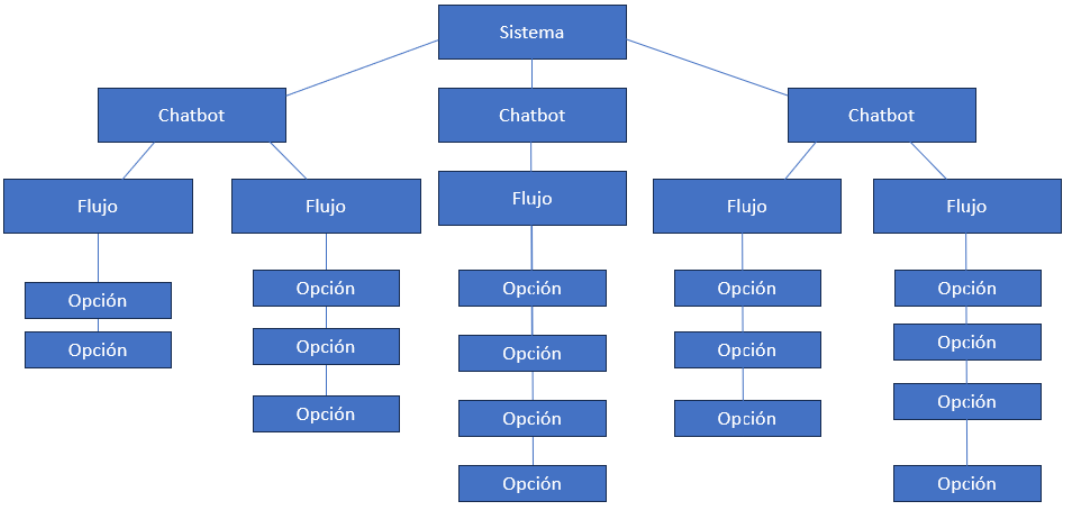
Resultados y Autoevaluación

Los resultados alcanzados fueron bastante satisfactorios, ya que se logró implementar todos los requisitos funcionales de buena manera, cumpliendo sus requisitos, crear las clases, interfaces, métodos y el menú según correspondía, todo lo anterior sin ninguna dificultad mayor, considerando el grado de dificultad inherente a cada requisito funcional y cumpliendo los requisitos específicos para cada una de estos, aparte se logró implementar métodos no obligatorios tanto el paradigma como los conceptos aprendidos más en profundidad y a su vez facilitaron la implementación de otros métodos. El proceso de prueba para obtener los resultados mencionados anteriormente fue ejecutar el programa tanto en la terminal del computador como la del IDE, se puede observar el resultado una prueba del programa en la "Imagen 3" disponible en la sección "Anexos" de este trabajo. Por último existe una tabla de autoevaluación disponible en la sección "Anexos" como "Tabla 1" indicando el puntaje que se cree haber obtenido en cada requerimiento funcional de este trabajo.

Conclusión

Finalmente a modo de conclusión se puede decir que se logró solucionar el problema de buena manera, como se mencionaba en el apartado "Resultados y Autoevaluación", es importante considerar que el alcance la solución es mayor a la de los anteriores laboratorios, debido a que se simula de buena manera y funciona, pero a diferencia de los otros laboratorio, este incluye el menú que permite al usuario interactuar en tiempo real, haciendo así que esta implementación pudiera ser utilizada en un contexto de la vida real, también cabe considerar que a diferencia de los laboratorios anteriores al no tener que usar recursión ni que los tipos de datos estén construidos en base a esta, se reduce considerablemente el tiempo de interacción, por otra parte ya comparando la implementación realizada mediante el paradigma funcional y lógico, la mayor dificultad en este laboratorio en comparación a los anteriores es que al venir desde el paradigma declarativo, que agrupa al paradigma funcional y lógico, y basarse más en el "Qué" que en el "Cómo" los lenguajes de por si ofrecían mecanismos más complejos que facilitaban la implementación del sistema de chatbots, recalcando la variedad de funciones en Racket y el backtracking automático de Prolog, es por esto que a la hora de iniciar este laboratorio el hecho de tener que idear y/o diseñar el "Cómo" implementar cada requisito fue una tarea bastante difícil, por suerte la mayoría de clases funcionaban de maneras similares, por lo cual una vez implementado un requisito, terminaba siendo de mucha ayuda posteriormente para la implementación de otro requisito.

Anexos



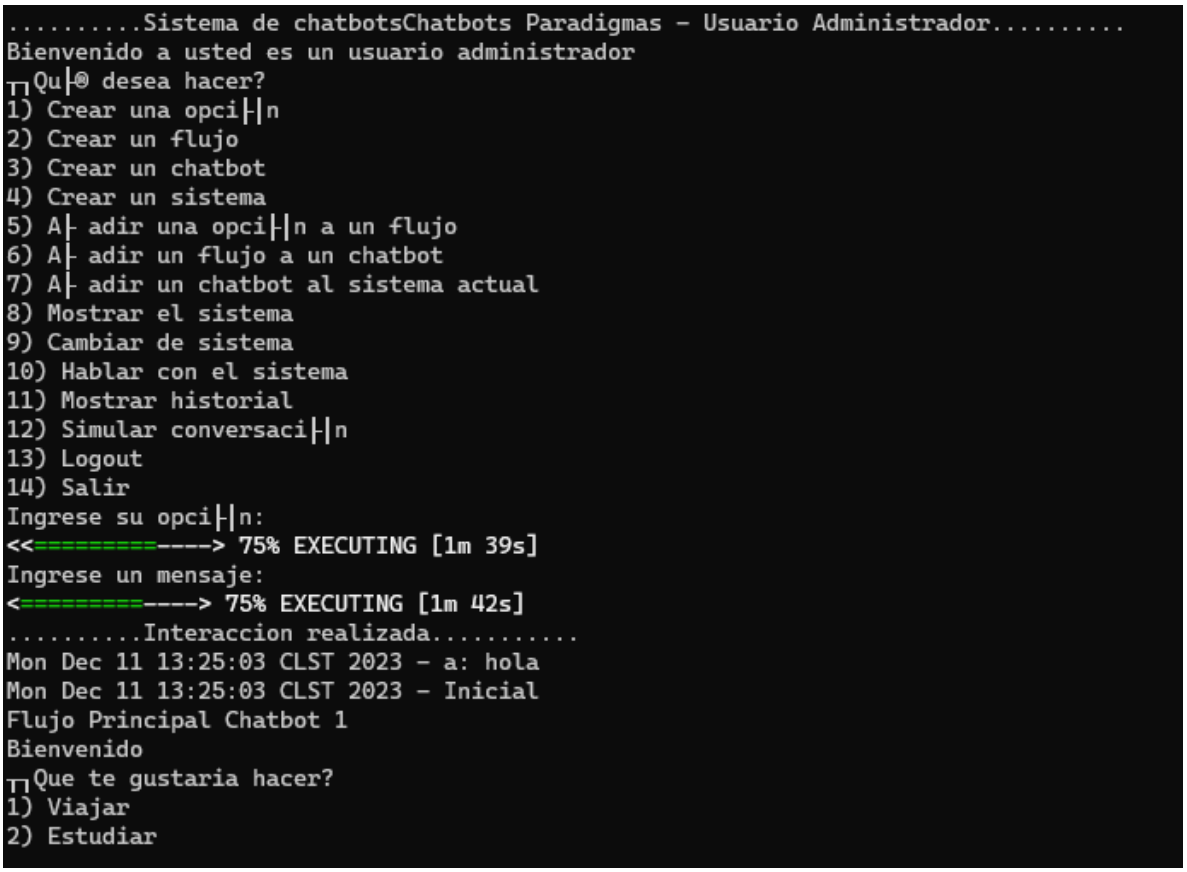
(Imagen 1: Representación del sistema)

```
PS C:\Users\benja\Desktop\Programación\Java\lab3_21337525_ZunigaJofre> .\gradlew.bat run
Starting a Gradle Daemon, 1 incompatible and 1 stopped Daemons could not be reused, use --status for details

> Task :run
..........Seleccion de sistema..........
1) Chatbots Paradigmas,0,[],[],[],[0,Inicial,Bienvenido
┬┐Que te gustaria hacer?,1,[1,Flujo Principal Chatbot 1
Bienvenido
┬┐Que te gustaria hacer?,[1,1) Viajar,1,1,[viajar, turistear, conocer], 2,2) Estudiar,2,1,[estudiar, aprender, perfeccio
narme]]], 1,Agenda Viajes,Bienvenido
┬┐Donde quieres viajar?,1,[1,Flujo 1 Chatbot1
┬┐Donde te gustaria ir,[1,1) New York, USA,1,2,[Estados unidos, New York], 2,2) Paris, Francia ,1,1,[Paris, Eiffel], 3,3
) Torres del Paine, Chile,1,1,[Chile, Torres, Paine, Torres Paine, Torres del Paine], 4,4) Volver,0,1,[Regresar, Salir,
Volver]], 2,Flujo 2 Chatbot1
┬┐Que atractivos te gustaria visitar?,[1,1) Central Park,1,2,[Central, Park, Central Park], 2,2) Museos,1,2,[Museo], 3,3
) Ningun otro atractivo,1,3,[Ningun otro atractivo], 4,4) Cambiar destino,1,1,[Cambiar, Volver, Salir]], 3,Flujo 3 Chatb
ot1
┬┐Vas solo o acompa├▒ado?,[1,1) Solo,1,3,[Solo], 2,2) En pareja,1,3,[Pareja], 3,3) En familia,1,3,[Familia], 4,4) Agrega
r m├ís atractivos,1,2,[Volver, Atractivos], 5,5) En realidad quiero otro destino,1,1,[Cambiar destino]]], 2,Orientador A
cademico,Bienvenido
┬┐Que te gustaria estudiar?,1,[1,Flujo 1 Chatbot2
┬┐Que te gustaria estudiar?,[1,1) Carrera T├®cnica,2,1,[Tecnica], 2,2) Postgrado,2,1,[Doctorado, Magister, Postgrado], 3
,3) Volver,0,1,[Regresar, Salir, Volver]]]]

Ingrese el numero del sistema con el cual quiere trabajar:
<=========----> 75% EXECUTING [12s]
> :run
```

(Imagen 2: Ejecución del programa mediante terminal)



(Imagen 3: Menú de usuario administrado y resultado de hablar con un sistema de chatbots)

N° Requerimiento funcional	Puntaje
1	1
2	0,5
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1

(Tabla 1: Puntajes de autoevaluación)

Referencias

- Pavón, J. 2007, Fundamentos de la Programación Orientada a Objetos. Extraido de: https://www.fdi.ucm.es/profesor/jpavon/poo/1.1.objetos%20y%20clases.pdf
- Ponce de León, P y Cachero, C. s.f. Programación orientada a objetos. Extraido de: https://rua.ua.es/dspace/bitstream/10045/15995/1/POO-3-Herencia-10-11.pdf
- Barnes D.J y Kölling, M, 2007. Programación orientada a objetos con Java. Una introducción práctica usando BlueJ. Pearson Educación,
- Budd, T. 2001. An introduction to Object-Oriented Programming (Third
- Edition)