Contenido

[Caso Jedi Order 2](#_Toc8311928)

[Caso Mundial 2018 3](#_Toc8311929)

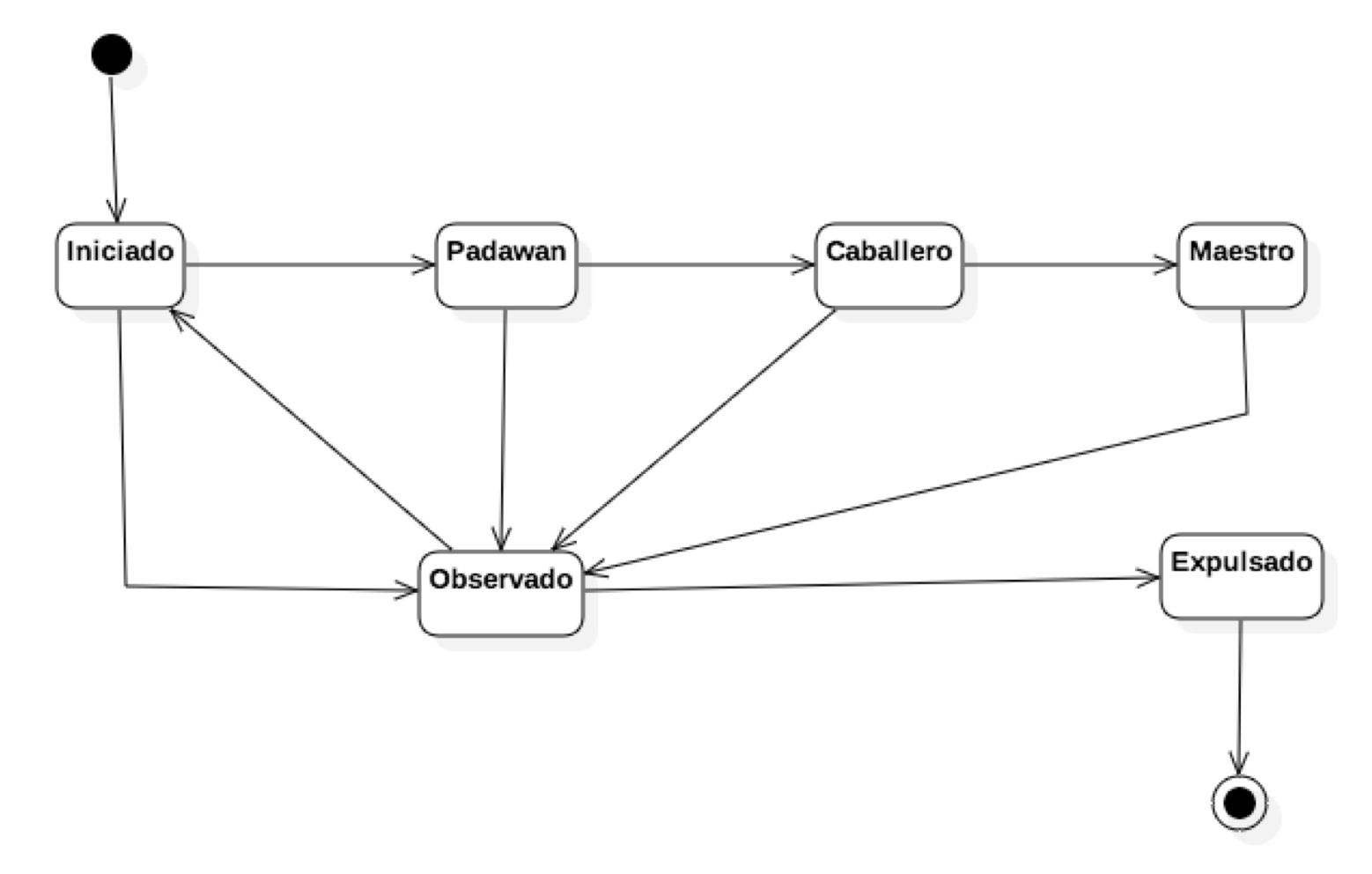
# Caso Jedi Order

Un Jedi era un miembro de la **Orden Jedi**, que estudiaba, servía y usaba las energías místicas de la **Fuerza,** normalmente el lado luminoso de la fuerza. Los Jedi luchan por la paz y la justicia en la República Galáctica, comúnmente contra sus enemigos mortales, los Sith y Jedi Oscuros, que estudiaban el lado oscuro de la Fuerza.

La Orden Jedi (agrupación que acoge a todos los Jedi de la galaxia) ha sido refundada por Luke Skywalker, último Jedi sobreviviente de la gran purga realizada por el emperador Palpatine (Darth Sidious) así como por su aprendiz Darth Vader.

Al refundar la Orden Jedi, se está buscando no volver a cometer los mismos errores del pasado y realizar una evaluación minuciosa de las habilidades y experiencias de sus miembros. Para esto, se le ha contratado para desarrollar un sistema de información que permita evaluar los ascensos de los distintos Jedis pertenecientes a la orden.

Los niveles que pueden tener los Jedis son los siguientes:



Para poder ascender de nivel, cada Jedi tiene que cumplir con una minuciosa evaluación específicamente de 2 variables:

**Nivel de Fuerza:** Cantidad de fuerza que ha desarrollado el Jedi. Notar que si el nivel es negativo el Jedi se encuentra en el lado oscuro.

**Cantidad de misiones exitosas:** Cantidad de misiones que el Jedi ha completado con éxito.

Las condiciones de cambio de nivel son las siguientes:

|  |  |  |
| --- | --- | --- |
| **Nivel Actual** | **Nivel a Ascender** | **Condición cambio de nivel** |
| Iniciado | Padawan | Debe tener más de 10 misiones exitosas así como tener nivel de Fuerza superior a 100. |
| Padawan | Caballero | Debe tener más de 30 misiones exitosas así como tener nivel de Fuerza superior a 200. |
| Caballero | Maestro | Debe tener más de 50 misiones exitosas así como tener nivel de Fuerza superior a 500. |
| Observado | Expulsado | Si estando en el estado Observado tiene un nivel negativo (menor a 0). |
| Observado | Iniciado | Si estando en el estado Observado tiene un nivel de fuerza igual o mayor a cero. |

**Nota Importante:** De cualquier estado (Iniciado, Padawan, Caballero o Maestro) al tener un nivel de fuerza menor a -100, deberá pasarse a estado Observado (el Jedi tiene tendencia a pasarse al lado oscuro).

Crear una clase Evaluador e implementar el método evaluarAscensos.

# Caso Mundial 2018

Como parte de nuestro proyecto queremos hacer una aplicación que se encargue de predecir los resultados de los partidos de la copa del mundo. De esta manera podemos tener una idea

de ver cuantos partidos nuestros usuarios podrían acertar.

Luego de la ejecución de nuestro programa, nuestro programa nos muestra en pantalla la lista de los partidos del grupo de Perú con el pronóstico correspondiente.

Debido a que el algoritmo de predicción utiliza técnicas complejas de inteligencia artificial y toma datos de fuentes diversas, este nos es proporcionado por la librería forecast.js.

Debemos tomar en cuenta que posiblemente en un futuro queramos contar con otra librería por lo que el diseño de nuestra app debería permitirlo.

Reimplementar la aplicación para cumplir con todos los requerimientos mencionados (debe aplicar 3 patrones de diseño).

# Autenticación segura

Se le pide implementar un software que permita el registro seguro y confiable de usuarios para que así estos puedan autenticarse de manera segura. Por esto se ha implementado una clase llamada GestorUsuarios con los métodos necesarios para hacer estas dos tareas.

Por temas de seguridad, al momento de registrar los datos de un nuevo usuario se ha optado por encriptar el password utilizando el algoritmo SHA256, disponible mediante la librería crypto de python. De esta manera, para realizar una autenticación, el software implementado antes encripta el string ingresado por el usuario, y lo compara con la contraseña encriptada que se encuentra en el software.

En un futuro, se piensa trabajar también con otro algoritmo de encriptación llamado AES, esto para mejorar la seguridad de la aplicación.

Se le ha entregado un código en python totalmente funcional que simula el registro de un usuario así como su posterior autenticación, esto utilizando dos algoritmos de encriptación AES y SHA256.

1. Cree un branch en git donde vaya implementando cada patrón. El branch deberá llamarlo con el nombre de los patrones que está utilizando, separados por “\_”.
2. Refactorice el código seleccionado 3 patrones de diseño e implementándolos. Sustente el por qué de su elección (esto como comentarios dentro de su código).
3. Realice un merge en el branch master con todos sus cambios realizados.

# Aprobación de créditos

Estamos realizando el software para un banco y uno de los puntos más importantes es saber quién puede aprobar un crédito. Por lo tanto el banco define las siguientes reglas de negocio:

* Si el monto no supera los $ 10.000 entonces el ejecutivo de cuenta pueda aprobar el préstamo.
* Si el monto está entre los $10.000 y $50.000 entonces la persona indicada para realizar la aprobación es el líder inmediato de dicho ejecutivo.
* Si el monto se encuentra entre $ 50.000 y $100.000 entonces es el Gerente quién debe realizar dicha aprobación.
* Por montos superiores a los $100.000 entonces la aprobación la realizará el Director.

1. ¿Qué patrón deberá utilizar para realizar una implementación sin deuda técnica? Sustente su respuesta.
2. Implemente el patrón elegido. Debe crear un archivo llamado pregunta2.py donde se encuentre todo su código necesario para la implementación del patrón, con una función main de ejemplo que pruebe la ejecución de su patrón.

# Android Killing Process

El Sistema Operativo Android está diseñado para gestionar de manera inteligente los procesos de un dispositivo móvil. Su lógica de funcionamiento es la siguiente:

* En caso que el proceso sea de prioridad 3 y esté consumiendo más de 100 megas, debe ser matado por el agente de nivel 1.
* En caso que el proceso sea de prioridad 2 y esté consumiendo más de 100 megas, debe ser matado por el agente de nivel 2.
* En caso que el proceso sea de prioridad 1 y esté consumiendo más de 100 megas, debe ser matado por el agente de nivel 3.

Se le proporciona el código en el archivo android.py que funciona correctamente pero no cumple con un buen diseño.

# Smart Car

Debido a los últimos avances en inteligencia artificial, especificamente en el campo de machine learning, muchas empresas estan optando por la implementación de servicios autónomos que permitan facilitar la realización de labores complejas y rutinarias de los seres humanos. Una de las labores complejas que se está atacando es la conducción de vehículos. Empresas como Google, Amazon, Uber están desarrollando vehículos que puedan utilizarse sin la necesidad de un conductor y dentro de poco tiempo ya estarán disponibles para ser adquiridos por los usuarios.

En Perú, se ha comenzado a implementar un proyecto que permita el manejo autónomo por parte de automóviles (Proyecto SMARTCAR), en el que participan varias universidades, cada una encargándose de cierto módulo del software. A la Universidad de Lima le ha correspondido el desarrollo del Módulo para la Gestión del Motor. La idea es permitir que el consumo de combustible sea lo más eficiente posible para así poder realizar trayectos más baratos.

El módulo a desarrollar se encargará de interactuar con el software del motor para las acciones principales del automóvil como son:

* Encender el auto
* Acelerar
* Frenar
* Apagar el auto

Luego de apagar el automóvil, el sistema debe de proporcionar la cantidad de combustible gastado y este debe de enviar los resultados al Módulo de Visualización, el cual está siendo desarrollado por otra institución. Además, se plantea que estos datos sean guardados en una base de datos SQLite, para que en una segunda fase, pueda utilizarse algoritmos de machine learning y el sistema aprenda la mejor manera de trabajar con el motor.

Para esta fase del proyecto, se está trabajando con dos fabricantes de Motores (en el futuro pueden ser más) que nos han proporcionado las librerias necesarias para poder interactuar con su hardware. Los fabricantes son:

* Motores GFX
* Yotato Engines

El comando a ejecutar es el siguiente (dentro de la carpeta \dist):

* Para tipo de motor Yotato: : $node smartcar.js yotato
* Para tipo de motor GFX: $ node smartcar.js gfx

El programa deberá pedirle que ingrese una letra y presione enter para que se simule la interacción con los motores. Al detener (parar) el motor, la aplicación le botará la cantidad de gasolina gastada.

Se le pide solucionar la deuda técnica existente en el proyecto, para eso debe realizar los siguiente:

1. Implementar el patrón Adapter.
2. Implementar el patrón Factory para la creación de Adapters específicos. (2 puntos)
3. El proyecto SmartCar es una prueba. Se ha hecho que se ejecute por línea de comandos solamente para probar los resultados de la interacción con las librerías de los motores. En un futuro, este módulo de Gestión del Motor interactuará con otros módulos (ya sean desarrollados por la Universidad de Lima o por otras instituciones) directamente mediante clases Fachada. Se le pide implementar el Patron Fachada que asegure la cohesión del módulo que ha implementado.

D. Implementar el patrón Singleton para las clases Factory y Fachada creadas.