Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Programación Orientada a Objetos

Familiarización Con el paradigma

Hechos QUE CONECTAN

Introducción

- Se han trabajado los fundamentos de la programación y se han aplicado al entorno de Python.

- La Programación Orientada a Objetos (POO) no necesariamente cambia lo estudiado.

- Simplemente consiste en una forma diferente de pensar o abstraer el problema que estamos resolviendo algorítmica o computacionalmente.

Forma de Representación

- El enfoque no consiste en pensar únicamente en contenedores de datos y la actualización de los mismos para resolver el problema.

- En este enfoque todos estos elementos se integran (variables y funciones de actualización) para representar elementos relevantes del mundo real que queremos representar en el algoritmo o sistema de información que estamos creando.

Forma de Representación

Forma de Representación

- Estos elementos (entidades) se crean en virtud de sus atributos, que eran las variables y arreglos que contenían la información relevante en la programación estructurada que hemos visto.

- E igualmente se representa el comportamiento de dichas entidades en virtud de funciones como las que usábamos previamente, pero organizadas para actualizar los atributos de cada una de estas entidades identificadas en la realidad que estamos representando.

Programación Orientada a Objetos

Podríamos definir la Programación Orientada a Objetos (**POO** u **OOP** en inglés) como:

- Una forma de programar en la que se plantean las cosas intentando realizar una asociación con objetos de la vida real, y expresándolas mediante un conjunto determinado de técnicas de programación.

Programación Orientada a Objetos

Por ejemplo, si pensamos en automóviles, nos daremos cuenta que tienen aspectos en común:

- Determinadas características.
- Realizan las mismas tareas en muchos casos.

Programación Orientada a Objetos

Características (Atributos):

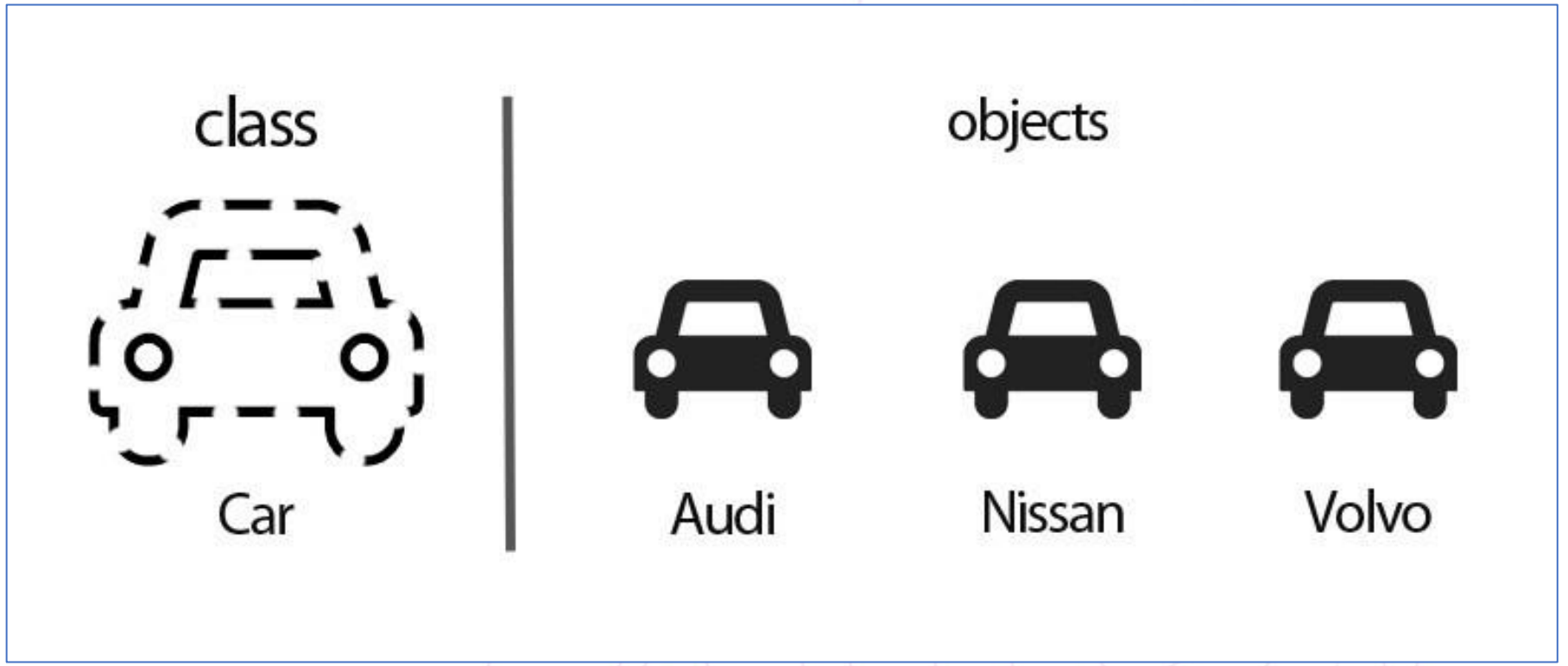
- Marca
- Modelo
- Color
- Cilindraje, etc.

Programación Orientada a Objetos

Acciones (Métodos):

- Arrancar
- Acelerar
- Frenar
- Apagar, etc.

Ejemplo Clase y Objetos



Programación Orientada a Objetos

- El uso de una buena POO facilita enormemente la **modularidad** de un programa informático, permitiendo dividirlo en partes más pequeñas e independientes, así como la detección y depuración de errores, su posterior mantenimiento, y la reutilización del código fuente en otros programas informáticos.

¿Qué es Modularidad?

- Propiedad que permite subdividir una aplicación en partes más pequeñas (**llamadas módulos**), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

- Las funciones como hemos visto, son una forma de dividir las aplicaciones en pequeñas partes. Los módulos se pueden componer de una o muchas funciones.

Lenguajes Populares POO

- Dentro de este tipo de lenguajes de programación, los más conocidos son C++, Java, Python y Javascript.

- Cabe destacar que no todos ellos implementan las mismas características definidas en dicha metodología.

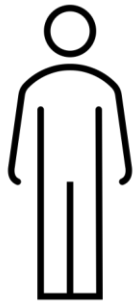
Clases y Objetos

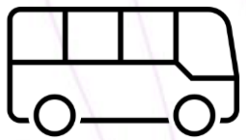
- Podemos entender un **Objeto** como la representación de una **Entidad** de la vida real con la cual podremos interactuar en el programa.

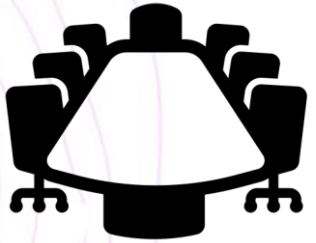
- Antes de poder crear un **Objeto** es necesario crear una definición del mismo, por lo que primeramente deberemos crear una **Clase**, la cual contendrá como **Miembros**.

Entidad

- **Entidad:** Cualquier elemento de la vida real o del mundo del problema que queremos representar en el sistema de información.





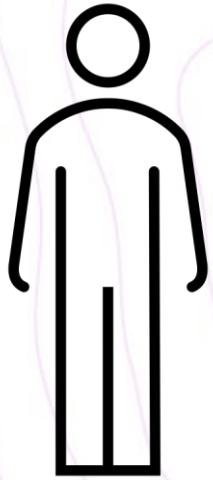


Clase

Clase: Representación del elemento de la vida real o del mundo del problema en el sistema de información. En otras palabras, es una **Plantilla** de dicha representación para construir los objetos.

En general se componen de:

- Atributos
- Métodos (Funcionalidades)



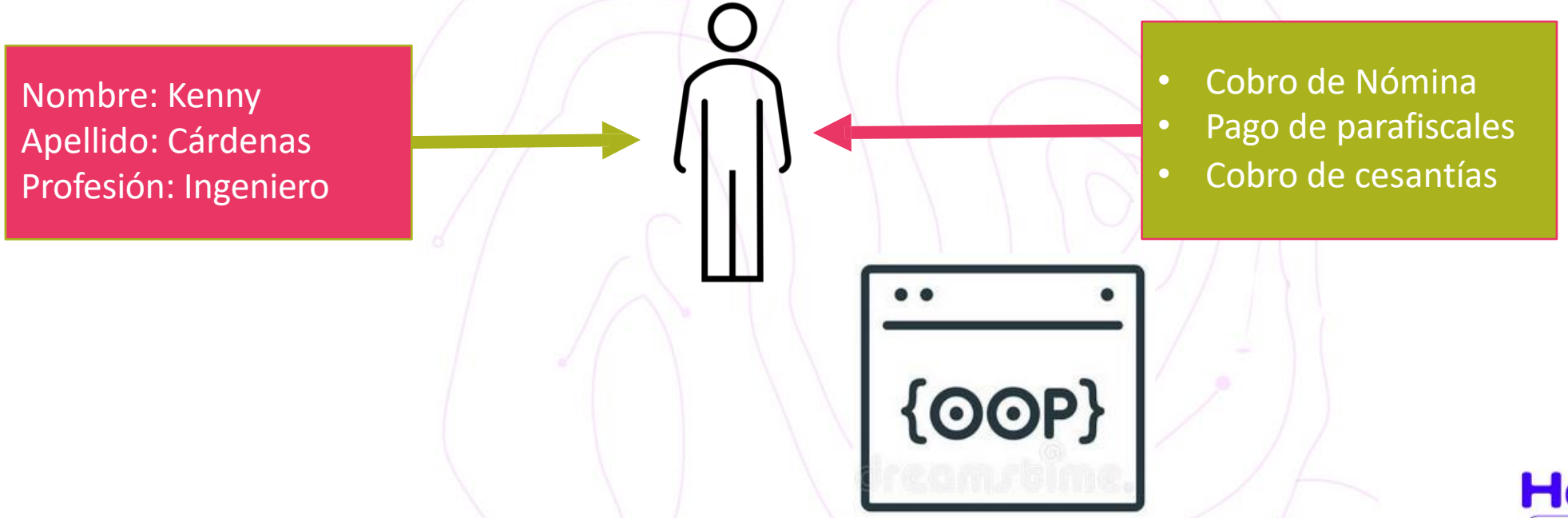
Clase

- **Atributos:** Características de la clase (Ej: Cédula, Nombre, Edad, Cargo, etc…)

- **Funcionalidades (Métodos):** Comportamiento de la clase (Ej: Cobrar Nómina, Solicitar Vacaciones, Realizar Funciones en el Trabajo)

Instancia

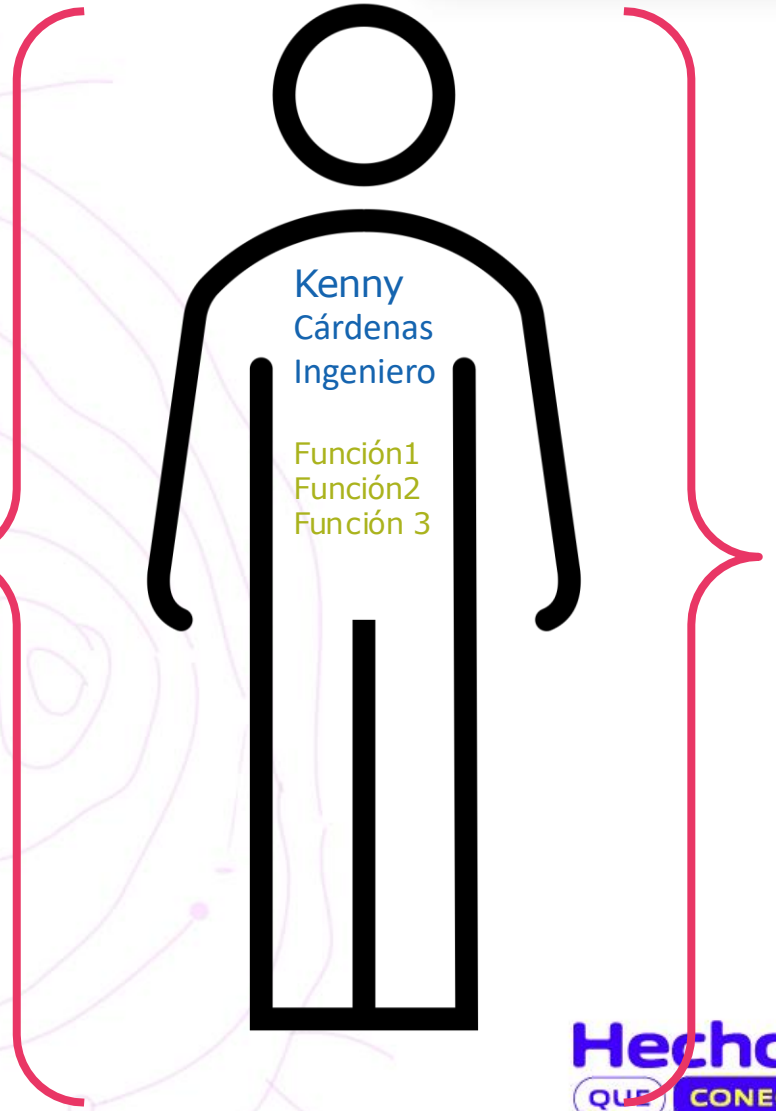
- **Instanciar:** Declarar una **variable** tipo **clase**, convirtiéndose en un **objeto** (tangible en nuestro sistema de información), cuyos atributos y funcionalidades son adquiridas al construirlos en este proceso.



Objeto

- **Objeto:** Tipo Abstracto de Dato que se utiliza para representar las entidades una vez instanciadas.

Empleado



Clase y sus Miembros en POO

- **Propiedades / Atributos:** variables que describen características del Objeto o estados del mismo.

- **Métodos:** se crean de forma parecida a las funciones, y son usados tanto para asignar o devolver el valor de las **Propiedades**, como para describir la forma en que se comporta el Objeto.

Acceso a Miembros de Clase

- Según el caso, no todos los **Miembros** de una **Clase** deben poder ser accesibles desde fuera de ella.

- Para ocultarlos usaremos lo que se conoce como **encapsulamiento**.

Tipos de Encapsulamiento

- **public:** se puede acceder a ellos desde cualquier lugar en el que sea posible acceder a la Clase, y también desde las que hereden de ella.

- **private:** sólo es posible acceder a ellos usando los método proporcionados por la propia Clase (tampoco pueden acceder directamente las clases que hereden de ella).

- **protected:** accesibles desde las clases que hereden de ella, y desde otras que estén en el mismo package o paquete.

Polimorfismo

- Usando un mismo nombre de funciones o métodos de las clases, podemos obtener comportamientos (formas) diferentes, lo que se consigue por medio de la **sobreescritura** y **sobrecarga** de métodos y el uso de **interfaces**.

- A continuación se explica el ejemplo más sencillo de polimorfismo que sucede con el **constructor del objeto**.

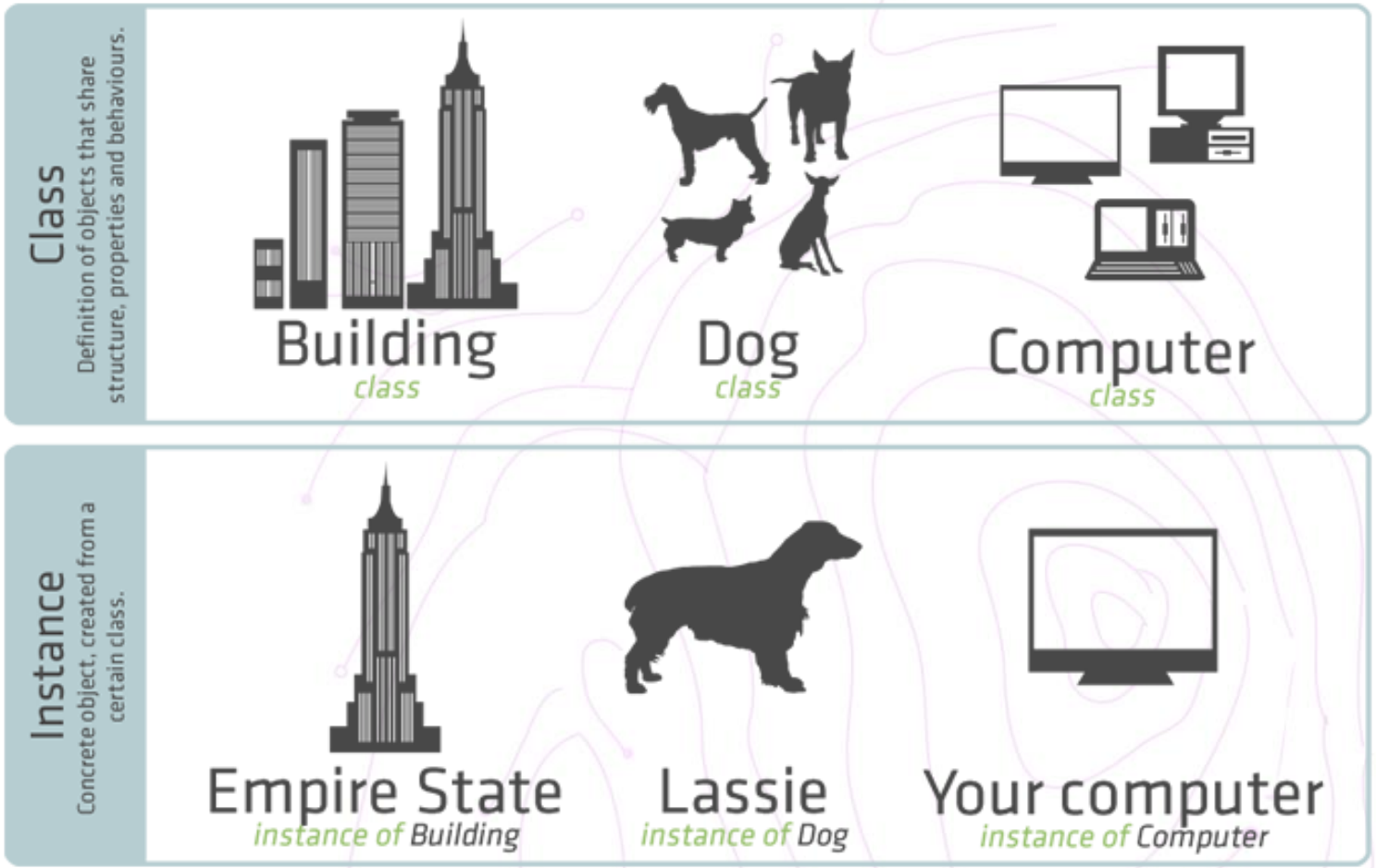
Instanciación y Constructores

- Las clases deben tener un método denominado **constructor**, a partir del cuál se crearán **Instancias** / **Objetos**.

- Es posible que una clase tenga más de un constructor (con el mismo nombre) siempre y cuando tengan parámetros de entrada diferentes.

- Ello se denomina **sobrecarga de métodos** (también es aplicable a sus otros métodos). Y es un caso de polimorfismo.

Ejemplos Instanciación Clases



Programación Orientada a Objetos

Casos de Estudio

Casos de Estudio

- Construcción del Mundo del Problema del clásico Agente Viajero (TSP por sus siglas en inglés *Travelling Salesman Problem*): https://es.wikipedia.org/wiki/Problema_del_viajante

- Requerimiento Sistema de Notas: Contraste solucionando el requerimiento bajo paradigma procedural estructurado y paradigma orientado a objetos.

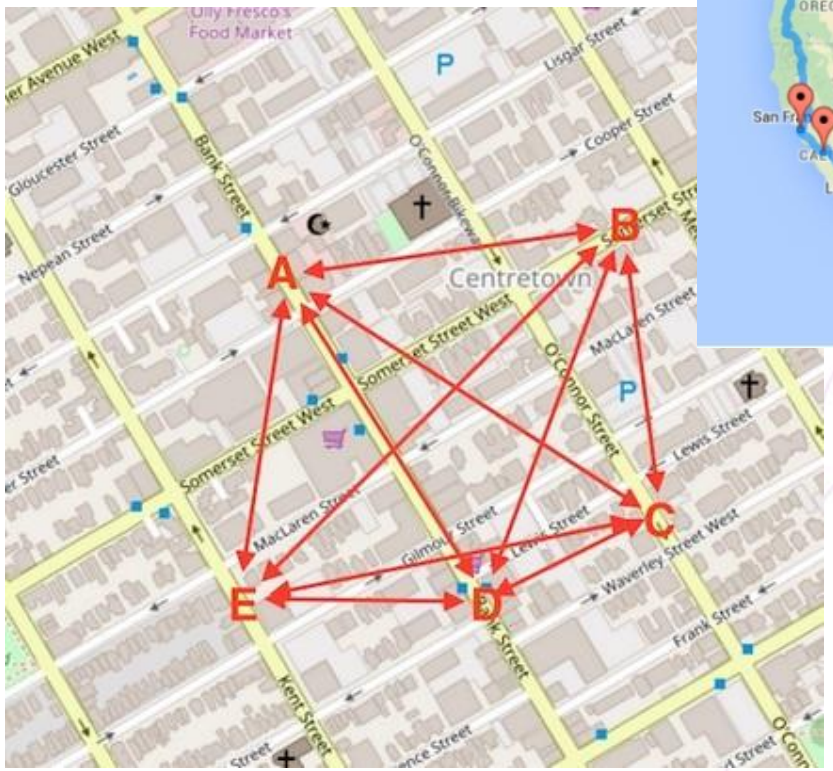
Problema Agente Viajero

- Se requiere una solución de SW para establecer itinerarios de cobertura a red de puntos geográficamente dispersos.

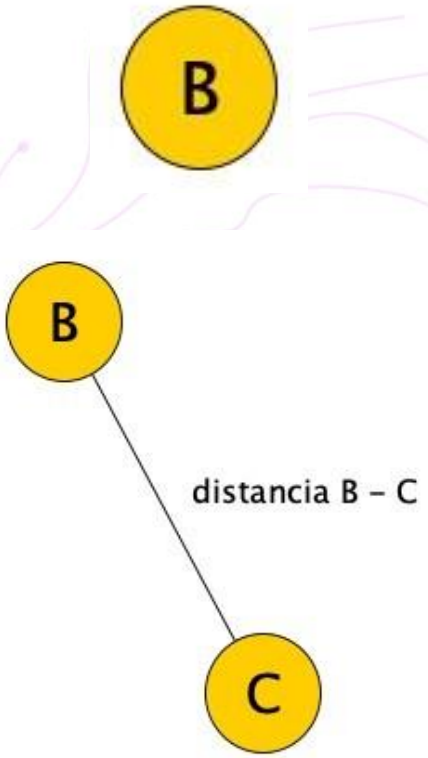
- Requerimientos Funcionales:
  - Cargar red desde archivo de coordenadas
  - Generar itinerarios adecuados
  - Asignar vehículos a los itinerarios

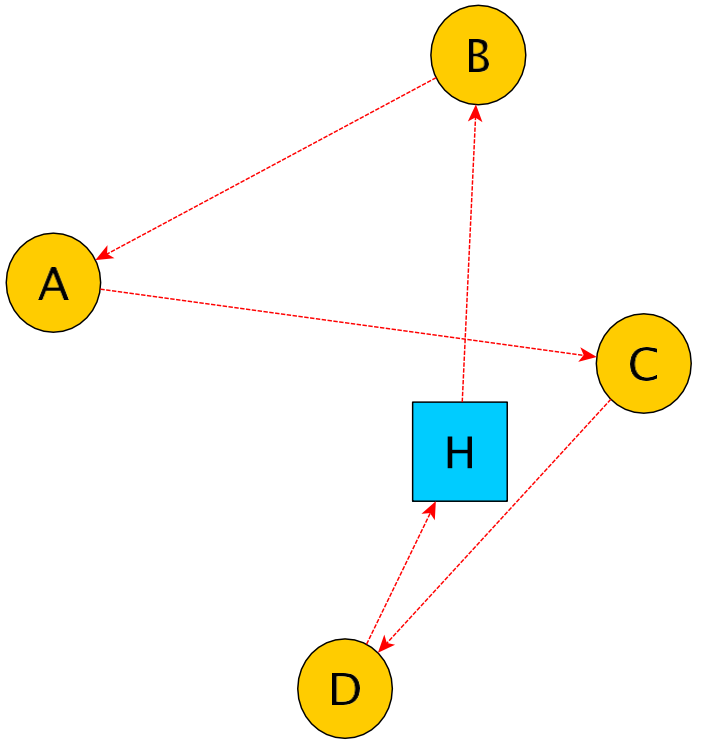
- Mundo del Problema: Abstracción de la red.

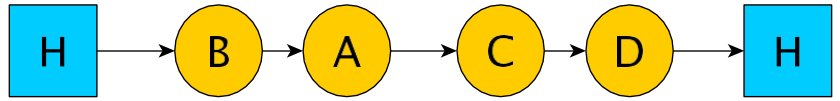
Mundo del Problema



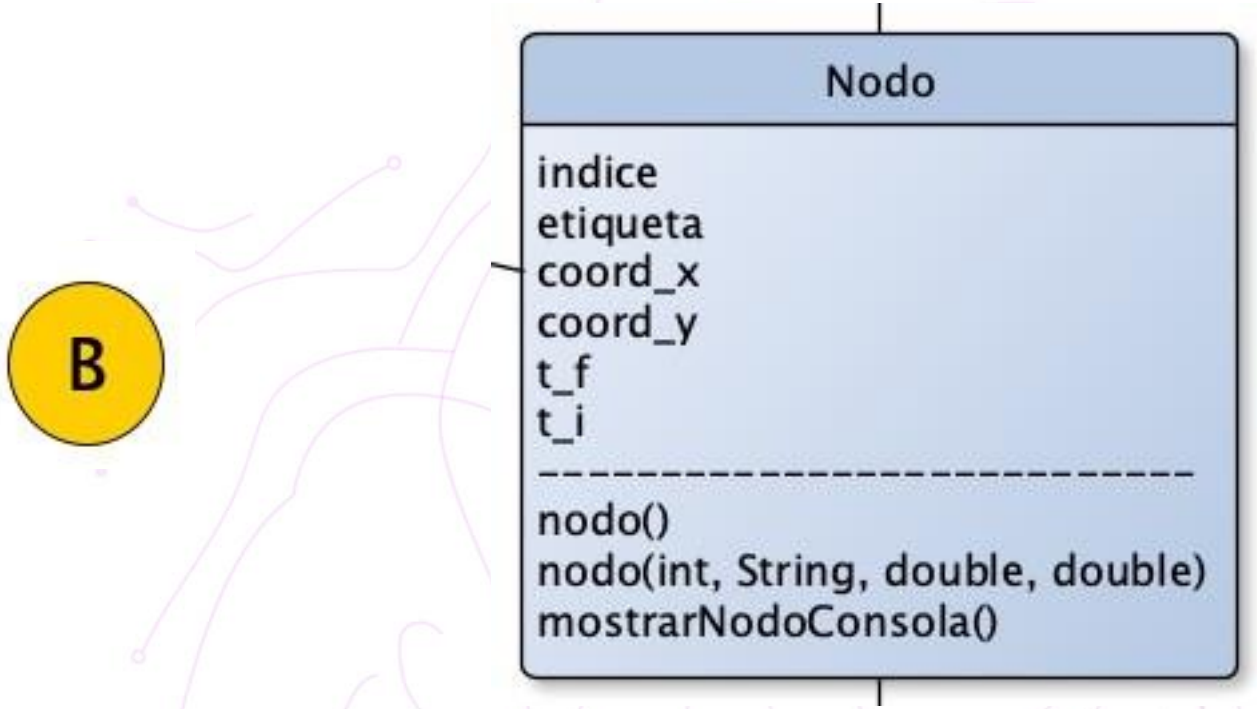
Mundo del Problema



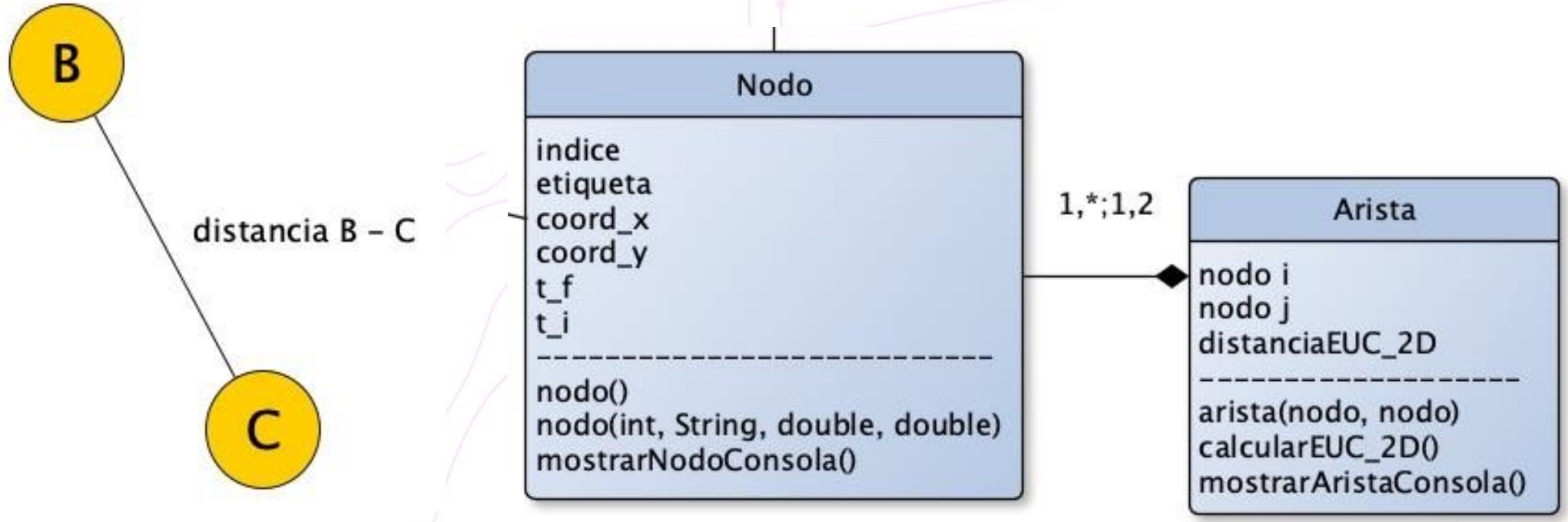




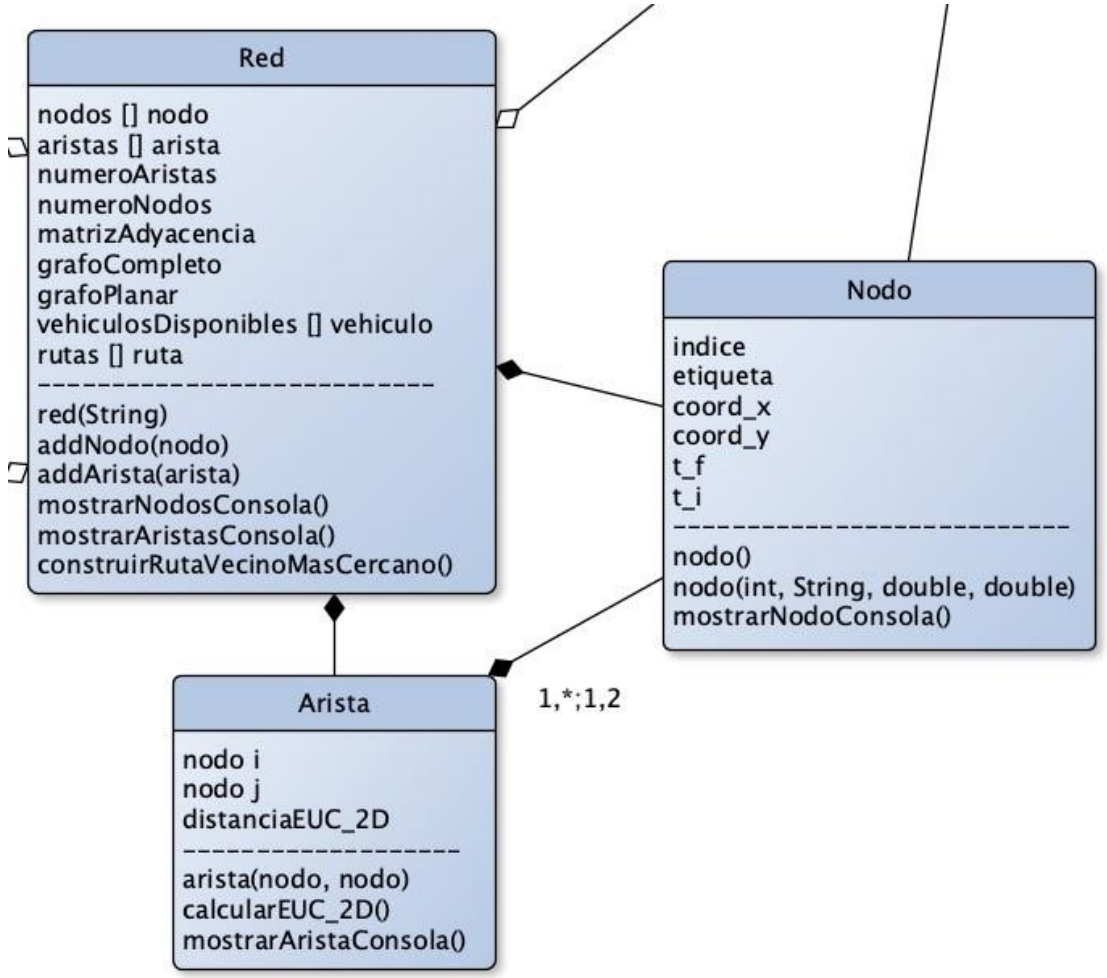
Mundo del Problema

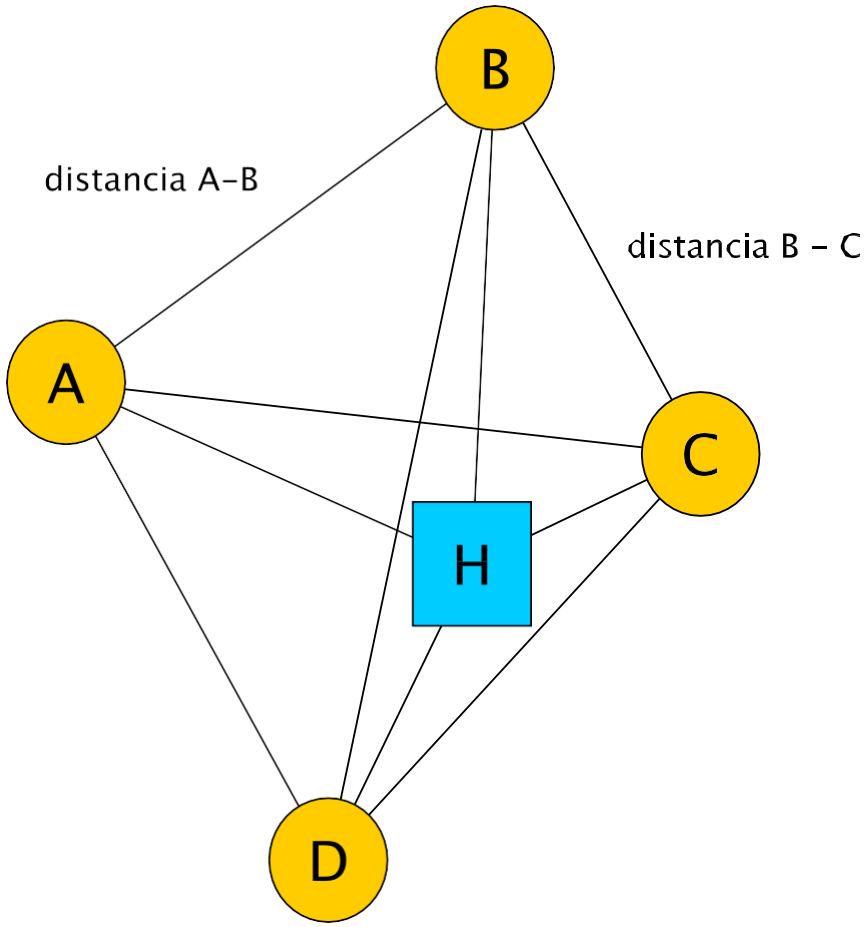


Mundo del Problema

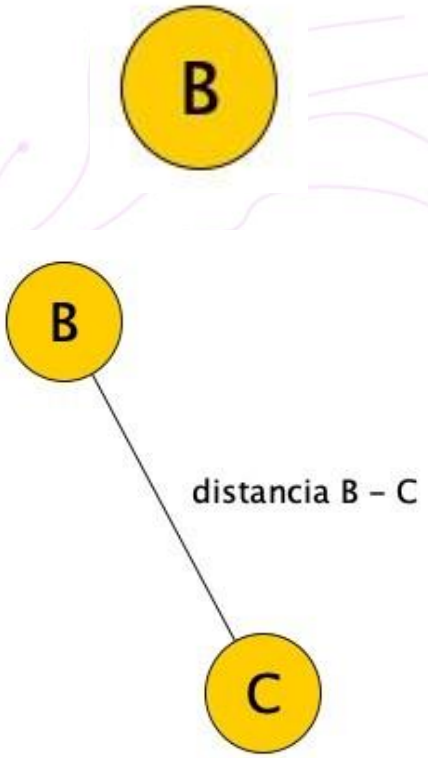


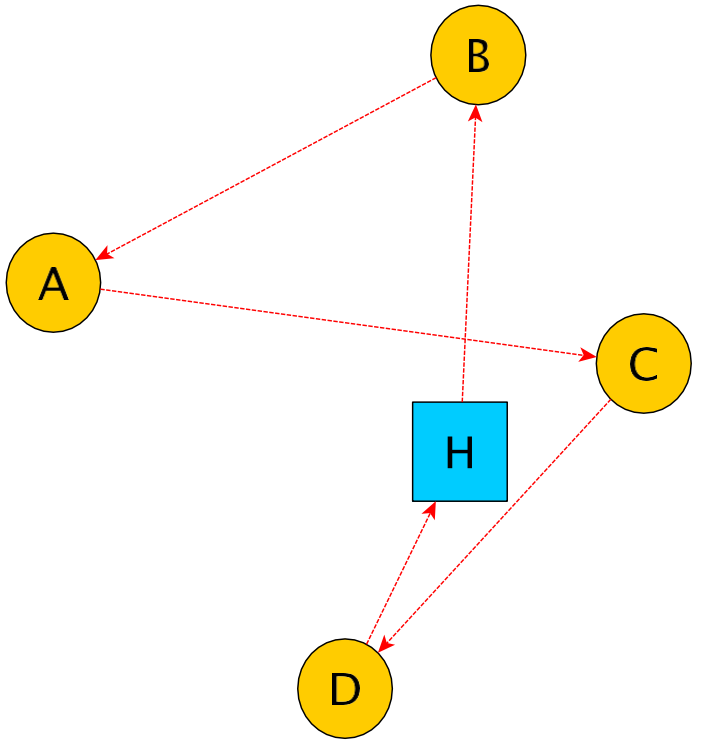
Mundo del Problema

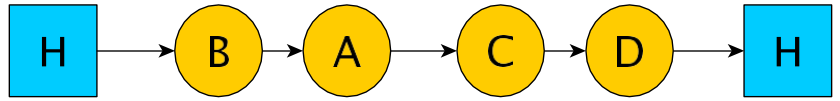




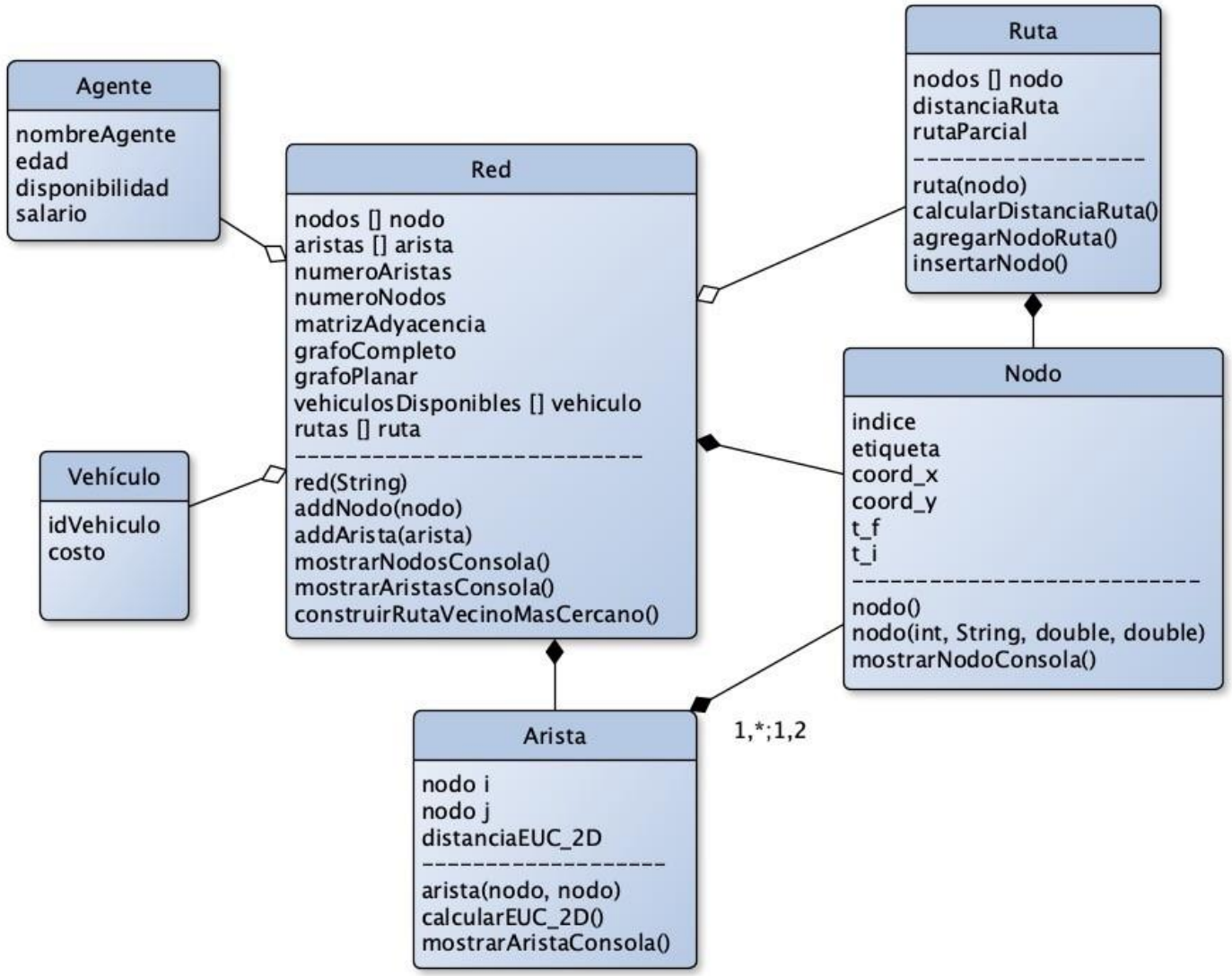
Mundo del Problema







Mundo del Problema



Requerimiento Notas

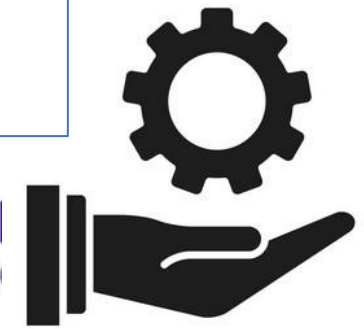
- Un profesor debe calcular el promedio de la nota de quices de sus estudiantes para subirla a la plataforma de notas finales. Sin embargo, el profesor acordó con sus estudiantes que los ayudará eliminando la peor de las 5 notas antes de calcular el promedio que finalmente reportará. Adicionalmente, el profesor se ha dado cuenta qué las notas registradas en su planilla se encuentran en una escala de números enteros de 0 a 100 pero la plataforma está diseñada para recibir el promedio únicamente en la escala estándar de la universidad: de 0 a 5, redondeado a dos decimales.

Especificación Requerimiento

- Escribir una función que reciba como parámetros: una cadena con el código alfanumérico del estudiante y cinco números enteros (nota1, nota2, nota3, nota4, nota5) que representan las notas de los quices del semestre y retorne una cadena de caracteres que le proporciona al profesor la información que desea obtener. La cadena debe tener la siguiente estructura: "El promedio ajustado del estudiante {código} es: {promedio}" dónde, el promedio reportado debe cumplir con las especificaciones mencionadas anteriormente (redondeado a dos decimales, en escala de 0 a 5 y calculado eliminando la peor de las cinco notas del estudiante).

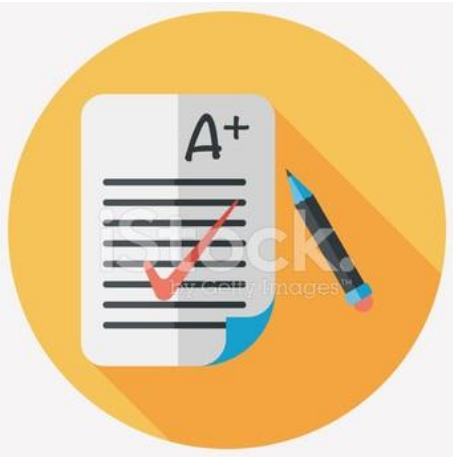
Posible Solución: Programación Estructurada

```python
#Solución paradigma imperativo (procedural/estructurado)
def nota_quices(codigo: str, nota1: int, nota2: int, nota3: int, nota4: int, nota5:int) -> str :
    medicion_maxima_promedio = 5
    max_nota_ = 100
    total_nota_sin_menor = 4
    calcular = 0
    if(nota1 < nota2 and nota1 < nota3 and nota1 < nota4 and nota1 < nota5):
        calcular = (nota2+nota3+nota4+nota5)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    elif(nota2 <= nota1 and nota2 <= nota3 and nota2 <= nota4 and nota2 <= nota5):
        calcular = (nota1+nota3+nota4+nota5)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    elif(nota3 <= nota1 and nota3 <= nota2 and nota3 <= nota4 and nota3 <= nota5):
        calcular = (nota2+nota1+nota4+nota5)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    elif(nota4 <= nota1 and nota4 <= nota2 and nota4 <= nota3 and nota4 <= nota5):
        calcular = (nota2+nota3+nota1+nota5)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    elif(nota5 <= nota1 and nota5 <= nota2 and nota5 <= nota3 and nota5 <= nota4):
        calcular = (nota2+nota3+nota4+nota1)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    calcular = round(calcular, 2)
    return f"El promedio ajustado del estudiante {codigo} es: {calcular}"
```



Mundo del Problema Notas

- El caso de estudio tiene varias entidades relevantes interactuando que pueden integrarse o representarse mejor:



Mundo del Problema Notas