

**Estructura de Dades**

**Pràctica 2: Grafs**

**curs 2022-23**

**Estudiant:** Matías Larrosa

**Professor/a:** Marc Ruiz

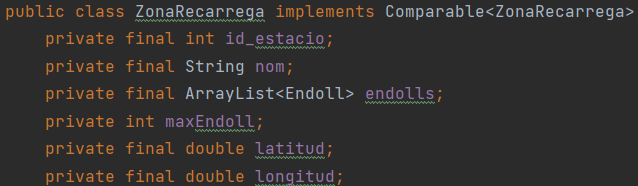
**Data de lliurament:** 29/05/2022

# Decisions de disseny

L’estructura que s’ha implementat per el graf es la millora de la multillista d’adjacència que només guarda una relació entre els dos nodes estalviant així espai.

## ZonaRecarrega

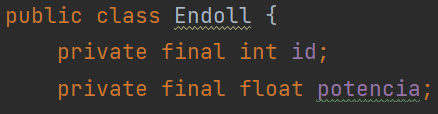
Aquesta es la classe que guarda la informació de cada node.



Guardo la id de cada estació, que es unica per a cadascuna.

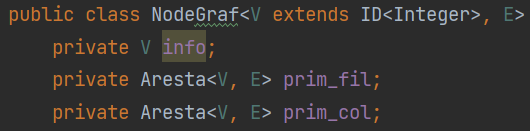
El nom de l’estació els endolls que hi ha en aquella estació, guardant l’index potencia màxima d’aquests.

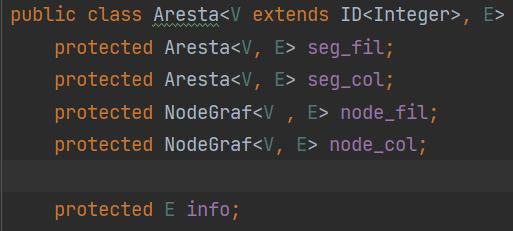
Les coordenades de la Zona de Recarrega latitud, longitud.



Endoll només guarda un identificador del endoll i el valor de la seva potencia.

## NodeGraf i Aresta

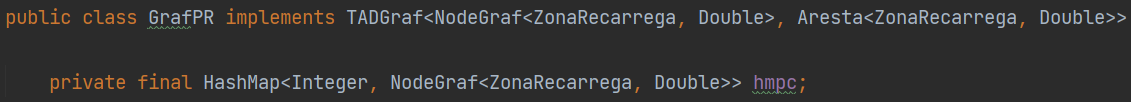




NodeGraf guardarà la informació de les zones de recarrega del graf i Aresta guardarà la informació de la relació entre 2 nodes del graf, una referencia a cadascun d’aquests nodes i a les següents arestes.

## GrafPR

És la classe principal de la practica que guarda la informació extreta de icaen.json.



### crear()

Afegeix totes les zones de recarrega, diferenciades per id\_estacio, al hashMap. I crida a enllacarNodes() que afegirà les arestes necessàries entre els nodes. Al enllaçar nodes si no s’ha pogut enllaçar un node amb cap altre ja que la distancia era superior a 40 km s’enllaçarà amb el que tingui la distancia mínima.

### afegirAresta(NodeGraf, NodeGraf, Aresta)

Aquest mètode afegirà la referencia de la aresta als dos nodes on correspongui (fil o columna) i guardarà una referencia dels nodes a la aresta. Sempre guardaré la referencia de la aresta en les files si el node es més petit o en les columnes si es mes gran.

### existeixAresta(NodeGraf, NodeGraf)

Existeix aresta recorre les files del primer node, si el primer és més petit que el segon, o les columnes, si el primer es més gran que el segon. Buscant en les arestes la referencia a la columna o fila respectivament de el segon graf.

### valorAresta(NodeGraf, NodeGraf)

Valor aresta troba la relació entre els dos nodes i retorna l’aresta si ha trobat la relació o null si no l’ha trobat

### adjacents(NodeGraf)

Retorna un arrayList contenint tots els nodes que tenen relació amb el node.

### camiOptim(int, int, int)

Fent servir A\* a partir d’una id origen (id\_estacio), una id destí i una autonomia recorreré el graf triant per explorar els nodes que siguin més propers al actual i els que més propers estiguin del destí. La meva funció heruristica només pre en compte la distancia entre el node i el destí. Retornaré una arrayList d’un node especial que conté un nodeGraf actual un anterior i un booleà per saber si cal fer una parada per recarregar o no. Per saber quin node toca explorar tinc una estructura minHeap que mantindrà l’element més petit (més proper al node actual i al final) al inici fent l’accés constant.

### zonesDistMaxNoGarantida(int, int)

Fent servir un recorregut en profunditat comprovaré si el node actual es accessible o no.

## Joc de Proves

Per fer el joc de proves tinc dos classes una que fa test sobre els mètodes basics d’un graf i sobre els algoritmes.