# Práctica 2 – Grafs

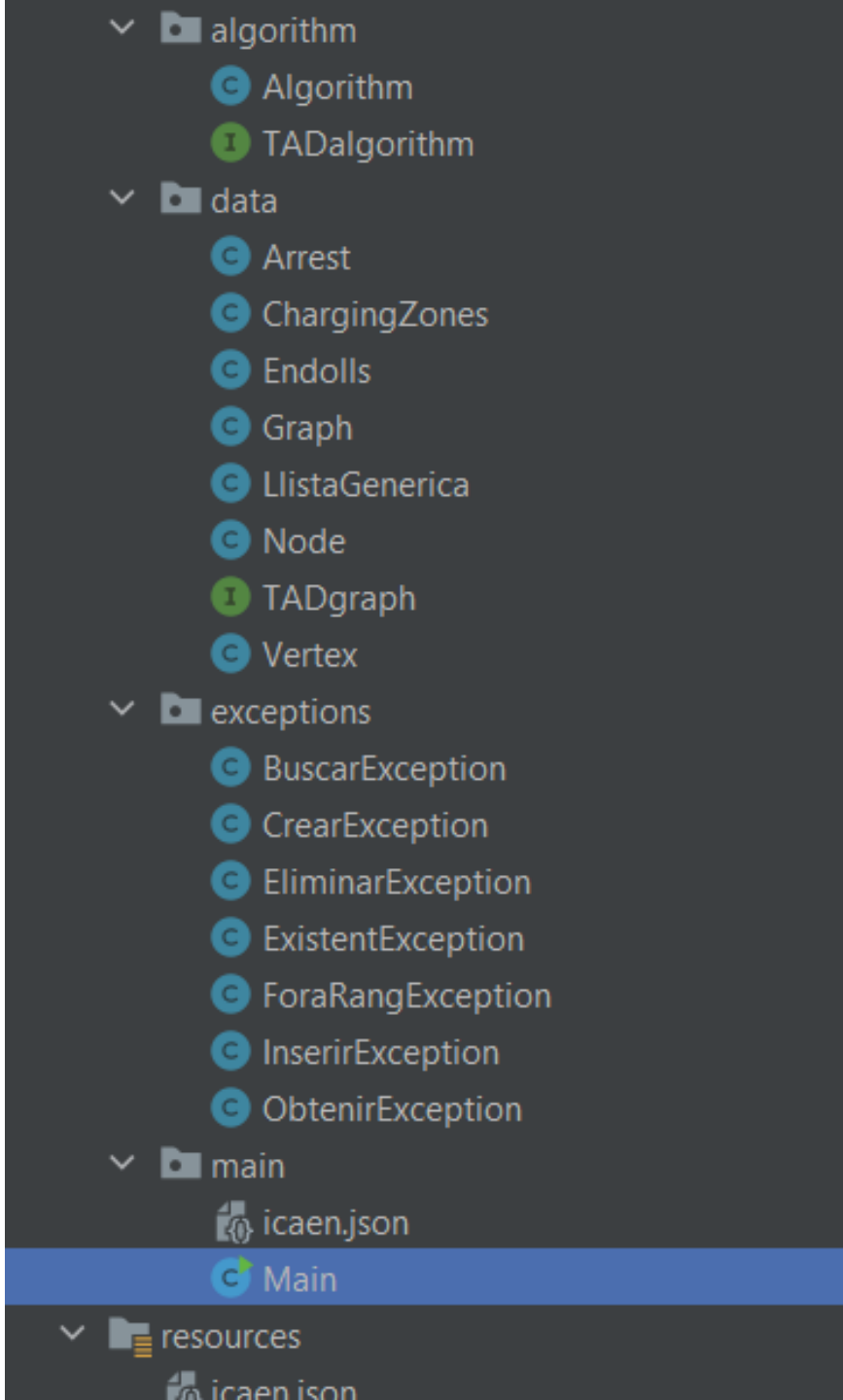
Massin Laaouaj Madrouni

ED - Lab. 3 (Profesora: Cristina Llort)

# DECISIONS DE DISSENY

General:

Per poder implementar tota la estructura que es demanava implementar he separat els diferents apartats en packages, on cadascun del packages engloba els apartats corresponents, com els algorismes, excepcions,...



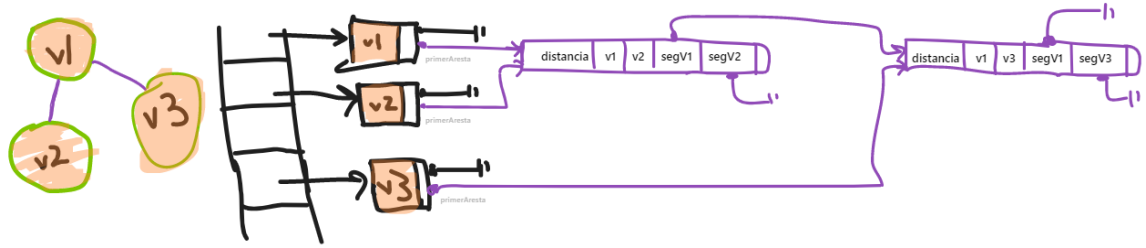
# PLANTEJAMENT

## Estructura i Dades

Per tal d'estructurar les dades que es demanen amb les estructures, s'ha dividit en dos parts.

DADES; Són classes que actuaran com objectes per després emmagatzemar-ho a la nostra estructura.

ESTRUCTURA; S'ha implementat una taula de Hash on emmagatzema tots els vertexs, i un punter de cada vertex a un Node on emmagatzema les connexións entre els dos vertexs.



# PLANTEJAMENT

Arestes

Per les arestes s'ha escollit implementar-ho amb una multillista millorada.

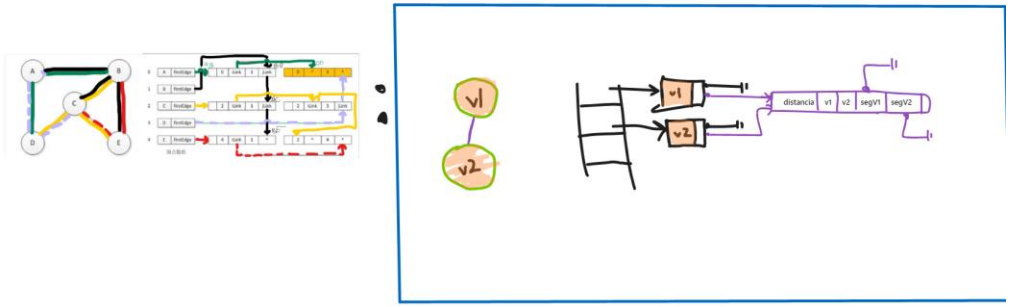
Data; Emmagatzema el valor de l'etiqueta, amb aquest cas les distancies.

iVex; El vertex amb el id més gran

jVex; El vertex amb el id més petit

NextPosA; Següent vertex iVex

NextPosB; Següent vertex jVex




```java
public class Arrest<T> {

    private T data; //distancia
    private Vertex<T> iVex; //iVex--> puntero al vector V1
    private Vertex<T> jVex; //jVex--> puntero al vector V2
    protected Arrest<T> nextPosA; //iLink--> siguiente Aresta del vector V1
    protected Arrest<T> nextPosB; //jLink--> siguiente Aresta del vector V2
```

# ASPECTES MÉS RELLEVANTS – Part 1

Taula de Hash

El primer plantejament, era unificar tot, tindre una classe amb la taula estàtica de Hash ( TaulaHashEl() ), tindré una altre classe on emmagatzemi la key, value/data del hash ( HashEl() ), i finalment tindre la classe de la llista doblement enllaçada per emmagatzemar per a cada posició de la taula de Hash.

```
public class Graph<T, V, E> implements TADgraph<V,E> {
    public HashMap<T, Vertex<V>> vertex= new HashMap<~>(); //Tener almacenado cada Vertice (V:key, Vertex<V> Vertice)
    private Arrest<E> arrest; //Inicializar la lista de Arestas
```

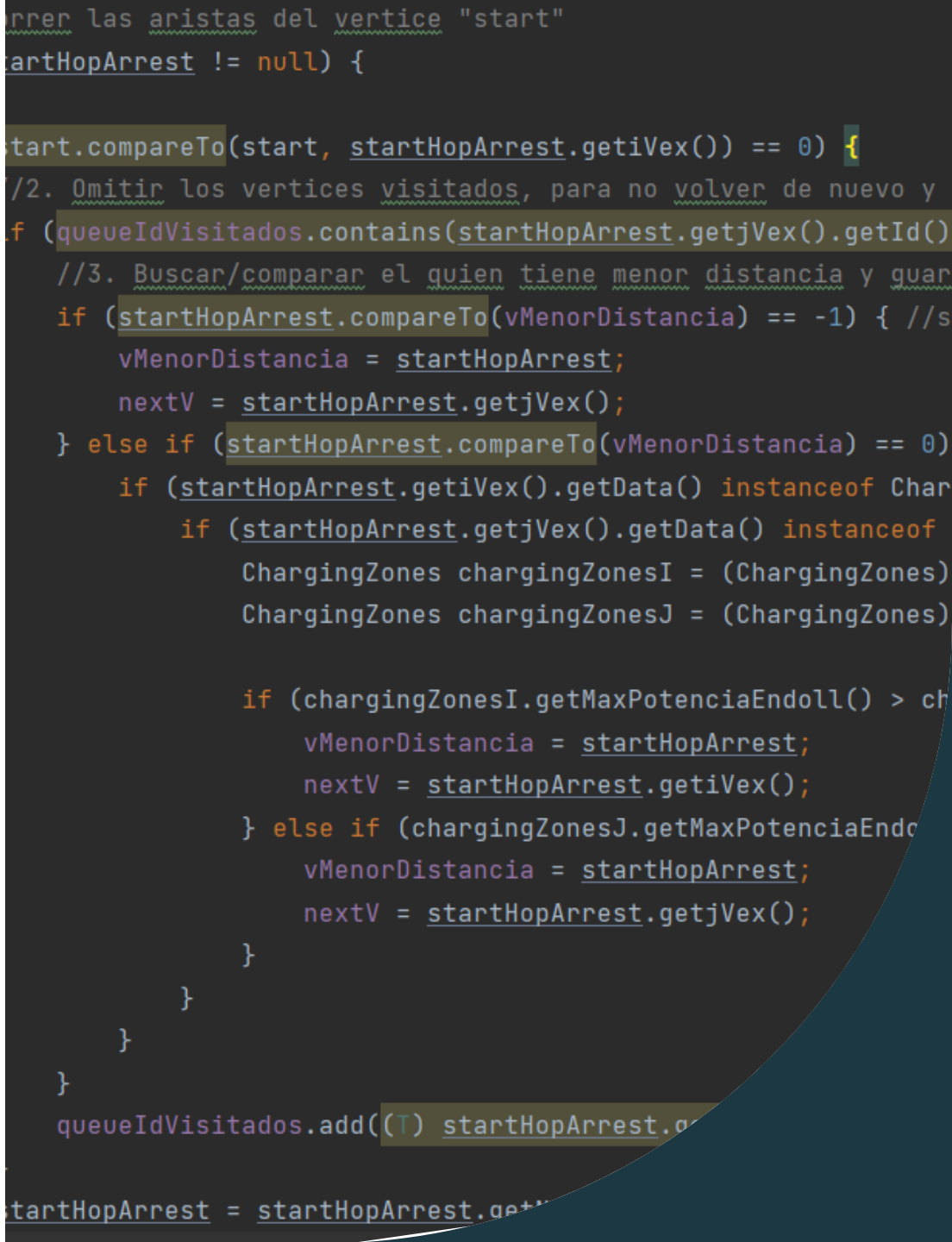
Però després he realitzat una altre plantejament més òptim i adaptat a la segona part de la practica, la de realitzar una llista encadenada per emmagatzemar el value i la key.

# ASPECTES MÉS RELLEVANTS – Part 2

## Recorregut arestes

Per recorre les arestes amb la multillista millorada, es compara el factor més gran, es a dir el vertex amb el id més gran que l'altre que estigui connectat directament amb ell.

Es compara quin es el més, per ubicar-ho, tenim el mètode compareTo, on li pasem els dos vertex a comparar i ens ubica facilement el vertex que ens interessa seguir amb el nextPosA/B.



# JOC DE PROVES

| PROVA | DATOS INTRODUCIDOS | ESPERADO | RESULTADOS |
|---|---|---|---|
| Lectura .json i volcat de dades | Icaen.json | Inserció correcta, vertex sense arestes | Inserció correcta, vertex sense ares tes |
| Unió dels Vertexs amb menos distancia que 40km | Llista vertex | Unions correctes | Unions correctes |
| Buscar ruta optima entre `7088636 i 7088635, autonomia 400` | 7088636 i 7088635, autonomia 400 | Retorna els ids dels vertexs, connectats a menys de 40km | Correcte,<br>retorna els ids dels vertexs, connectats a menys de 40km |
| Buscar element | Buscar per ID | Retorna el objecte | Correcte, retorna el objecte |

```
package algorithm;

import data.*;
import exceptions.*;

import java.util.*;

public class Algorithm<T> implements TADalgorithm{
    Graph graph;

    Map<Vertex, Integer> distances = new HashMap<>();
    Queue<T> queueIdVisitados = new PriorityQueue<T>();
    Queue<Vertex> queueNoArribarAutonomia = new PriorityQueue<Vertex>();
    Arrest vMenorDistancia;
    Vertex nextV= null;

    public Algorithm(Graph graph) {
        this.graph= graph;
    }

    /**
     * @param identificador_origen
     * @param indentificador_desti
     * @param autonomia
     * @return
     */
    @Override
    public LlistaGenerica<String> camiOptim(String identificador_origen, String indentificador_desti, int autonomia) throws CrearException {
        boolean exit= false;
        int id_origen= Integer.parseInt(identificador_origen);
        int id_desti= Integer.parseInt(indentificador_desti);
        LlistaGenerica<String> llIdCamins= new LlistaGenerica<String>();
```

```java
package data;

import com.google.gson.internal.bind.ObjectTypeAdapter;
import exceptions.*;

import java.util.HashMap;

public class Graph<T, V, E> implements TADgraph<V,E> {
    public HashMap<T, Vertex<V>> vertex= new HashMap<T, Vertex<V>>(); //Tener almacenado cada Vertice (V:key, Vertex<V> Vertice)
    private Arrest<E> arrest; //Inicializar la lista de Arestas

    public void unioArestes() throws InserirException {
        float latitudV1, longitudV1, latitudV2, longitudV2;
        float distanciaComprar= 40;

        for (T i: vertex.keySet()) {
            Vertex v1= vertex.get(i);
            ChargingZones v1CZ = (ChargingZones) v1.getData();
            latitudV1= v1CZ.getLatitud();
            longitudV1= v1CZ.getLongitud();

            for (T j : vertex.keySet()) { //Aqui hacer otro for para comparar de este a los otros vertices
                Vertex v2= vertex.get(j);
                ChargingZones v2CZ= (ChargingZones) vertex.get(j).getData();
                latitudV2= v2CZ.getLatitud();
                longitudV2= v2CZ.getLongitud();
                float distancia= calcDist(latitudV1,longitudV1,latitudV2,longitudV2);

                if (calcDist(latitudV1,longitudV1,latitudV2,longitudV2)<distanciaComprar) {
                    //1r--> Comprobar que la conexión no exista entre v1 y v2
                    boolean unionExistente= this.existeixAresta((V)v1,(V)v2);

                    if (unionExistente== false) {
                        //2n--> En caso de no existir creamos una nueva Arista
                        //2.1--> Para tener una referencia comparamos los ids de v1 y v2, quien sea el mas grande sera nextPosA, y el menor sera nextPosB
                        Arrest newArrest= new Arrest(distancia);
                        //3r--> Vinculamos en el v1 la Arista creada en la ultima posición de la lista de Aristas del vector v1
                        this.afegirAresta((V)v1,(V)v2,(E)newArrest);
                    }
                }
            }
        }
    }

    /**
     * @desc check is key exists in the HashMap
     * @param key
     * @return
     */
    public boolean comprovarVertex(T key) {
        boolean keyPresent= false;
        if (this.vertex!= null) {
            keyPresent= this.vertex.containsKey(key);
        }

        return keyPresent;
    }

    public Vertex getVertexExistent(T key) {
        Vertex<V> auxVertex= null;

        if (this.vertex!= null) {
            if (vertex.containsKey(key) == true) {
                auxVertex = this.vertex.get(key);
            }
        }

        return auxVertex;
    }

    public void addNewVertex(T key, Vertex value) {
        this.vertex.put(key,value);
    }

    /**
     * https://www.youtube.com/watch?v=mqfzEZ1CkJc
     * @param latitudV1
     * @param longitudV1
     * @param latitudV2
     * @param longitudV2
     * @return
     */
    public float calcDist(float latitudV1, float longitudV1, float latitudV2, float longitudV2) {
        latitudV1= (float) Math.toRadians(latitudV1);
        longitudV1= (float) Math.toRadians(longitudV1);
        latitudV2= (float) Math.toRadians(latitudV2);
        longitudV2= (float) Math.toRadians(longitudV2);
```

```java
package main;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Properties;
import java.util.Scanner;

import com.google.gson.Gson;
import data.*;
import algorithm.*;
import exceptions.CrearException;
import exceptions.InserirException;

public class Main_Manual<T> {
    static Scanner keyboard = new Scanner(System.in);

    public static void main(String[] args) throws Exception {
        Gson gson = new Gson();
        Graph<Object, ChargingZones,Arrest> graph= new Graph();

        Properties[] data= readJSON(gson);
        saveData(graph,data);

        graph.printa();
        unio(graph);
        menu(graph);
    }

    private static void menu(Graph<Object, ChargingZones,Arrest> graph) throws CrearException {
        String posIni,posFin;
        int autonomia;
        Algorithm algo= new Algorithm(graph);
        LlistaGenerica<String> list;

        System.out.println("\n\nINFO USUARI:\n"
                         + "==========");
        System.out.print("Posició inicial -->  ");
        posIni= keyboard.next();
        System.out.print("Autonomia -->  ");
        autonomia= keyboard.nextInt();
        System.out.print("Destí -->  ");
        posFin= keyboard.next();

        list= algo.camiOptim(posIni,posFin,autonomia);

        String printa= list.print();
        System.out.println(printa);
    }

    private static void unio(Graph graph) throws InserirException {
        graph.unioArestes();
    }

    private static Properties[] readJSON(Gson gson) {
        String content= "";
        try (BufferedReader br= new BufferedReader(new FileReader("src/main/resources/icaen.json"))) {

            String line;
            while ((line= br.readLine()) != null) {
                content+= line;
            }

        } catch (FileNotFoundException ex) {
            System.out.println(ex.getMessage());
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }

        Properties[] data = gson.fromJson(content, Properties[].class);

        return data;
    }

    private static void saveData(Graph graph, Properties[] data) {

        for (int i= 0; i<data.length; i++) {

            /* Save info */
            String id_estacioString= data[i].getProperty("id_estacio");
            int id_estacio= Integer.parseInt(id_estacioString);

            String idString= data[i].getProperty("id");
            int id= Integer.parseInt(idString);

            String potenciaString= data[i].getProperty("potencia");
```

```java
package main;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Properties;
import java.util.Scanner;

import com.google.gson.Gson;
import data.*;
import algorithm.*;
import exceptions.CrearException;
import exceptions.InserirException;

public class Main_Manual<T> {
    static Scanner keyboard = new Scanner(System.in);

    public static void main(String[] args) throws Exception {
        Gson gson = new Gson();
        Graph<Object, ChargingZones,Arrest> graph= new Graph();

        Properties[] data= readJSON(gson);
        saveData(graph,data);

        graph.printa();
        unio(graph);
        menu(graph);
    }

    private static void menu(Graph<Object, ChargingZones,Arrest> graph) throws CrearException {
        String posIni,posFin;
        int autonomia;
        Algorithm algo= new Algorithm(graph);
        LlistaGenerica<String> list;

        System.out.println("\n\nINFO USUARI:\n"
                         + "==========");
        System.out.print("Posició inicial -->  ");
        posIni= keyboard.next();
        System.out.print("Autonomia -->  ");
        autonomia= keyboard.nextInt();
        System.out.print("Destí -->  ");
        posFin= keyboard.next();

        list= algo.camiOptim(posIni,posFin,autonomia);

        String printa= list.print();
        System.out.println(printa);
    }

    private static void unio(Graph graph) throws InserirException {
        graph.unioArestes();
    }

    private static Properties[] readJSON(Gson gson) {
        String content= "";
        try (BufferedReader br= new BufferedReader(new FileReader("src/main/resources/icaen.json"))) {

            String line;
            while ((line= br.readLine()) != null) {
                content+= line;
            }

        } catch (FileNotFoundException ex) {
            System.out.println(ex.getMessage());
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }

        Properties[] data = gson.fromJson(content, Properties[].class);

        return data;
    }

    private static void saveData(Graph graph, Properties[] data) {

        for (int i= 0; i<data.length; i++) {

            /* Save info */
            String id_estacioString= data[i].getProperty("id_estacio");
            int id_estacio= Integer.parseInt(id_estacioString);

            String idString= data[i].getProperty("id");
            int id= Integer.parseInt(idString);

            String potenciaString= data[i].getProperty("potencia");
```

```java
package data;

import java.util.ArrayList;

public class ChargingZones {
    private float latitud;
    private float longitud;
    private int id_estacio;
    private String nom;
    private String carrer;
    private String ciutat;
    public ArrayList<Endolls> llistaEndolls=  new ArrayList<Endolls>();

    public boolean addNewEndoll(Endolls newEndoll) {
        boolean exitNewEndoll= false;
        exitNewEndoll= llistaEndolls.add(newEndoll);

        return exitNewEndoll;
    }

    public boolean comprovarEndoll(Endolls endoll) {
        boolean endollPresent= false;
        endollPresent= llistaEndolls.contains(endoll);

        return endollPresent;
    }


    public ChargingZones() {
        this.latitud= -1;
        this.longitud= -1;
        this.id_estacio= 0;
        this.nom= null;
        this.carrer= null;
        this.ciutat= null;
        //this.llistaEndolls= new ArrayList<Endolls>();
    }

    public ChargingZones(int id_estacio, float latitud, float longitud, String nom, String carrer, String ciutat, Endolls endolls ) {
        this.latitud= latitud;
        this.longitud= longitud;
        this.id_estacio= id_estacio;
        this.nom= nom;
        this.carrer= carrer;
        this.ciutat= ciutat;
        this.llistaEndolls.add(endolls);
    }

    public float getMaxPotenciaEndoll() {
        float maxPotencia= 0;

        for (int i= 0; i< llistaEndolls.size(); i++) {
            if (llistaEndolls.get(i).getPotencia()> maxPotencia) {
                maxPotencia= llistaEndolls.get(i).getPotencia();
            }
        }

        return maxPotencia;
    }

    /* GETTERS AND SETTERS */
    public int getId_estacio() {
        return id_estacio;
    }
    public void setId_estacio(int id_estacio) {
        this.id_estacio = id_estacio;
    }
    public float getLatitud() {
        return latitud;
    }
    public void setLatitud(float latitud) {
        this.latitud = latitud;
    }
    public float getLongitud() {
        return longitud;
    }
    public void setLongitud(float longitud) {
        this.longitud = longitud;
    }
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public String getCarrer() {
        return carrer;
```

```
package data;

public class Endolls {
    private int id;
    private float potencia;
    private String estat;
    private String tipus;

    public Endolls() {
        this.id= 0;
        this.potencia= 0;
        this.estat= null;
        this.tipus= null;
    }

    public Endolls(int id, String estat, float potencia, String tipus) {
        this.id= id;
        this.potencia= potencia;
        this.estat= estat;
        this.tipus= tipus;
    }


    /* GETTERS AND SETTERS */
    public float getPotencia() {
        return potencia;
    }
    public void setPotencia(float potencia) {
        this.potencia = potencia;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getEstat() {
        return estat;
    }
    public void setEstat(String estat) {
        this.estat = estat;
    }
    public String getTipus() {
        return tipus;
    }
    public void setTipus(String tipus) {
        this.tipus = tipus;
    }
}
```