# Projeto de Programação 2

## Parque Eólico

Trabalho realizado por: Diogo de Matos (52710) e Miguel Gordicho (52664).
Engenharia Mecatrónica.

## Introdução

Neste projeto tínhamos como objetivo criar um empreendimento eólico na linguagem de programação JAVA, o empreendimento era composto por um conjunto de aerogeradores em funcionamento, em que cada era caracterizado pelos dados de instalação e detalhes sobre a turbina eólica incluindo altura do eixo até ao solo. Neste empreendimento deve ser possível adicionar, remover ou modificar (fazer a manutenção) de um aerogerador e ainda calcular a potência gerada por um ou mais aerogeradores segundo vários fatores. Para a leitura mais fácil do código e ainda para cumprir número de páginas do relatório foi utilizado JAVADOC.

## Classes, subclasses e métodos

Para criar o empreendimento foi necessário criar algumas classes, sendo estas a classe Atmosfera, a classe Aerogerador, a classe abstrata CP e as suas subclasses e ainda a classe Parque_Eolico. Também se criou uma classe apenas para testar o empreendimento de uma forma mais idêntica ao que se faz nos exercícios do moodle.

- ➔ **Classe Atmosfera (*public class Atmosfera*):** classe responsável por calcular a densidade no ar para uma certa altitude, esta classe é utilizada no método getPt da classe Aerogerador.
  - **Variáveis de instância:**
    **- t_inicial (private static final double):** Temperatura atmosférica padrão ao nível do mar, tendo esta um valor constante de 288,15 K;
    **- l (private static final double):** taxa de gradiente adiabático, tendo esta um valor constante de 0,0065 K/m;
    **- p_inicial (private static final double):** pressão atmosférica padrão ao nível do mar, tendo esta um valor constante de 101325 Pa;
    **- g (private static final double):** aceleração da gravidade ao nível do solo, tendo esta um valor constante de 9,80665 m/$s^2$;
    **- r (private static final double):** constante do gás ideal, tendo este um valor constante de 8,31447 J/($mol \cdot Km$);
    **- m (private static final double):** massa molar do ar seco, tendo esse um valor constante de 0,0289644 Kg/mol.
  - **Métodos da classe:**
    **- getCalcDensidade (double altitude):** método que retorna o valor da densidade do ar segunda uma altitude, este método é utilizado no método getPt da classe Aerogerador.
- ➔ **Classe Aerogerador (*public class Aerogerador*):** classe responsável por caracterizar por assim dizer um aerogerador e ainda calcular a sua potência gerada.
  - **Variáveis de instância:**
    **- tipo_gerador (int):** variável que guarda o tipo de gerador de um aerogerador;
    **- raio (double):** variável que guarda o raio da zona de captação do vento de um aerogerador;
    **- altitude (private double):** variável que guarda a altitude onde foi instalado um aerogerador
    **- alturaEixo (private double):** variável que guarda a altura do eixo de rotação ao solo de um aerogerador;

- **longitude (private double):** variável que guarda a longitude onde foi instalado um aerogerador;
- **latitude (private double):** variável que guarda a latitude onde foi instalado um aerogerador;
- **vel (private double):** variável que guarda o valor da velocidade do vento num certo instante e localização.

- **Constructor:** guarda os valores recebidos no parâmetro sendo estes as características de um aerogerador.
- **Métodos da classe:**
  - **setVelocidade(double vel):** método que guarda o valor recebido da velocidade do vento num instante.
  - **getPt ():** método responsável pelo retorno da potência gerada por um aerogerador, faz o cálculo da área circular da zona das pás para posteriormente utilizar esse valor do cálculo do Pv. Por fim, no método, tempos duas condições. Se o tipo de gerador for igual a 1 e se o raio for menor a 8 metros, irá retornar o produto entre o cálculo do Pv e o valor do CP para o Cenário A que é calculado no método getCPA da subclasse CenarioA. Se o raio for maior ou igual a 8 metros ou o tipo de gerador igual a 2, irá retornar o produto entre o cálculo do Pv e o valor do CP para o Cenário B que é calculado no método getCPB da subclasse CenarioB.
  - **toArray ():** método que retorna as características principais de um aerogerador em forma de um array do tipo double de tamanho seis. (Para mais informações consultar JAVADOC do método).

➔ **Classe CP (*public abstract class CP*):** classe responsável por calcular nos métodos das suas subclasses o valor do coeficiente de potência (CP).

- **Variáveis de instância:**
  - **val_tabeladosVel (public static final double[]):** Variável do tipo array double com os valores tabelados e constantes da velocidade.

➔ **Classe CenarioA (*class CenarioA extends CP*):** Subclasse que é constituída pelo método getCPA.

- **Método da classe:**
  - **getCPA (double vel):** método que retorna o valor do tipo double do coeficiente de potência, utilizado na primeira condição do método getPt da classe Aerogerador. (Para mais informações consultar JAVADOC do método).

➔ **Classe CenarioB (*class CenarioB extends CP*):** Subclasse que é constituída pelo método getCPB.

- **Método da classe:**
  - **getCPB (double vel):** método que retorna o valor do tipo double do coeficiente de potência, utilizado na segunda condição do método getPt da classe Aerogerador. (Para mais informações consultar JAVADOC do método).

➔ **Classe Parque Eólico (*public class Parque_Eolico*):** classe responsável pelo empreendimento eólico, utilizada a biblioteca java.util.ArrayList para a criação do ArrayList. (**Nota:** para todos os métodos desta classe que tenham o parâmetro n, n é o número de aerogerador que se quer retornar, ou seja, n só pode ser a partir de 1 e não 0).

- **Variáveis de instância:**
  - **lista (public static ArrayList<Aerogerador>):** ArrayList para guardar todos os aerogeradores do parque eólico.
  - **colecaoVel (public static ArrayList<Double>):** ArrayList para guardar os valores das velocidades durante um instante/dia.
- **Constructor:** construtor sem parâmetros.

- **Métodos da classe:**
- - **addAerogerador (Aerogerador aerogerador):** método que recebe um aerogerador e adiciona-o ao ArrayList lista. (Para mais informações consultar JAVADOC do método).
  - **getAerogerador (int n):** método responsável por retornar as caracteristicas de um aerogerador em forma de array utilizando o método toArray da class Aerogerador.
  - **removeAerogerador (int n):** método responsável por remover um aerogerador ao ArrayList lista.
  - **getQuantAero ():** método que retorna a quantidade de aerogeradores existentes no parque eólico. (Para mais informações consultar JAVADOC do método).
  - **getQuantAlturaAcima (double altura_ao_solo):** método que retorna a quantidade de aerogeradores que têm a turbina acima de certa altura em relação ao solo. (Para mais informações consultar JAVADOC do método).
  - **manutencaoGerador (int n, int tipo_gerador):** método responsável pela manutenção do gerador. (Para mais informações consultar JAVADOC do método).
  - **manutencaoRotor (int n, double raio):** método responsável pela manutenção do rotor. (Para mais informações consultar JAVADOC do método).
  - **manutencaoAmbos (int n, double raio, int tipo_gerador):** método responsável pela manutenção do gerador e do rotor. Chama os dois últimos métodos falados em cima para fazer a manutenção de ambos.
  - **getCalcPTNumInstante (int n, double vel):** método que retorna o valor da energia/potência produzida por um aerogerador segundo uma velocidade num instante e numa certa localização. (Para mais informações consultar JAVADOC do método).
  - **addVelocidades (double vel):** método responsável por adicionar velocidades ao ArrayList colecaoVel.
  - **clearVelocidades():** método responsável por limpar o ArrayList colecaoVel.
  - **getCalcPTVariosInstantes (int n, double[] colVel):** método que retorna uma coleção dos valores da energia/potência produzida por um aerogerador segundo certas velocidades em vários instantes/dias. (Para mais informações consultar JAVADOC do método).
  - **public double[] getAllCalcPT (double vel):** método que retorna o valor total da energia/potência gerada por todos os aerogeradores existentes no ArrayList lista para um dado instante, ou seja, velocidade do vento. (Para mais informações consultar JAVADOC do método).

# Observações

Neste projeto temos algumas coisas observações a apontar, uma delas foi a dificuldade em fazer algumas exceções nos métodos, outra dificuldade que se teve foi no desenvolvimento do método getCalcPTVariosInstantes, porque inicialmente pensou se em fazer o arraylist colecaoVel como se fosse uma característica de um aerogerador, ou seja, iria se fazer dentro da classe Aerogerador, mas devido a outputs inesperados e sem resolução acabou se por fazer o método clearVelocidades para remover todas as velocidades inseridas quanto a um aerogerador para posteriormente utilizar o arraylist com novos valores de um outro aerogerador, sendo estas as observações menos boas. Também temos algumas observações boas, como o facto deste trabalho ter aumentado a capacidade de resolver problemas no desenvolvimento de código e elevou a capacidade de entender e de escrever código na linguagem JAVA.