Proyecto: Eolo

Autores: José A. Troyano, José C. Riquelme

Última modificación: 04/04/2019

1 Material

Para la realización de esta práctica se dispone de los siguientes elementos contenidos en el fichero zip:

- **/doc/Enunciado.pdf**: fichero PDF con este enunciado
- **/data/**: carpeta de datos
  - **/data/registros_viento.csv**: fichero CSV con datos de registros de viento de varias ciudades andaluzas
- **/src/fp.eolo.test**: paquete Java con las clases de test para las distintas clases que habrá que desarrollar en el proyecto
- **/src/fp.utiles**: paquete Java con utilidades de la asignatura

2 Datos disponibles

En este proyecto trabajaremos sobre datos de registros de viento. En estos datos encontramos solo un tipo de entidad:

- **Registro:** contiene información relativa al viento en una ciudad y día concretos

Los datos están disponibles en formato CSV. En la siguiente figura se muestran las primeras líneas del fichero de datos.

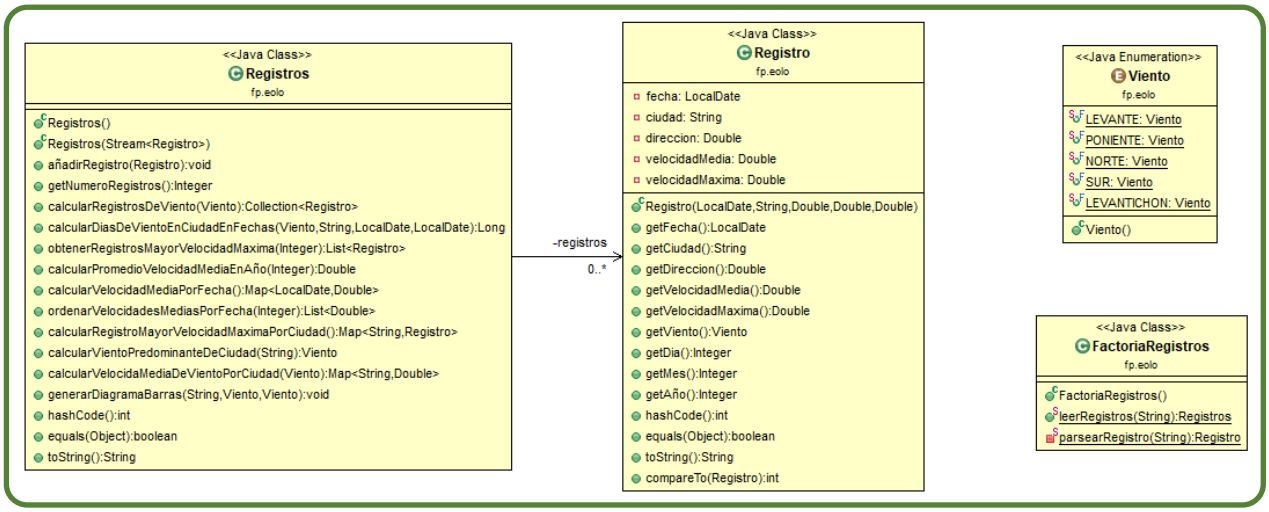
Datos de *registros*

	A	B	C	D	E
1	fecha	ciudad	direccion	velocidad_media	velocidad_maxima
2	2000-01-01	Jaén	190.0	3.96	16.92
3	2000-01-02	Jaén	185.0	5.04	15.12
4	2000-01-03	Jaén	180.0	5.04	16.92
5	2000-01-04	Jaén	170.0	2.16	25.92
6	2000-01-05	Jaén	155.0	6.12	12.96
7	2000-01-06	Jaén	140.0	5.04	21.96
8	2000-01-07	Jaén	150.0	1.08	19.08
9	2000-01-08	Jaén	230.0	2.16	14.04
10	2000-01-09	Jaén	170.0	2.88	14.04
11	2000-01-10	Jaén	140.0	6.84	47.16

3 Modelo

En el siguiente diagrama se muestran todos los elementos que habrá que implementar en este proyecto. Todos ellos se incluirán en el paquete **fp.eolo**. Los aspectos más destacables del modelo son:

- **Registro**: clase para implementar el tipo básico.
- **Registros**: tipo contenedor que incluye, además, algunos métodos de consulta basados en tratamientos secuenciales.
- **FactoriaRegistros**: clase para dar soporte a la creación de objetos **Registro** y **Registros** a partir de datos en un fichero CSV.
- **Viento**: tipo enumerado con los distintos vientos que consideraremos en el análisis.



Este diagrama ha sido generado con el plugin de Eclipse ObjectAid
URL de instalación: http://www.objectaid.com/update/current

4 Ejercicios

EJERCICIO 1

Crear el tipo enumerado **Viento** con los siguientes valores posibles

- **NORTE, SUR, PONIENTE, LEVANTE, LEVANTICHON**

EJERCICIO 2

Crear la clase **Registro** con los siguientes atributos

- **fecha:** atributo *LocalDate* la fecha del registro de viento
- **ciudad:** atributo *String* con la ciudad del registro de viento
- **direccion:** atributo *Double* con la dirección (en grados) del viento registrado
- **velocidadMedia:** atributo *Double* con velocidad media (en km/h) del viento
- **velocidadMaxima:** atributo Double con velocidad máxima (en km/h) del viento. La velocidad media no puede superar a la máxima.

EJERCICIO 3

Crear los siguientes métodos de la clase **Registro** comprobando las restricciones de los atributos en los casos en los que sea necesario

- **Registro:** constructor de la clase a partir de los atributos, en el orden que se indica en el ejercicio anterior
- Métodos *getters*: para todos los atributos de la clase
- **Registro::getDia, Registro::getMes, Registro::getAño:** propiedades derivadas que calculan los números correspondientes a partir de **fecha**
- **Registro::getViento:** propiedad derivada que calcula el tipo de viento a partir de **direccion** y **velocidadMedia**. Los tipos son **SUR** (entre 135° y 225°), **PONIENTE** (entre 225° y 315°), **NORTE** (más de 315° y menos de 45°), **LEVANTE** (entre 45° y 135°) y **LEVATICHON** (misma **direccion** de **LEVANTE** pero con **velocidadMedia** menor de 3km/h)
- **Registro::toString:** mostrando todos los atributos
- **Registro::equals:** usando los atributos **fecha** y **ciudad** determinar la igualdad
- **Registro::hashCode:** usando la misma selección de atributos que el método **equals**
- **Registro::compareTo:** para ordenar de forma natural primero por **ciudad** y luego por **fecha**

EJERCICIO 4

Crear la clase **Registros** con los siguientes atributos y métodos

- **Registros:** atributo con un conjunto de objetos **Registro**

- **Registros:** constructor vacío de la clase **Registros**
- **Registros:** constructor de la clase **Registros** a partir de un *Stream* de **Registro**
- **Registros::añadirRegistro:** método para añadir un **Registro** al conjunto de registros
- **Registros:: getNumeroRegistros:** método para calcular el número de registros del conjunto
- **Registros::toString:** mostrando todos los atributos
- **Registros::equals:** usando el atributo **Registros** para determinar la igualdad
- **Registros::hashCode:** usando la misma selección de atributos que el método **equals**

EJERCICIO 5

Crear la clase **FactoriaRegistros** con los siguientes métodos estáticos

- **FactoriaRegistros::parsearRegistro:** método privado para construir un objeto **Registro** a partir de una línea CSV del fichero de entrada
- **FactoriaRegistros::leerRegistros:** método que devuelve un objeto **Registros** a partir de la ruta del fichero en el que se encuentran los datos de los registros

EJERCICIO 6

Crear la clase **Registros** con los siguientes atributos y métodos

- **Registros::calcularRegistrosDeViento:** filtra el conjunto de registros por un tipo de viento recibido como parámetro.
- **Registros::calcularDiasDeVientoEnCiudadEnFechas:** calcula el número de días de un tipo de viento en una ciudad y entre dos fechas determinadas. La fecha de inicio debe ser anterior que la fecha de fin. Las cuatro informaciones se reciben como parámetros del método.
- **Registros::obtenerRegistrosMayorVelocidadMaxima:** obtiene los **n** registros con mayor velocidad máxima. El número **n** es recibido como parámetro.
- **Registros::calcularPromedioVelocidadMediaEnAño**: calcula el promedio de las velocidades medias en un año recibido como parámetro.
- **Registros::calcularVelocidadMediaPorFecha:** calcula un diccionario cuyas claves son las fechas, y el valor es el promedio de las velocidades medias de todas las ciudades en la fecha correspondiente.
- **Registros::ordenarVelocidadesMediasPorFecha:** devuelve, ordenados por fecha, los valores del diccionario calculado con **Registros::calcularVelocidadMediaPorFecha**. Produce como salida una lista con los **n** primeros valores. El número **n** es recibido como parámetro.
- **Registros::calcularRegistroMayorVelocidadMaximaPorCiudad:** calcula un diccionario cuyas claves son las ciudades, y el valor es el registro con mayor velocidad máxima de la ciudad correspondiente.
- **Registros::calcularVientoPredominanteDeCiudad:** calcula el viento que más se repite en una ciudad recibida como parámetro.
- **Registros::calcularVelocidaMediaDeVientoPorCiudad:** calcula un diccionario cuyas claves son las ciudades, y el valor es el promedio de las velocidades medias del viento (recibido como parámetro) en la ciudad correspondiente.

EJERCICIO 7

Implementar el siguiente método de la clase **Registros** que genera un informe gráfico usando la API de **Google Charts**

- **Registros::generarDiagramaBarras:** genera un diagrama de barras que muestre las medias de velocidad de dos tipos de viento en todas las ciudades.
  El método recibirá los siguientes parámetros:
  - ficheroSalida: un *String* con el nombre del fichero **HTML** en el que se generará la salida
  - viento1, viento2: dos objetos **Viento**, uno para cada uno de los vientos de la comparativa

  El proceso será el siguiente:
  - Construir los diccionarios **velocidadesViento1** y **velocidadesViento2** con el método **Registros::calcularVelocidaMediaDeVientoPorCiudad**
  - Construir el diccionario **medias** con las mismas claves que **velocidadesViento1** y **velocidadesViento2**, y con las medias de sus valores
  - Construir la lista de *String* **ciudadesOrdenadas** con las claves del diccionario **medias** ordenadas de menor a mayor por el valor del diccionario
  - Generar la salida **HTML** en un fichero de la carpeta **/out** con la siguiente instrucción:

```
GraphTools.barChart("out/"+ficheroSalida, ciudadesOrdenadas,
                    viento1.toString(), velocidadesViento1,
                    viento2.toString(), velocidadesViento2);
```