Objetivos:
- Poner en práctica todos los conceptos vistos en la asignatura

1. Descripción de la práctica

En los últimos años la práctica deportiva ha ido creciendo en varios sectores. En especial, la afición de pruebas de resistencia ha aumentado notablemente en los últimos 10-15 años. Hoy en día es bastante normal conocer gente que son aficionados a correr maratones, ultramaratones o alguna prueba similar de mucho fondo. De forma similar, debido a la pandemia la práctica de ciclismo ha crecido significativamente haciendo que falten bicis en el estocaje mundial.

Por otro lado, el uso de redes sociales se ha normalizado en la sociedad, y en combinación al factor de la afición deportiva redes sociales especializadas como STRAVA[1] hace que sean una de las aplicaciones más usadas por corredores y ciclistas de toda condición. Esto implica que el uso de relojes y dispositivos GPS sean usados por la mayoría y tengamos acceso a una ingente cantidad de datos.

En este proyecto procesaremos los datos recopilados con dispositivos GPS utilizado por ciclistas y generar informes de sus actividades describiendo el rendimiento alcanzado en su actividad. Para ello, se proporciona un proyecto que contiene la definición de los tipos de datos y funcionalidades básicas. En esta práctica se debe completar el proyecto implementando la funcionalidad indicada en el apartado 3.

2. Proyecto proporcionado

Para facilitar la realización de la práctica se proporciona un proyecto en el que deberéis llevar a cabo las tareas descritas en la Sección 3. A continuación, se describen los tipos de datos definidos y los subprogramas proporcionados en el proyecto.

2.1. Ficheros de datos de las actividades

El proyecto proporcionado incluye una carpeta llamada *TrackFiles* que contiene varias carpetas, cada uno de los cuales contiene los ficheros que contienen los tracks de cada actividad. Los nombres de los ficheros están estandarizados y siguen un patrón simple

actividad-*identificador*-*num*.csv

donde:

1. **identificador**: código alfanúmerico que identifica al ciclista (p.e. “Athlete2”).

[1] `https://www.strava.com/`

2. **num**: número de dos dígitos que identifica cada actividad del ciclista (p.e. "05").

De esta manera la actividad 01 del atleta *Athlete1* se guardará en el siguiente fichero:

TrackFiles/Athlete1/activity-Athlete1-01.csv

En total *TrackFiles* contiene actividades de 7 atletas ([1-7]), donde para cada atleta tenemos 5 actividades ([01-05]).

El contenido de los ficheros tiene este formato:

```
Duracion, Longitud, Latitud, Elevacion, PC
0, 13.908066535368562, 44.80675270780921, 28.0, 107
1, 13.90806695446372, 44.80675270780921, 28.0, 108
2, 13.908067038282752, 44.80675279162824, 28.0, 109
3, 13.90806645154953, 44.806752959266305, 28.0, 110
4, 13.908065780997276, 44.80675337836146, 28.0, 112
...
```

Cada fila contiene los datos del *trackpoint* recopilados con cierta frecuencia por el dispositivo GPS. Cada trackpoint contiene la siguiente información:

- *Duración*: Duración de la actividad en segundos hasta ese momento.
- *Longitud*: La posición geográfica en la longitud.
- *Latitud*: La posición geográfica en la latitud.
- *Elevación*: Elevación en metros del ciclista sobre el nivel del mar.
- *PC*: Frecuencia cardíaca del corredor en el momento del muestreo.

2.2. Tipos de datos

Para la representación de los datos se ha definido el tipo de dato *InfoLogTrack* que se muestra a continuación.

Listado 1: Implementación del tipo de datos `InfoLogTrack`

```
public class InfoLogTrack {
  // Informacion temporal en segundos
  public int[] tiempo;
  // Informacion posicional
  public double[] longitud;
  public double[] latitud;
```

```
  // Otra informacion
  public double[] altitud;
  public int[] frecCardiaca;
}
```

Para almacenar las estadísticas obtenidas de los datos de la actividad se han definido dos tipos de dato. El primero almacenará la información básica de la actividad en el tipo *EstadisticasBasicas*. El segundo, se utilizará para almacenar la información más avanzada, será del tipo *EstadisticasAvanzadas*.

Listado 2: Implementación del tipo de datos `EstadisticasBasicas`

```
public class EstadisticasBasicas {
  public int duracion; // segundos
  public double distancia; // km
  public double velocidad; // km/h
  public double fCMedia;  // pulsos/minuto
}
```

Listado 3: Implementación del tipo de datos `EstadisticasAvanzadas`

```
public class EstadisticasAvanzadas {
  public double calorias;
  public double velocidadMaxima;
  public double velocidadMaxKm;
  public double desnivel;
}
```

2.3. Subprogramas proporcionados

Para facilitar algunas de las operaciones requeridas, el proyecto proporciona los subprogramas que se describen a continuación y que están definidos en el fichero *FuncionalidadAuxiliar.java*.

2.3.1. Cargar la información de una actividad

La función *cargarInfoCSV* devuelve una estructura de datos que contiene los datos de la actividad almacenados en el fichero indicado.

Listado 4: Función `cargarInfoCSV`

```
/**
 * Procesa un fichero que contiene el log de una actividad y devuelve
     una estructura
 * con toda la informacion del track
 * @param pFich el fichero que contiene el log de la actividad
```

```
 * @return la estructura que contiene toda la informacion del track
 */
public static InfoLogTrack cargarInfoCSV(String pFich) {}
```

2.3.2. Graficar la información procesada

El subprograma *generarTrackPlot*, que dados 3 vectores que representan los datos recopilados (de las distancias recorridas, las frecuencias cardiacas y las elevaciones, respectivamente) en distintos momentos de la actividad realizada, genera la gráfica correspondiente. La gráfica se almacenan en el fichero indicado.

Listado 5: Función `generarPlotTrack`

```
/**
 * Genera la grafica con los datos de la actividad especificada.
 * @param pDist array que contiene las distancias recorrida cada
    instante
 * @param pFC array que contiene la frecuencia cardiaca de cada
    instante
 * @param pElev array que contiene la altura de cada instante
 * @param pPlotFileName el nombre del fichero en el que se quiere
    guardar la
 * imagen
 * @param pShowPlot valor booleano que indica si se quiere mostrar
    la grafica
 * o no
 **/
public static void generarTrackPlot(double[] pDist, double[] pPulso
   , double[] pElev, String pPlotFileName, boolean pShowPlot) {}
```

2.3.3. Obtener el peso del ciclista

La función *obtPesoAtleta* devuelve el peso del o la ciclista indicado mediante su identificador (p.e. "Athlete2"). En el caso que el atleta indicado no exista devolverá el peso de 68, 0kg como valor por defecto.

Listado 6: Función `obtPesoAtleta`

```
/**
 * Devuelve el peso corporal del atleta indicado
 * @param pId
 *            identificador del atleta ("e.g. Athlete2")
 * @return el peso corporal del atleta
 **/
public static double obtPesoAtleta(String pId) {
```

3. Tareas

En este proyecto se deben **diseñar e implementar** los subprogramas que se describen a continuación y que están definidos en el fichero *AnalisisLogTrack.java* del proyecto proporcionado.

3.1. Estadísticas de la actividad

La persona que practica el ciclismo suele fijarse en ciertas informaciones que resumen la actividad. Las estadásticas básicas de la actividad reflejan la siguiente información:

- Duración de la actividad en segundos.
- Distancia total en kilometros.
- Velocidad media (km/h) de la actividad realizada.
- Frecuencia cardíaca media.

La función *obtEstadisticasBasicas*, dados los datos de la actividad guardados en el track, devuelve las estadísticas correspondientes a la actividad.

3.1.1. Distancia recorrida entre dos puntos

Para obtener las estadísticas básicas necesitamos calcular la distancia recorrida entre dos *trackpoints*. Teniendo en cuenta que los puntos se representan mediante las coordenadas de latitud y longitud, y asumiendo que las tierra es una esfera perfecta, podemos obtener la distancia (d) en kilómetros aplicando la formula de Haversine.

$$\begin{aligned} \Delta Lat &= \Phi_2 - \Phi_1 \\ \Delta Long &= \lambda_2 - \lambda_1 \\ a &= \sin^2(\frac{\Delta Lat}{2}) + \cos(\Phi_1)\cos(\Phi_2)\sin^2(\frac{\Delta Long}{2}) \\ d &= 2R\arctan 2(\sqrt{a}, \sqrt{1-a}) \end{aligned} \tag{1}$$

donde Φ_1 y Φ_2 son las latitudes de los dos puntos, λ_1 y λ_2 son las longitudes de los dos puntos, y R el radio de la tierra que equivale a 6371 kms. La formula asume que las longitudes y latitudes están en radianes.

`Math` de java (ver documentación) ofrece varias funcionalidades trigonométricas útiles para implementar la función que calcula la distancia entre dos puntos:

- Podemos utilizar `Math.toRadians` para transformar los grados en radianes.
- Las funciones `Math.sin`, `Math.cos`, `Math.atan2` sirven para calcular sin, cos y $arctan2$, respectivamente.

3.2. Perfil del recorrido

Un dato importante para el analisis de la actividad del corredor es medir la correlación que hay entre la elevación y la frecuencia cardíaca. Esto nos permite entender mejor como se ha llevado a cabo la actividad y determinar de forma cualitativa la dureza del entrenamiento. En este sentido, es interesante obtener el gráfico que muestre el perfil del recorrido realizado por el corredor (*distancia x elevación*), junto a la evolución de la frecuencia cardíaca (*distancia x frecuencia-cardíaca*) durante el recorrido. Esto nos permitirá analizar como varía la frecuencia cardíaca según cambia la pendiente de la carretera.

Hay varias formas de **calcular la elevación** en un rango de distancia o tiempo. Asumiendo que la el lapso de tiempo entre dos muestreos del GPS es pequeña (alrededor de un segundo), la forma más simple de representar la elevación (o altura) es utilizar directamente la información guardada por el GPS. Recordad que la información de cada *trackpoint* esta representada en cada fila del fichero CSV y cargada en el tipo de dato *InfoLogTrack*.

De forma similar, para **calcular la evolución del la frecuencia cardíaca** durante el recorrido podemos utilizar directamente las lecturas obtenidas por el GPS y almacenadas en cada *trackpoint*.

La función *graficarPerfil*, dados los datos de la actividad, generará el gráfico que muestra el perfil del recorrido junto con la evolución de la frecuencia cardíaca y lo guardará en *GeneratedPlots* siguiendo la siguiente forma:

plot-*identificador*-*num*-perfil.png

donde:

1. **identificador**: Código alfanumérico que identifica al ciclista ("*Athetle2*").
2. **num**: Número de dos dígitos que identifica cada actividad del ciclista.

La figura 1 muestra el ejemplo del perfil de un recorrido (la linea en rojo) junto con la variación de la frecuencia cardíaca (la linea en azul) durante la actividad.

3.3. Análisis de la frecuencia cardíaca

Otra forma interesante para analizar el esfuerzo es extraer los rangos de los ritmos cardiacos acumulados durante la actividad. De esta forma podemos saber cómo se ha distribuido el esfuerzo según la frecuencia cardíaca. Los esfuerzos se categorizan en zonas dependiendo de los rangos de la frecuencia cardíaca. Normalmente, cada persona suele tener asociado unos rangos distintos. En cambio, para completar la práctica utilizaremos los mismos rangos predefinidos para todos los atletas, que son los 5 que se muestran en la Tabla 1.

La función *obtDistribucionFC*, dados los datos de la actividad, devolverá el número de *trackpoints* obtenidos para cada rango de la frecuencia cardíaca.

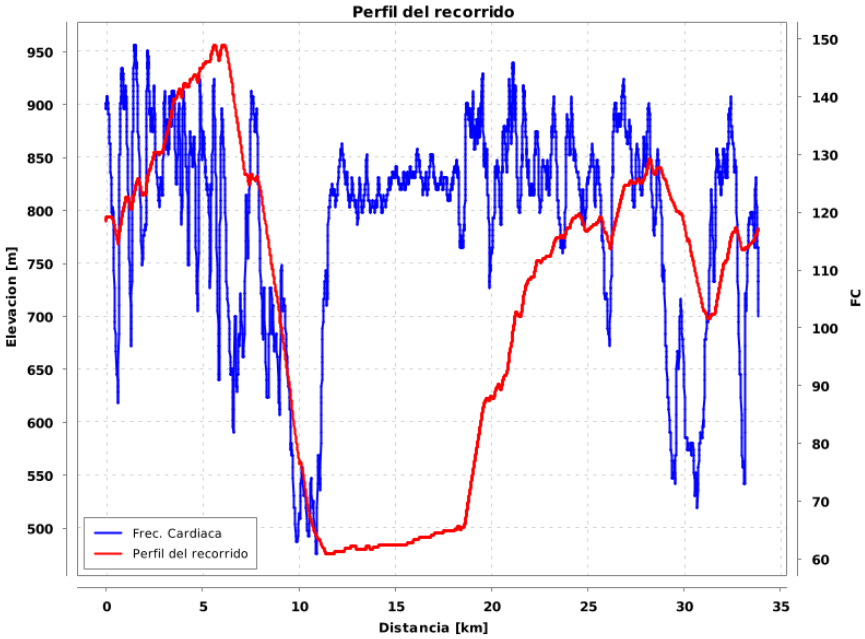

Figura 1: Ejemplo del gráfico que muestra el perfil del recorrido junto a la evolución de la frecuencia cardíaca.

Zona	Descripción	rango
1	Resistencia	<123
2	Moderado	[123, 153)
3	Ritmo	[153, 169)
4	Umbral	[169, 184)
5	Anaeróbico	>184

Tabla 1: Rangos predefinidos de las zonas de esfuerzo

3.4. Consumo energético

Dependiendo de la intensidad y la duración de la actividad el consumo energético suele ser distinto. A mayor duración del ejercicio mayor suele ser el gasto energético, tal como pasa con la intensidad. Cuando mayor la intensidad, mayor es el consumo energético. En nuestro caso el consumo energético lo calcularemos en calorías.

Uno de los métodos más fáciles para registrar la intensidad de una actividad física es el método *Metabolic Equivalent Task* (MET). El consumo de energía en distintas actividades se determina monitorizando el consumo de oxígeno durante la actividad, y así poder determinar un consumo promedio de oxígeno por unidad de tiempo y compararlos con el consumo de oxígeno en reposo. Una unidad de MET es la energía gastada en reposo, dos MET indican que la energía gastada es el doble que en reposo, tres MET es el triple del gasto energético

en reposo, etc.

Para la estimación del MET de la actividad ciclista se puede usar las siguiente formula:

$$MET = -1{,}52 + 0{,}510 * km/h \tag{2}$$

Para el cálculo del gasto calórico nos podemos basar en la relación del peso (en kilogramos) del ciclista y el MET generado en una duración de tiempo entre dos muestras consecutivas (Δt, expresado en minutos), como se muestra en la formula 3

$$calorias_{\Delta t} = \Delta t * (3{,}5 * MET * kg)/200 \tag{3}$$

Como el valor de MET varía en cada instante, una opción interesante del problema de cálculo es plantearlo como un problema de integración. Para ello, debemos integrar (esto es, sumar) la función que determina los consumos calóricos que se realiza entre dos mediciones. Como podemos observar en la fórmula 4 el consumo viene dado por el MET generado entre dos momentos del muestreo del reloj GPS. Tened en cuenta que la estimación del valor de MET depende de la velocidad, y que la velocidad se estima entre dos muestras (posicion más tiempo) del reloj GPS.

$$totalCalorias = \sum_{i=1}^{n} calorias_{\Delta t_i} = minutos_{\{t_{i-1},t_i\}} * (3{,}5 * MET * kg)/200 \tag{4}$$

La función *estimarConsumoCalorias*, dados el peso corporal del ciclista y los datos de la actividad, estima el consumo de calorías realizado durante la actividad.

3.5. Estadísticas avanzadas

En muchas ocasiones el corredor suele estar interesado en obtener un resumen de su actividad que complemente la información ofrecidad por las estadísticas básicas. Así, es típico que los relojes GPS ofrezcan la siguiente información al terminar la actividad:

- Velocidad (km/h) máxima alcanzada en la actividad.
- Velocidad (km/h) del kilómetro más rápido.
- Desnivel positivo acumulado del recorrido.
- Consumo de calorías.

Esta tarea requiere extraer los pasos kilométricos del recorrido para poder decidir el decidir qué kilometro es el más rápido y así obtener su valor en km/h. Debido al error provocado por el muestreo, puede que extracción del kilometro no sea exacta y nos pasemos en unos pocos metros.

La función *obtEstadisticasAvanzadas*, dados los datos de la actividad, devuelve las estadísticas correspondientes definidas más arriba.

3.6. Realizar un informe de los *tracks*

El subprograma *generarInformesTracks* procesa todos los ficheros que contienen los datos de las actividades almacenadas en la carpeta *TrackFiles* y genera los informes correspondientes. El informe de una actividad consiste en:

- Informe con las estadísticas básicas y avanzadas de la actividad que se muestra en la salida estándar del sistema. El informe tendrá el siguiente formato:

```
Log: Athlete7/activity-Athlete7-03.csv

Duración: 2h:00m:20s
Distancia: 44.70 km
Velocidad media: 22.29 km/h
Velocidad máxima: 61.56 km/h
Km + rápido: 51.98 km/h
Desnivel: 965.40 m
Calorías: 1735.62 kcal
Frecuencia cardiaca media: 128.78 p/m
Distribución de zonas FC:
Z1 (Resistencia): 41.30%
Z2 (Moderado): 41.66%
Z3 (Ritmo): 12.07%
Z4 (Umbral): 4.97%
Z5 (Anaeróbico): 0.00%
```

- Gráfica que muestra el perfil del recorrido junto con la evolución de la frecuencia cardíaca (ver figura 1). Los ficheros correspondientes a las gráficas se guardarán en la carpeta *GeneratedPlots*. Los nombres de los ficheros están estandarizados de acuerdo al patrón

  plot-*identificador*-*num*-perfil.png

  donde:

  1. **identificador**: codigo alfanúmerico que identifica al ciclista.
  2. **num**: número de dos dígitos que identifica cada actividad del ciclista.

Recordad que las actividades están en las carpetas de cada atleta. Para obtener los ficheros existentes podéis generar el *path* relativo para acceder directamente a los ficheros de forma automática. Esto es si queremos acceder a la actividad 4 del o la ciclista 1, generaríamos el siguiente nombre de fichero:

TrackFiles/Athlete1/activity-Athlete1-04.csv

4. Evaluación

Para la evaluación de la práctica se aplicarán los siguientes criterios:

- La práctica debe incluir los diseños de los subprogramas desarrollados. **Toda práctica que carezca de los algoritmos, tendrá 0 como calificación**.
- Los diseños se deben corresponder con la implementación proporcionada. **Si los algoritmos presentados en la documentación no corresponden a los subprogramas implementados, la práctica tendrá 0 como calificación**.
- La práctica debe funcionar adecuadamente. **Toda práctica con errores de compilación o que no realice lo que se pide, tendrá una calificación de 0**.
- Las prácticas **son individuales**. Podéis comentar los métodos para realizar las operaciones solicitadas, pero **si se detectan prácticas copiadas la calificación será 0**.
- Se valorará la calidad de la solución propuesta, considerando los subprogramas identificados.
- Se valorarán la claridad del algoritmo y del código, así como la documentación del proyecto.

Se proporcionará dos formas de alcance en el cumplimiento de la práctica. El alumno que opté realizar las tareas de la sección 3.1 (estadísticas básicas) hasta la 3.4 (consumo energético) y modificando la tarea 3.6 para que no muestre las estadísticas avanzadas, podrá obtener una **calificación máxima de 8 sobre 10**. **Los alumnos que entreguen la implementación de todas las tareas se les evaluará sobre 10**.

5. Documentación a entregar

Se debe subir a eGela un archivo comprimido que contenga el proyecto (con la implementación de la práctica y la documentación del código) y un documento con los algoritmos de los suprogramas implementados.

Fecha de entrega: 23 de Diciembre