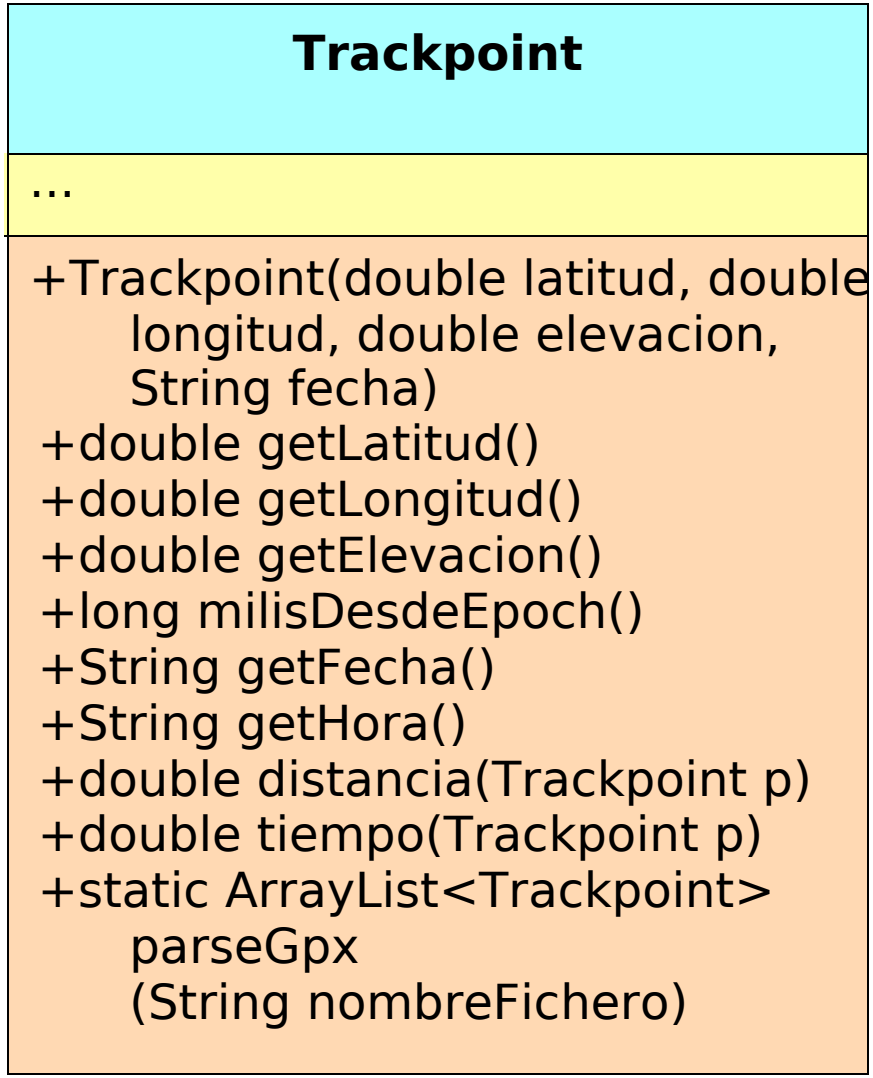
Práctica 10

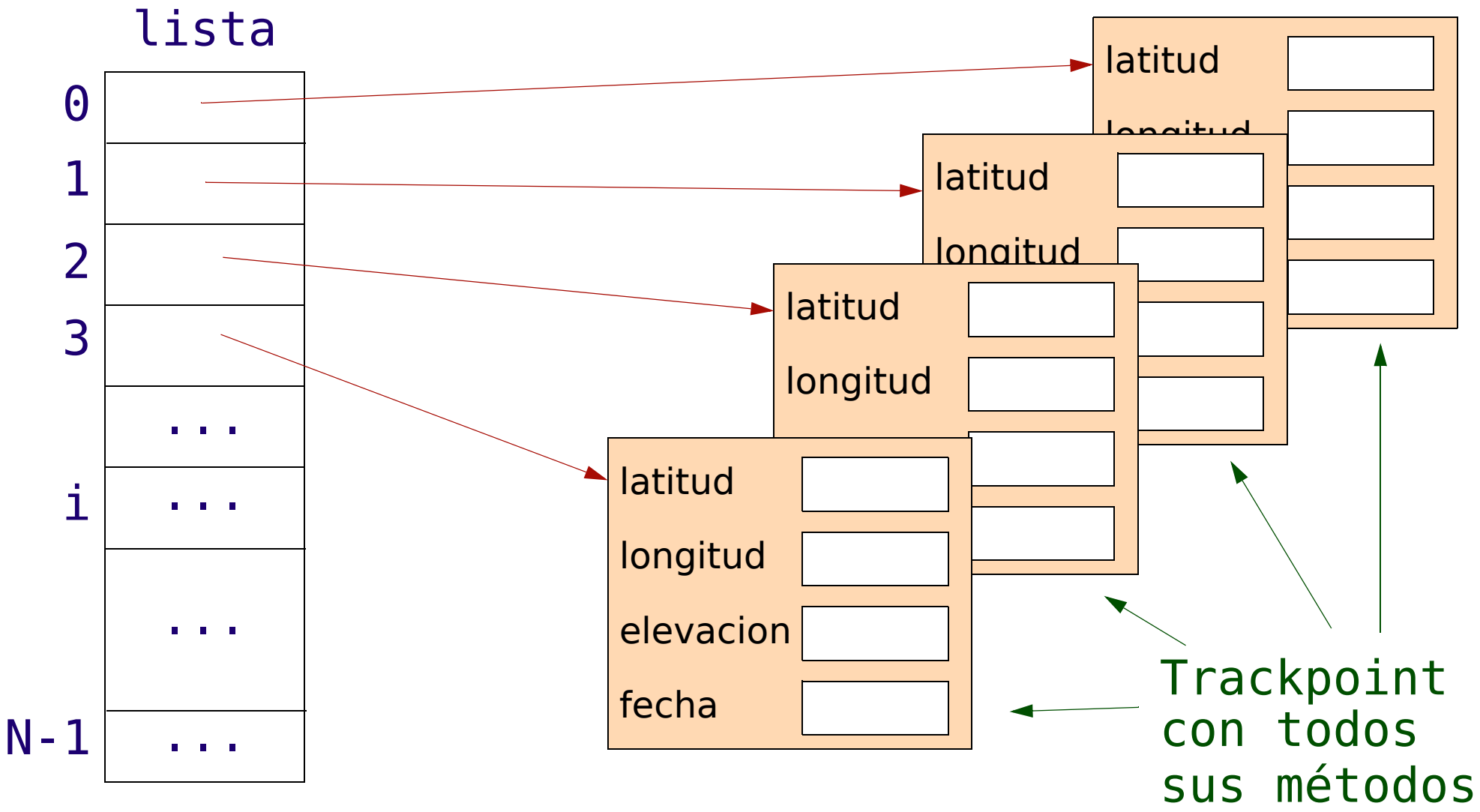
Objetivos: Practicar recorridos y búsquedas en un `ArrayList`

Descripción: Se dispone de una clase que representa un punto de una ruta registrada mediante un GPS

- la ruta completa se puede leer de un fichero en formato GPX
- consultar la documentación de la clase para saber cómo utilizar los métodos

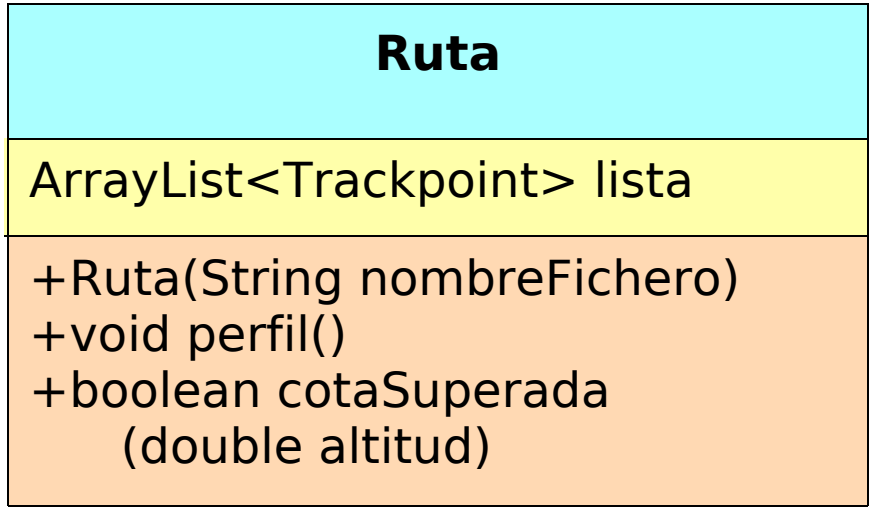


Estructura del ArrayList de Trackpoints



Clase **Ruta**

Se desea escribir la clase `Ruta`, que tiene como atributo un `ArrayList` de objetos de la clase `Trackpoint`



Métodos:

- *Constructor*: introduce en el atributo los `Trackpoints` guardados en el fichero cuyo nombre se pasa como argumento
  - Usa para ello el método `parseGpx()`
  - Se dispone de varios ficheros con rutas GPX para probar este constructor

Clase **Ruta** (cont.)

`perfil`(): representa, utilizando un objeto de la clase `Grafica`, el perfil de la ruta, es decir, la elevación de cada punto en m frente a la distancia recorrida en km

- Para obtener la distancia recorrida, habrá que crear una variable acumuladora donde se vayan sumando las distancias entre el punto actual y el anterior a medida que se vaya recorriendo el `ArrayList`
- Para calcular la distancia entre dos puntos se dispone del método `distancia`()

`cotaSuperada`(): retorna un valor lógico que indica si la ruta asciende por encima de un valor que se pasa como parámetro, en m

Programa principal

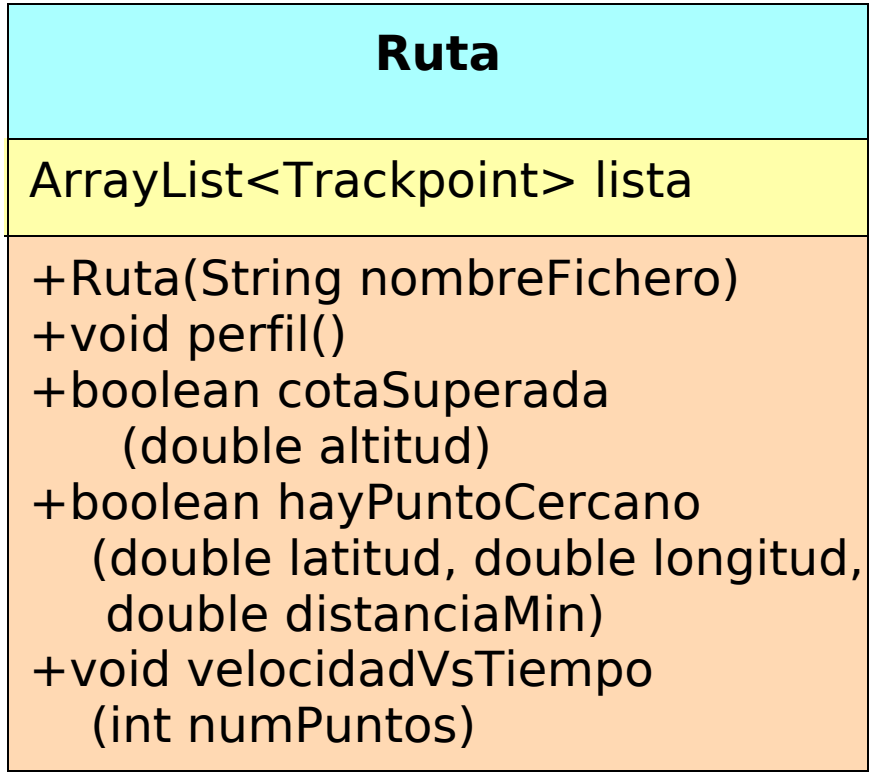
Se pide escribir un programa principal, contenido en otra clase, que haga lo siguiente:

- Crear un objeto de la clase `Ruta` a partir del fichero `Sierra de Hijar.gpx`
- Indicar si la ruta ha superado las cotas de 1000, 1500 y 2500 m
- Mostrar la gráfica del perfil de la ruta

Parte avanzada

Se pide añadir a la clase `Ruta` los dos métodos siguientes, y probarlos añadiendo al `main` instrucciones que los invoquen:

- `hayPuntoCercano`(): Retorna un valor lógico que indica si algún punto de la ruta se encuentra a una distancia menor a la que la que se pasa como parámetro, en km, respecto al punto cuya latitud y longitud también se pasan como parámetros, en grados

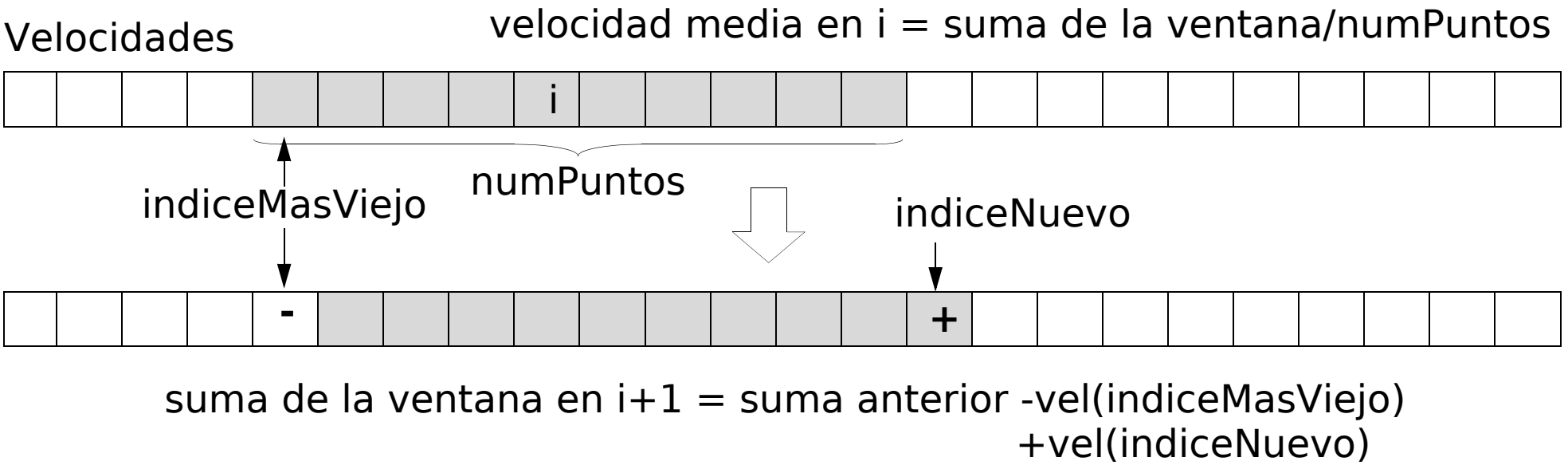


Parte avanzada

- `velocidadVsTiempo`(): representa una gráfica de la velocidad, en km/h, en funcion del tiempo transcurrido, en horas.
  - Recibe como argumento el numero de medidas de velocidad que promedia para evitar fluctuaciones en la velocidad (típicamente, 10 o mas)
  - La velocidad en cada punto es la distancia recorrida desde el punto anterior dividida entre la diferencia de tiempo entre ambos puntos
  - Si la ruta tiene un numero de `Trackpoints` menor o igual que `numPuntos+2` se pone en pantalla un mensaje de error y no se pinta la gráfica
  - Para este método usar el pseudocódigo que se indica a continuación

Diseño del método velocidadVsTiempo

Utilizaremos la técnica de la "*ventana*" móvil para calcular de forma eficiente la media de las velocidades en el intervalo `numPuntos`



Para poder calcular bien la media se hace el cálculo desde `primerIndice=(numPuntos+1)/2` hasta `ultimoIndice=lista.size()-numPuntos/2 - 1`

Pseudocódigo

Definimos dos métodos privados:

```
// retorna el intervalo desde el punto i-1 al i en horas
método intervalo (entero i) retorna real
    retorna (milisDesdeEpoch() del punto i de lista -
          milisDesdeEpoch del punto i-1 de lista)/1000.0/3600.0
fin método

// retorna la velocidad entre los puntos i-1 e i en km/h
método velocidad (entero i) retorna real
   retorna (distancia entre los puntos i e i-1 de lista)/
           1000.0/(intervalo del punto i)
fin método
```

Pseudocódigo (cont.)

```
método velocidadVsTiempo (entero numPuntos)
  // Comprobación de errores
  si tamaño de lista < numPuntos+2 entonces
     Mostrar mensaje de error
  si no
     Grafica g=nueva Grafica(títulos)
     real tiempoTotal=0
     real sumaVelocidades=0
     entero primerIndice=(numPuntos+1)/2
     entero ultimoIndice=tamaño de lista - numPuntos/2-1

     // Calcular el tiempo hasta antes del primer índice
     //  y sumar las velocidades de esos puntos
     para i desde 1 hasta primerIndice-1
        //añadimos el tiempo desde el punto anterior en horas
        tiempoTotal=tiempoTotal + intervalo(i)
        // sumamos la velocidad en Km/hora
        sumaVelocidades= sumaVelocidades + velocidad(i)
     fin para

     // Añadir velocidades hasta completar numPuntos-1
```

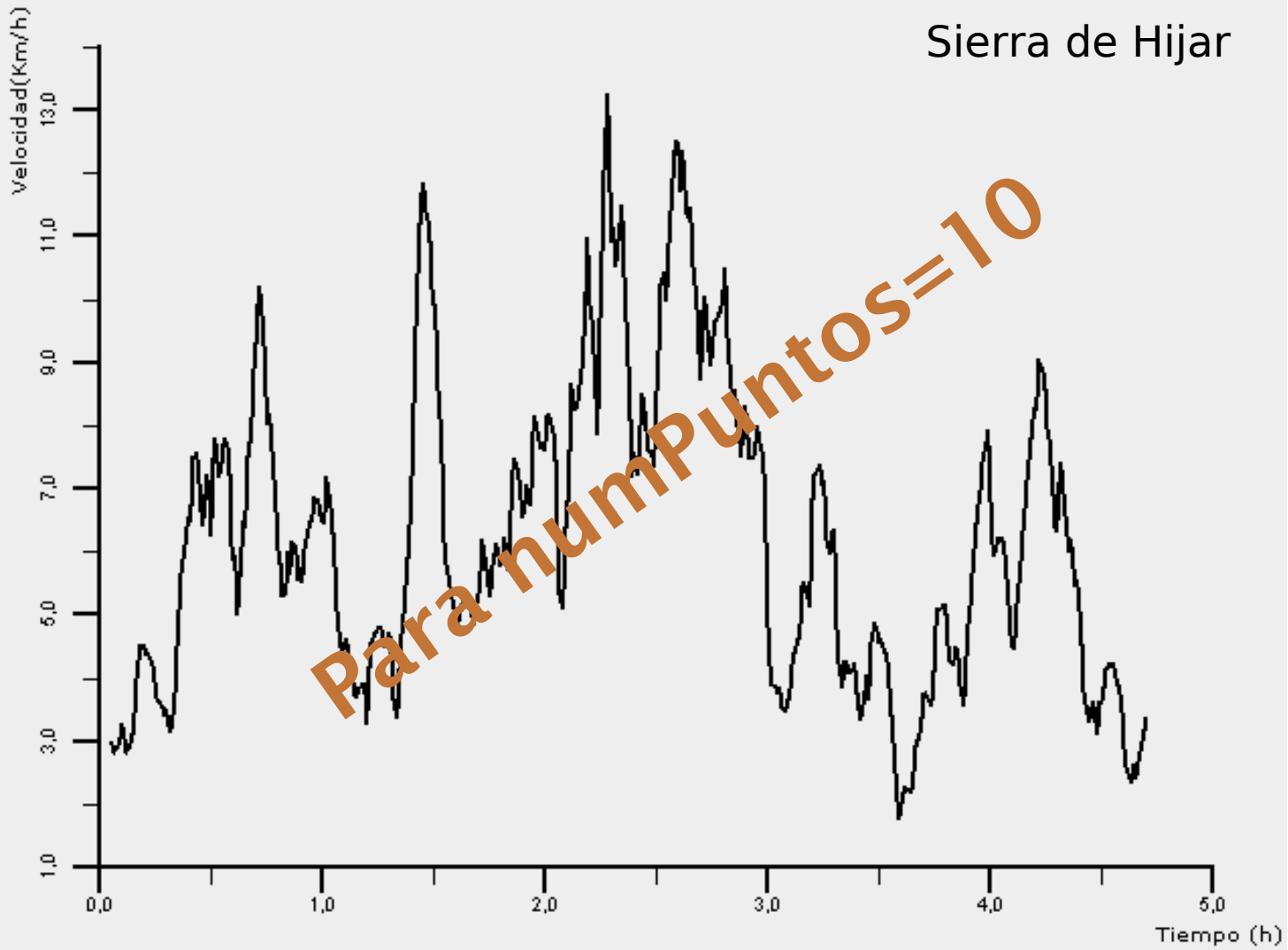
```
      para i desde primerIndice hasta numPuntos-1
         sumaVelocidades = sumaVelocidades + velocidad(i)
      fin para

       // recorre los puntos entre el primerIndice y el ultimoIndice
      para i desde primerIndice hasta ultimoIndice
         // actualizar el tiempoTotal y la suma de velocidades
         tiempoTotal = tiempoTotal + intervalo(i)
         entero indiceNuevo = i+numPuntos/2;
         sumaVelocidades = sumaVelocidades + velocidad (indiceNuevo)
         // calcular la velocidad media en este punto
         real velocidad = sumaVelocidades/numPuntos

         // meter el punto en la gráfica
         g.inserta(tiempoTotal,velocidad)

         // quitar a la suma de velocidades la del punto mas viejo
         entero indiceMasViejo = i-(numPuntos-1)/2
         sumaVelocidades = sumaVelocidades - velocidad(indiceMasViejo)
      fin para
      g.pinta()
   fin si
fin método
```

Parte avanzada (ejemplo de la gráfica)



Entregar

Informe con:

- El código Java de la clase `Ruta`
- El código Java de la clase que contiene el programa principal
- Una captura de pantalla de la gráfica y la salida obtenida en la consola de Java

Parte *avanzada*

- El código Java de los nuevos métodos
- Una captura de pantalla de la `Grafica` obtenida con el nuevo método `velocidadVsTiempo`