Interfaces Persona Computador DSIC 2015-16

Second Deliverable

# Tracking and analysis of physical activity

## Overview of the project: functional requirements

It is common for runners and cyclists to track their training sessions using GPS devices and other sensors. Particularly, it is very common to use a heart rate monitor and rhythm sensors to detect the pedaling or running rate. Different manufacturers have developed their own formats to store the information captured by those devices. On the other hand, the open standard GPX has been developed for making it easier to transfer data between different applications (https://es.wikipedia.org/wiki/GPX, http://www.topografix.com/gpx.asp).

In this deliverable you will develop a desktop application for evaluating the performance of runners or cyclists by enabling the user to analyze the training sessions and competitions stored in GPX files.

### Functional Requirements

A) Overview of a session. The summary of a session includes the following data:
   a. Start date and time.
   b. Duration (hours, minutes and seconds)
   c. Exercise time (hours, minutes and seconds)
   d. Total distance (Km)
   e. Accumulated slope (both up and down) (in meters)
   f. Maximum and average speed (Km/h)
   g. Heart Rate (HR) maximum, minimum and average (beats per minute)
   h. Maximum and average pedaling rate (pedals per minute)

B) Charts for displaying:
   a. Height x Distance (Track profile), using an AreaChart
   b. Speed x Distance, using a LineChart
   c. HR x Distance, using a LineChart
   d. Pedaling rate x Distance, using a LineChart

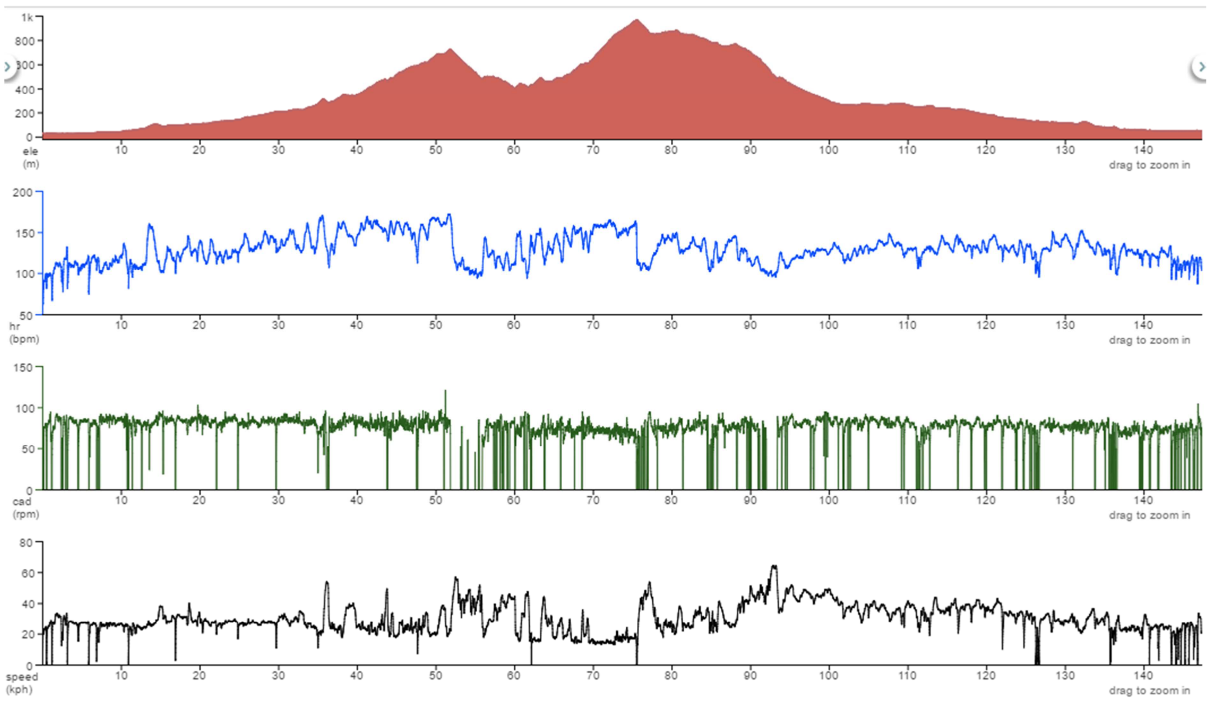
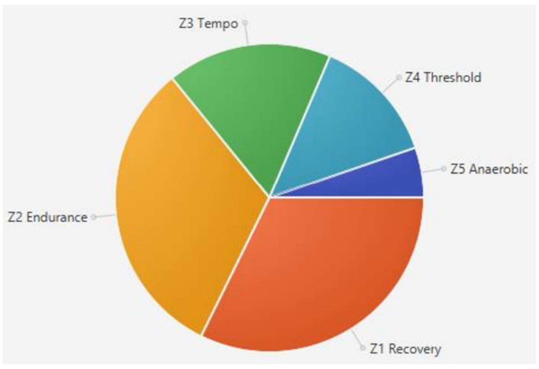


*Figure 1 Distance-based charts*

C) PieChart showing the time distribution the person spent in each of the following 5 heart rate zones. It is necessary to ask the sportsman/woman maximum HR, because the different thresholds depend on that value. After figuring out the limits of each zone, you will have to compute how long the sportsman spent in each zone. The following figure shows an example.

| | |
|---|---|
| Z1 Recovery | <60% of the maximum HR |
| Z2 Endurance | 60%-70% |
| Z3 Tempo | 70%-80% |
| Z4 Threshold | 80%-90% |
| Z5 Anaerobic | 90%-100% |



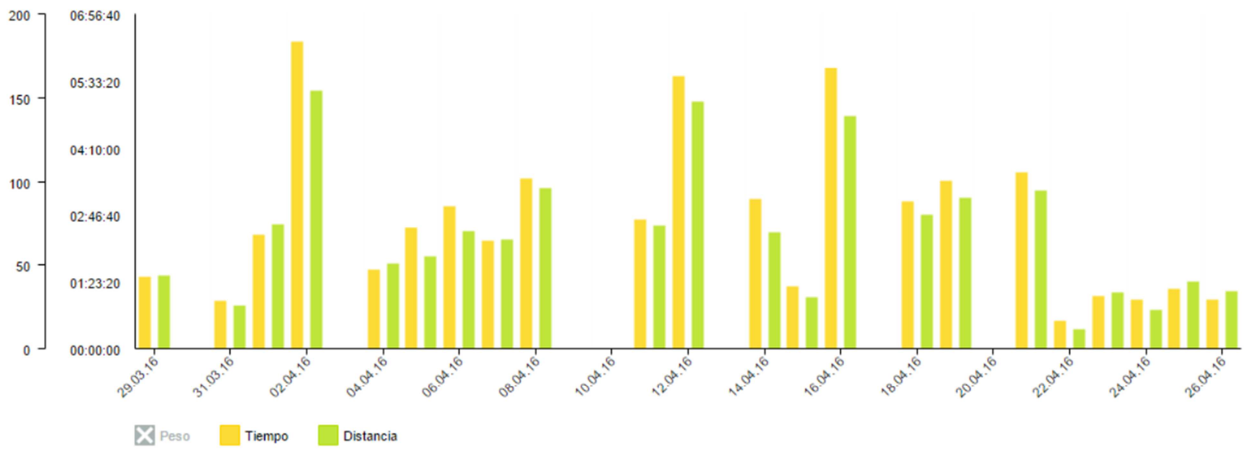
## Optional Extensions

You can implement the following extension for obtaining extra credit:

A) Allow the charts to display the information based on time instead of the distance. That is, the X axis will read the time (Height x Time, Speed x Time, etc.)

B) Combine the HR data, pedaling rate and speed in a single chart with several data series. Add controls for showing/hiding the different series.

C) Training notebook. Allow the user to load multiple GPX files and show an overview of the activity performed in the last month or in the last three months. For example, the following bar chart shows the daily activity durations and the total distance during a month.



## Guide for Solving the Assignment

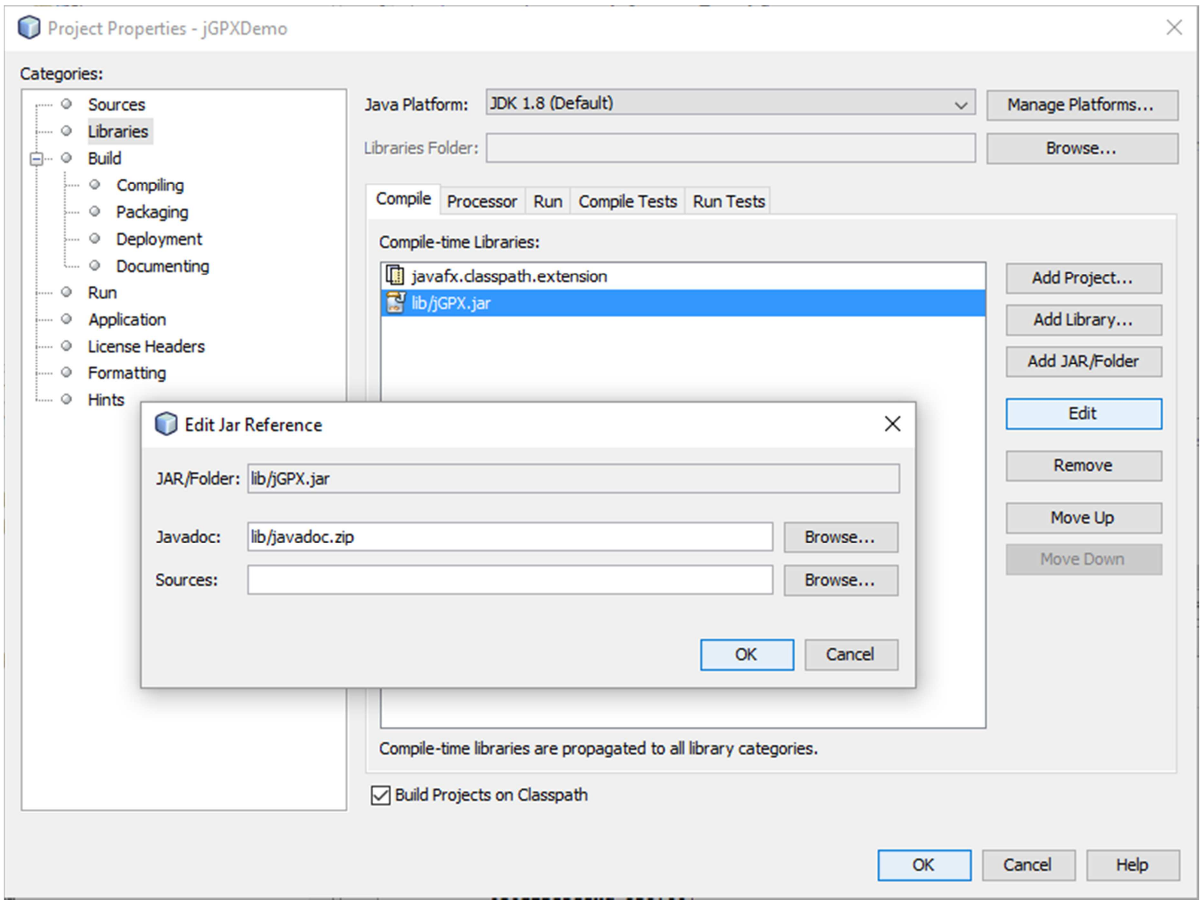
In this deliverable we provide a library (**`jGPX.jar`**), with all the required functionality for reading and processing GPX files.  For adding this library to your project, create a folder called **`lib`** besides the **`src`** folder and import it into your project using the option **`Properties`** > **`Libraries`** > **`Add JAR/Folder`**. When exporting the project to a ZIP file, Netbeans will include this library.

After adding the library, and in the same window, select the jGPX icon and select the button Edit. Then, in the Javadoc section of the dialog, select the provided file *javadoc.zip*, which contains the documentation of the library. The Figure 2 shows this Netbeans step.

We also provide an example on how to use the library. It is available in the file **`jGPXDemo.zip`**, that can be imported in Netbeans (both the library, jGPX.jar, and its documentation, javadoc.zip, should appear in the lib folder of the project). Study the source code of the example for learning how to use the library. You can also study the main classes of the project by pressing Alt+F1 while the cursor is in a class. It is also possible to right click on the library in the navigation panel of Netbeans and selecting the option “Show javadoc". The most important classes for this assignment are **`TrackData`** and **`Chunk`**.

For reading a GPX file you will use some helper classes ready to be used by the JAXB API. In particular, the class in charge of representing the contents of a complete GPX file is called **GpxType**. The following code shows how to load a GPX file into an object **GpxType**:

```
JAXBContext jaxbContext = new JAXBContext(GpxType.class,
                              TrackPointExtensionT.class);
Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();
JAXBElement<Object> root = (JAXBElement<Object>)
unmarshaller.unmarshal(file);
GpxType gpx = (GpxType) root.getValue();
```


Project Properties - jGPXDemo

Categories: Sources, Libraries, Build, Compiling, Packaging, Deployment, Documenting, Run, Application, License Headers, Formatting, Hints

Java Platform: JDK 1.8 (Default)

Libraries Folder:

Compile-time Libraries:

javafx.classpath.extension

lib/jGPX.jar

Compile-time libraries are propagated to all library categories.

☑ Build Projects on Classpath

Edit Jar Reference

JAR/Folder: lib/jGPX.jar

Javadoc: lib/javadoc.zip

Sources:

*Figure 2 Adding the library jGPX.jar to the project and editing the library for setting its Javadoc location*

After obtaining the GpxType object the next step is obtaining a TrackData object, which processes the data in the GPX file and obtains all the derived information required to analyze the activity, like duration, distance, speed, etc. Please, take into account that the last two lines in the previous code are a little bit different than in previous lab assignments. Do not change those lines, or the *unmarshal* method will fail.

```
TrackData trackData = new TrackData(new Track(gpx.getTrk().get(0)));
```

Note: in theory, a GPX file can contain several tracks, but the provided examples contain just one track, that is retrieved with the get(0) invocation.

After obtaining the TrackData object we can use its public methods for obtaining all the relevant information about the track, like:

- `getAverageSpeed()`
- `getMovingTime()`
- `getTotalDuration()`
- `Etc.`

In order to create the charts, it is necessary to access to the data in each data point in a track. Since two data points are required for computing some derived data (speed, time, distance, etc.), instead of using directly the data points, we will work with pairs of data points. This information is encapsulated in objects of type **Chunk**. The following code shows how to obtain the list of chunks from a track:

```
ObservableList<Chunk> chunks = trackData.getChunks();
```

The information stored in each chunk can be accessed with methods such as:

- `getSpeed()`
- `getDistance()`
- `getClimb()`
- `etc.`

## Delivery rules

Export the NetBeans project in a ZIP (http://politube.upv.es/play.php?vid=68666) and upload it to the poliformaT task for this assignment. Each group will upload the project only once, including in the comments section the name of both students.

## Evaluation:

- The projects that fail to compile or that fail to show the main window after launch will be graded with a zero.
- The user interface must always be responsive. Take into account that processing large GPX files for obtaining the data series or processing several GPX files can take a while.
- The application should include all the required confirmation dialogs, error dialogs, etc.
- The design of the interface will take into account the design guidelines studied in class.
- The main window must be resizable. The position and size of their controls should adjust to use the available space (use the containers studied in the lab).