Course 2018-2019

Deliverable 2. IPC Laboratory

INTERVAL TIMER FOR A CROSSFIT ROOM

IPC

UPV-DSIC

Content

1. *Interval Timer* for a CrossFit room. General Description

For the delivery of IPC practices 2, a desktop application must be developed to be used in a crossfit / fitness room, which will help to carry out the training sessions.

The application should act as an interval chronometer. This application will continuously indicate the different work and rest intervals that must be carried out during a training session. When the session is started, the chronometer will indicate to all the users of the room either the time executed, or the remaining time of the current interval (at your choice).

In case of working time, an exercise will be done. In the case of being rest time, it will serve as preparation for the following exercise.

The application must warn with acoustic and / or visual signals the beginnings or endings of each of these intervals. The interface of the application will be displayed on a screen, visible by all participants of the session and will be a replica of the interface of the computer screen.

The coach has to define the content training session before starting it. A session may consist of a previous warm-up time and then be structured in a series of exercises. These exercises will be done during a pre-established time that is known as work time. In our case, this time will be the same for all the exercises. At the end of each exercise, there will be a rest period between exercises (the same during the whole session). Once the circuit is completed with the different exercises proposed, there will be a rest time between circuits. At the end of this time, the previous circuit will be repeated. The number of repetitions of the circuit is another parameter of the session.

Once the coach initiates the interval manager session, he will have to adequately inform about the elapsed or remaining time in each of these intervals. In addition, it will indicate the number of the current exercise and the number of the current circuit. It is very important to warn with sound and / or light signals that the time in question is about to expire.

During the session, the coach can stop the chronometer at any of the intervals: during the performance of an exercise, during rest between exercises or during breaks between circuits. You can resume the time where you stopped or you can move directly to the next interval. He can also restart the session, in this case the whole process will start again but the time at which the session started will be maintained.

In order to have a record of the actual duration times of each session, the time elapsed from the beginning to the end of this session should be stored. A session ends when your last work interval ends.

The application will able to analyse the history of activity carried out in the sessions. In this way, in our model we are going to take into account the existence of groups, of which we are going to save:

- one code
- a description
- all sessions it do

From each of these sessions will be stored:

- date
- starting time of the session
- actual session duration
- the description of the work performed

Templates of sessions will be defined to describe sessions, so the coach will be able to configure a session with one of the previously saved template or create a new one.

Para mantener la integridad de las sesiones ya realizadas, una *sesión tipo* no se podrá borrar ni modificar. Cada sesión tipo tendrá:

To maintain the integrity of the sessions already made, a session template cannot be deleted or modified. Each session template will have:

- a code
- a warming time (zero indicates there is not warming time)
- the number of exercises
- time of the exercise working time
- rest time between exercises
- the number of circuit repetitions
- rest time between circuits

All the times will be indicate in seconds.

Para agilizar el trabajo de configuración de la sesión, un grupo tendrá una sesión tipo por defecto que siempre será el tipo de sesión realizado en su última sesión, en el caso de tener por lo menos una sesión realizada.

To speed up the session configuration work, a group will have a default session template that will always be the one used in the last session, in the case of having at least one session performed.

2. Features to develop

The features that must be develop are:

- Adding/modifying a Group
- Adding a Session Template
- Perform a session (stop, Realizar Sesión (detener, resume, advance)
- Show graphs of the sessions of a group

A history of the sessions will be shown in the graphs of the sessions of a group, which will allow to analyse the work of each group. Thus, a line graph will be shown with the work times of each session, rest times of each session and real time of the session. These three variables, as well as the group

must be configurable. It would be desirable to be able to choose the maximum number of sessions to be represented on the X axis

The design of the interface is open to the developer, it is recommended to make a prototype on paper previously and use all the guidelines and design principles described in the theory classes. In case of doubt, you can review the design with your laboratory teacher.

To facilitate the development of the delivery, we will provide the classes that represent the model, as well as the necessary functions to save and retrieve the information groups and their sessions in XML files. The persistence of the data in XML files is described in the laboratory material.

3. Time management

There are different ways to manage time in java, in our case it is recommended to use the notions of processing in the background of JavaFX that we have seen in the laboratory. To obtain precise values of the system clock we can use the function:

```
long nowTime = System.currentTimeMillis();
```

In this way we can obtain precisely the milliseconds elapsed between two instants by subtracting the time obtained in each one of them. For working with intervals, we have the Duration class:

```
Duration duration = Duration.ofMillis(totalTime);
final Long minutos = duration.toMinutes();
final Long segundos = duration.minusMinutes(minutos).getSeconds();
```

This code shows how to obtain the minutes and seconds contained in a time interval expressed in milliseconds.

4. Classes of the model

The classes that make up the model of this application are: Gym, Grupo, SesionTipo and Sesion. These classes are already included in a library that we provide. In order to use these classes it will be necessary to include the library in the project as indicated below.

Gym class

```
public class Gym {

    private ArrayList<Grupo> grupos = new ArrayList<>();
    private ArrayList<SesionTipo> tiposSesion;

    public Gym() {...2 lines }
```

Grupo class (for manage the groups)

```
public class Grupo {

    private String codigo;
    private String descripcion;
    private ArrayList<Sesion> sesiones;
    private SesionTipo defaultTipoSesion;
```

SesionTipo class (for manage the session templates)

```
public class SesionTipo {
    private int t_calentamiento;

    private int num_ejercicios;
    private int t_ejercicio;
    private int d_ejercicio;

    private int num_circuitos;
    private int d_circuito;
```

Sesion class (for manage the sessions)

```
public class Sesion {

    private LocalDateTime fecha;
    private SesionTipo tipo;
    private Duration duracion;
```

5. Classes to store and retrieve the data of the model in XML

The class AccesoBD, which implements the singleton pattern, has been created to store and retrieve the application data. In this way whenever we invoke the static method getInstance (), we will get a single instance of the class AccesoBD.

This class implements the methods:

```
public Gym getGym() {...18 lines }
```

This method returns an object of Gym type initialized with the data stored in the XML file. It is the responsibility of the application to use this object to add Grupos, Sesiones y SesionTipo.

```
public void salvar() {...18 lines }
```

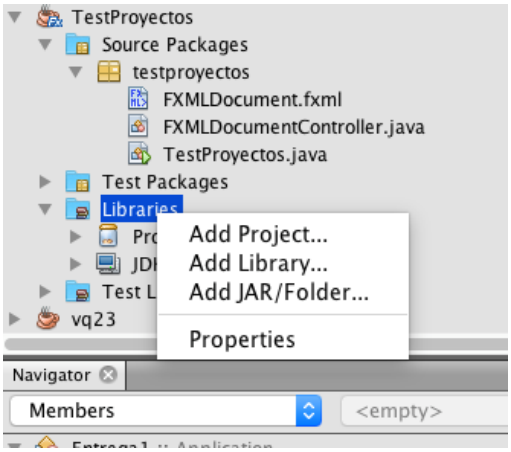
This method stores in the XML file all the changes made over the object Gym gotten when getGym() is called.

6. How to add a jar to a Java project

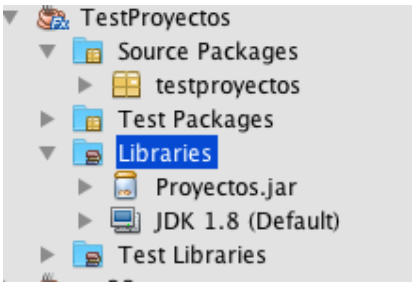
To add a library or a JAR class file to a java project, we recommend first store the library inside the project directory, in this way the references to the library will be through a relative path and the application can be executed without errors. For this, we recommend creating a folder in the root of the project that can be called “lib”, for example. Then, you will have to copy inside this folder the library that in this case is called AccesoBD.jar. In addition, the library is complemented with the javadoc and the classes so that you can access the code and the documentation, also copy the files AccesoBD -source.zip and AccesoBD-javadoc.zip.

Para añadir una librería al proyecto lo podremos hacer desde las propiedades del proyecto o simplemente desde el explorador del proyecto en la carpeta *Libraries* Si nos situamos sobre esta y pulsamos el botón derecho nos aparece la opción de añadir la librería.

To add a library to the project, we can do it from the properties of the project or simply from the project explorer in the folder *Libraries.* If we place ourselves on this and press the right button we will see the option to add the library



Then, you should select *Add JAR/Folder* and choose the file *Proyectos.jar* that is inside the lib folder. The final result should be:



7. Instructions for delivering

Export the NetBeans in a ZIP file (https://media.upv.es/player/?id=a0bfd620-021e-11e6-851a-656f7e06a374) and upload it to the corresponding task of PoliformaT. Only one of the team members should upload the Project, writing in the commentary field the name of the both students.
Evaluación:

- Projects that do not compile or that do not show the main screen when it starts will be rated with a zero.
- Confirmation dialogs, errors, etc. must be included that is considered necessary.

- To evaluate the design of the application interface, the guidelines studied in theory class will be taken into account.
- The design and dimensions of the interface must be appropriate to the environment of use of this application