

Estructura de Datos y Algoritmos

ITBA 2024-Q2

TP 1-

Ejer 5.1 y 5.2



Crear un nuevo proyecto Maven
AnalysisTime

**Vamos a usar nuestra
biblioteca hecha TimerJoda:**

**Asegurarse que TimerJoda fue
instalada en el repositorio
local**

Armar un nuevo proyecto mvn llamado **AnalysisTime** que use a la biblioteca **TimerJoda**, pero NO DECLARE a Joda Time en el pom.xml

Es decir, en este proyecto colocar solo:

```
package main;

...
public class Proof{
    public static void main(String[] args) {
        Timer myCrono = new Timer(10);
        myCrono.stop(10 + 93623040);

        System.out.println(myCrono);
    }
}
```

¿Cómo se declara TimerJoda (la nuestra) en este pom.xml?

Debe poder instalarse (compilar correctamente).

¿Ejecuta correctamente? Qué bibliotecas está usando? Como se resolvió?

TP 1-

Ejer 5.3, 5.4 y 5.5



Tiene `AnalysisTime.jar` dentro las 2 bibliotecas que sabemos usa?

Imaginemos que le entregamos `AnalysisTime.jar` a un cliente.

No podemos esperar que las bibliotecas `TimerJoda` y `Joda Time` (usada indirectamente) estén en su computadora.

¿Cómo resolver esta situación?

Si desde línea de comandos usamos (parados en cualquier lado)

```
$ java -cp  
c:\Users\lgomez\.m2\repository\ar\edu\itba\eda\AnalysisTime\1\Anal  
ysisTime-1.jar main.Proof
```

¿Qué sucede?

Si se quiere incluir en nuestro jar, otras bibliotecas que usar directa/indirectamente usamos otro mvn plugin. Regenerar mvn install

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-assembly-plugin</artifactId>
  <version>3.3.0</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
      <configuration>
        <descriptorRefs>
          <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
      </configuration>
    </execution>
  </executions>
</plugin>
```

TP 1-

Ejer 5.6



Intentar ejecutar desde línea de comandos el archivo jar.

```
$ java -cp  
c:\Users\lgomez\.m2\repository\ar\edu\itba\eda\A  
nalysisTime\1\AnalysisTime-1-jar-with-  
dependencies.jar main.Proof
```

Si bien funciona OK, es engorroso pedirle al usuario que indique la clase que contiene el main.

Si ejecutamos sin indicar “Proof”

```
$ java -jar  
c:\Users\lgomez\.m2\repository\ar\edu\itba\eda\AnalysisTime\1\AnalysisTime-1-jar-with-dependencies.jar
```

No funciona.

Agregar al pom “mainClass”: package.nombreClass

Si se quiere incluir en nuestro jar, otras bibliotecas que usar directa/indirectamente usamos otro mvn plugin. Regenerar mvn install

<plugin>

 <groupId>org.apache.maven.plugins</groupId>

 <artifactId>maven-assembly-plugin</artifactId>

 <version>3.3.0</version>

 <executions>

 <execution>

 <phase>package</phase>

 <goals>

 <goal>single</goal>

 </goals>

 <configuration>

 <descriptorRefs>

 <descriptorRef>jar-with-dependencies</descriptorRef>

 </descriptorRefs>

 </configuration>

 </execution>

 </executions>

</plugin>



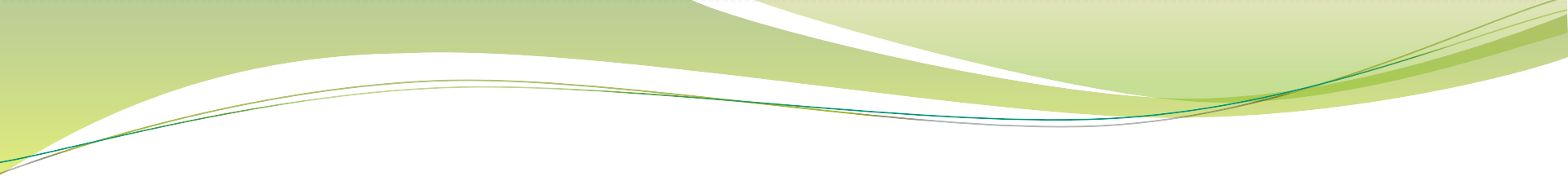
```
<archive>
  <manifest>
    <mainClass>main.Proof</mainClass>
  </manifest>
</archive>
```



¿Ejecuta el jar con dependencias ahora sin indicar la clase principal?

Explicar por qué anda OK.

Abrir el jar que generó. ¿Qué tiene el archivo Manifest.mf?



Actualmente, existe otra implementación del Timer: Java 8 (nativo) también agregó una biblioteca para manejo de timers

TP 1-

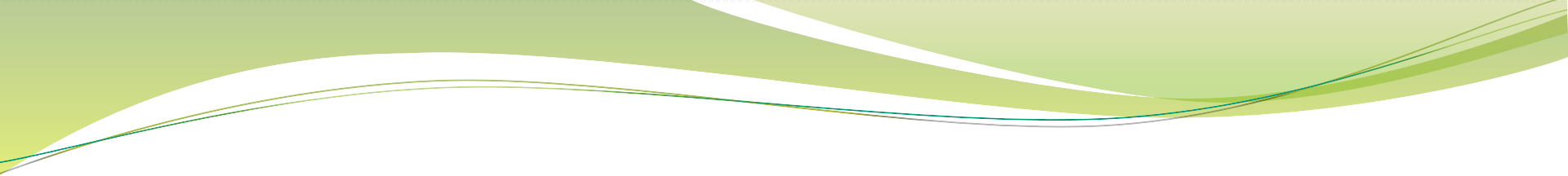
Ejer 6



A partir de Java 8, java agregó un manejo para Timers...

Armar un nuevo proyecto maven para usar dicha API.

Nuestra versión 3 será un wrapper sobre la misma.



```
<groupId>ar.edu.itba.eda</groupId>  
<artifactId>TimerNativo</artifactId>  
<version>3</version>
```

Y agregar en el pom lo necesario para que pueda empaquetarse, etc.

Investigar las clases Instant y Duration

Implementar nuestra versión usando dicha biblioteca.

- ¿ Cómo estar seguros de que funciona realmente correctamente las 3 implementaciones?
- ¿ Cómo asegurarnos que cuando implementemos nuevos algoritmos, sigue funcionando correctamente?