PAQUETES JAVA

Java utiliza los paquetes para facilitar la modularidad del código y su reutilización. **Un paquete es una agrupación de clases** que tienen algo en común. Las clases que pertenecen a un paquete pueden usar clases definidas en el mismo o en otros paquetes.

Los paquetes pueden tener subpaquetes y formar una jerarquía de paquetes que dará lugar en el disco duro a una estructura de directorios equivalente a la jerarquía formada y donde, cada clase, estará en el directorio/subdirectorio correspondiente al paquete/subpaquete al que pertenece).

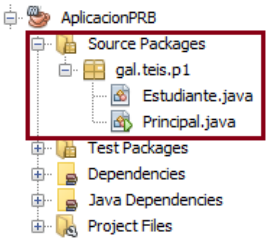
Las clases deben referenciar el paquete al que pertenecen antes de cualquier código específico de la mima, para ello deben utilizar la palabra clave **package** y a continuación el nombre del paquete. El nombre del paquete debe ser un grupo de palabras separadas por punto, estas palabras hacen referencia a una jerarquía que servirá para organizar las clases.

```
package gal.teis.p1;

...5 lines

/**...4 lines */
public class Estudiante {

    int a; // inicializado a c
    static int b=3; // inicial

    Estudiante() {
        a=1;
        // Constructor increme
```

El nombre de una clase debe ser único dentro del paquete donde se encuentre, pero **dos clases de distintos paquetes se pueden llamar igual** ya que la referencia de cada clase en Java es ***nombrePaquete.nombreClase***.

Por ejemplo, las clases del paquete *gal.teis.p1* deben tener como primera línea de código el paquete al que pertenece. La clase *Estudiante* del paquete *gal.teis.p1* tendrá su referencia en el contexto de Java como ***gal.teis.p1.Estudiante***.

PAQUETES DENTRO DE UN PROYECTO JAVA

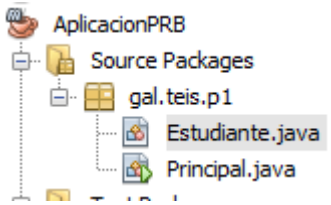
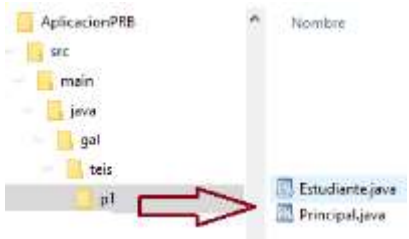


Dentro de un proyecto Java, los paquetes se encuentran dentro de la carpeta **Source Packages**. Los paquetes permiten organizar los ficheros fuente de las clases y las interfaces que se usan en un proyecto.

Los nombres de los paquetes son palabras separadas por puntos y se almacenan en directorios que coinciden con esos nombres.

Por ejemplo, el paquete ***gal.teis.p1*** del proyecto ***AplicacionPRB*** se encuentra en la ruta que se muestra a continuación:

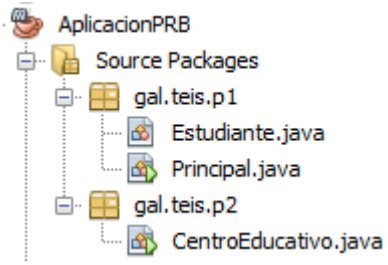
Dentro de la carpeta ***p1*** se encontrarán las clases del paquete ***gal.teis.p1***

Vista desde NetBeans	Vista desde el administrador de archivos
	

IMPORTAR PAQUETES

Si en una **clase A** queremos utilizar una **clase B** que está en otro paquete, en la **clase A** debemos importar el paquete/clase de la **clase B**.

Suponemos que tenemos la siguiente estructura de clases:



Desde la clase ***Principal*** podemos utilizar la clase ***Estudiantes*** directamente pero no la clase ***CentroEducativo***, pues está en otro paquete.

Para poder utilizar la clase ***CentroEducativo*** desde ***Principal*** tenemos que importar la clase, ya que están en paquetes diferentes.

Nota: supongo que las clases no tienen restricciones de uso por su modificador de acceso

Es posible utilizar una clase de un paquete diferente sin utilizar la sentencia **import**, indicando **nombrePaquete.NombreClase**. Por ejemplo:

```
public class Principal {

    public static void main(String args[]) {
        Estudiante s1 = new Estudiante();
        s1.mostraValores();
        Estudiante s2 = new Estudiante();
        s2.mostraValores();
        gal.teis.p2.CentroEducativo cl=new gal.teis.p2.CentroEducativo();
    }
}
```

En Java, no es necesario importar el paquete **java.lang** pues todas las clase de dicho paquete se pueden usar directamente. Para el resto de paquetes, bien sean de la API o no, será necesario realizar la operación de **import**.

RECOMENDACIONES

- Los nombre de los paquetes se deben escribir en minúscula.
- Utilizar el nombre del dominio de nuestra organización en forma inversa, por ejemplo, si nuestra empresa tiene como dominio miempresa.com, los paquetes deberían comenzar por **com.miempresa**.
- Nunca se debe crear una clase sin asignarle nombre de paquete, en este caso la clase pertenece al paquete *sin nombre* y no podrá ser referenciada por las clases de otros paquetes.

Para saber más

PROYECTOS MAVEN

https://juandyb.com/crear-un-paquete-jar-ejecutable-con-las-dependencias-incluidas-usando-maven/

http://www.javahispano.org/portada/2011/8/1/tutorial-y-screencast-de-introduccion-a-maven-3-por-erick-ca.html

https://soajp.blogspot.com/2017/02/anadir-librerias-manualmente-usando.html

http://panamahitek.com/jar-ejecutable-con-dependencias-incluidas-en-java/

Maven es un gestor de repositorios que permite acceder a los ficheros y librerías que se encuentran en repositorios como Github.

Maven también es un lenguaje y un conjunto de normas que permite definir características de nuestro proyecto como la documentación, la generación de ficheros ***jar***, las dependencias con otras librerías, etc.

Maven utiliza un **Project Oject Model** (*POM*) para realizar todas estas operaciones.

Un proyecto creados con **Maven** tienen una estructura característica que contiene en la carpeta raíz un fichero llamado **pom.xml**.

El fichero **pom.xml** es un fichero XML que permite establecer parámetros básicos del proyecto, como el nombre, la versión, la clase principal, etc. También se indican las dependencias y repositorios que el proyecto va a utilizar.

Para utilizar una librería, por ejemplo, *JFreeCharts* que permite hacer gráficas, tan solo tenemos que agregarla de la siguiente forma:

```
<dependency>
    <groupId>jfree</groupId>
    <artifactId>jfreechart</artifactId>
    <version>1.0.13</version>
</dependency>
```

Una de las ventajas de usar **Maven** en la construcción del proyecto es el manejo automático de las dependencias. No será necesario la descarga de librerías, lo único que hay que hacer es especificar su referencia en el fichero **pom.xml** y **Maven** se encargará de descargar las librerías al repositorio local.

CREAR UN JAR EJECUTABLE CON MAVEN

Maven permite crear ficheros **JAR** ejecutables a través de la configuración de todas las dependencias dentro de un solo archivo. Por ejemplo, para el proyecto **Prueba** podemos editar el fichero **pom.xml** siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>

<!—Declaración común a todos los proyectos-->
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<!--Versión del modelo de objetos que está usando este fichero pom-->
    <modelVersion>4.0.0</modelVersion>
<!--Referencia del grupo que creo el proyecto, por defecto, es el
nombre dominio especificado en los paquetes-->
    <groupId>gal.teis</groupId>
<!--Parte del nombre del fichero JAR que se generará-->
    <artifactId>Pruebas</artifactId>
<!--Versión del proyecto. SANPSHOT indica que está en desarrollo-->
    <version>1.0-SNAPSHOT</version>
<!--Tipo de paquete que se va a producir-->
    <packaging>jar</packaging>
<!--Propiedades relacionadas con el proyecto-->
    <properties>
      <!--Codificación. Para los idiomas latinos es UTF-8-->
        <project.build.sourceEncoding>UTF-8
        </project.build.sourceEncoding>
        <!--Plataforma de Java desde la que se crea y ejecuta el
proyecto-->
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>
    <!—Nombre del proyecto-->
    <name>AplicacionPRB</name>

</project>
```

GENERAR DOCUMENTACIÓN

Para generar documentación podemos añadir, antes de la etiqueta final de **</project>** la siguiente información:

```
<reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
      </plugin>
    </plugins>
</reporting>
```

HACER PAQUETE JAR PARA EJECUTAR

Se debe crear una nueva clase *Java Main* que inicie la aplicación con el código siguiente, siendo **Principal** la clase *Java Main* del proyecto.

```
public class Launcher {
    public static void main(String[] args) {
        gal.teis.p1.Principal.main(args);
    }
}
```

En el fichero **pom.xml** debemos añadir el plugin *shade* utilizando el siguiente código:

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>3.0.2</version>
                <configuration>
                    <archive>
                        <manifest>
                            <mainClass>
                                gal.teis.p0.Launcher
                            </mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

Desde NetBeans activamos **Run>Clean and Build** para generar el archivo **JAR** dentro de una carpeta llamada ***shade***

Para ejecutar un archivo JAR desde la línea de comandos:

`java -jar NombreArchivo.jar`

```
D:\IES\6_Teis\Proxectos AD\AplicacionPRB\shade>java -jar Pruebas.jar
Valor de a =1
Valor de b =4
Valor de a =1
Valor de b =5
```

FICHERO POM.XML COMPLETO

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org</groupId>
    <artifactId>DirectGeoreferenceMaven</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>

    <name>DirectGeoreferenceMaven</name>
    <url>http://maven.apache.org</url>

    <properties>
```

```
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <geotools.version>8.2</geotools.version>
        <src.dir>src/main/java/org/directgeoreference</src.dir>
    </properties>

    <build>
        <resources>
            <!-- Define un recurso -->
            <resource>
                <!-- Directorio del recurso dentro del proyecto Java -
->

<directory>src/main/java/org/directgeoreference/interfaz/resources</di
rectory>
                <!-- Destino que indica donde se han de copiar los
recursos dentro del jar -->

<targetPath>org/directgeoreference/interfaz/resources</targetPath>
            </resource>
            <resource>

<directory>src/main/java/org/directgeoreference/geo</directory>
                <!-- Excluyo todos los ficheros java para seleccionar
solo un .properties -->
                <excludes>
                    <exclude>**/*.java</exclude>
                </excludes>
                <targetPath>org/directgeoreference/geo</targetPath>
            </resource>
            <resource>

<directory>src/main/java/org/directgeoreference/help</directory>
                <!-- Excluyo dos carpetas que hay dentro de help (bin
y lib) -->
                <excludes>
                    <exclude>bin/**</exclude>
                    <exclude>lib/**</exclude>
                </excludes>
                <targetPath>org/directgeoreference/help</targetPath>
            </resource>
        </resources>
        <plugins>
            <plugin>
                <!-- Plugin para especificar la versión de la JVM con
la que compilar -->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.3.2</version>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                </configuration>
            </plugin>
            <!-- Genera un jar pero sin empaquetar las dependencias --
>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>2.4</version>
                <configuration>
```

```
                    <archive>
                        <manifest>
                            <addClasspath>true</addClasspath>

<mainClass>org.directgeoreference.App</mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
            <plugin>
                <!-- Plugin para ejecutar tareas de ant en este caso
buscar y reemplazar antes de compilar -->
                <artifactId>maven-antrun-plugin</artifactId>
                <version>1.7</version>
                <executions>
                    <execution>
                        <phase>process-sources</phase>
                        <configuration>
                            <target>
                                <replace
file="${src.dir}/interfaz/Ventana.java" token="panelImagen = new
javax.swing.JPanel();" value="panelImagen = new PanelImagen();"/>
                                <replace
file="${src.dir}/interfaz/Ventana.java" token="protected
javax.swing.JPanel panelImagen;" value="protected PanelImagen
panelImagen;"/>
                            </target>
                        </configuration>
                        <goals>
                            <goal>run</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <!-- Maven Shade Plugin para empaquetar las
dependencias con la aplicacion -->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>1.3.1</version>
                <executions>
                    <execution>
                        <!-- Define cuando se ha de ejecutar el plugin
para que se lance cuando NetBeans construye el proyecto -->
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            <minimizeJar>true</minimizeJar>
                            <transformers>
                                <!-- This bit sets the main class for
the executable jar as you otherwise -->
                                <!-- would with the assembly plugin
-->
                                <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResour
ceTransformer">
                                    <manifestEntries>
                                        <Main-
Class>org.directgeoreference.App</Main-Class>
```

```
                                        <Specification-Title>Java
Advanced Imaging Image I/O Tools</Specification-Title>
                                        <Specification-
Version>1.1</Specification-Version>
                                        <Specification-Vendor>Sun
Microsystems, Inc.</Specification-Vendor>
                                        <Implementation-
Title>com.sun.media.imageio</Implementation-Title>
                                        <Implementation-
Version>1.1</Implementation-Version>
                                        <Implementation-Vendor>Sun
Microsystems, Inc.</Implementation-Vendor>
                                        <Extension-
Name>com.sun.media.imageio</Extension-Name>
                                    </manifestEntries>
                                </transformer>
                                <!-- This bit merges the various
GeoTools META-INF/services files         -->
                                <transformer
implementation="org.apache.maven.plugins.shade.resource.ServicesResour
ceTransformer"/>
                            </transformers>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
    <repositories>
        <repository>
            <id>maven2-repository.dev.java.net</id>
            <name>Java.net repository</name>
            <url>http://download.java.net/maven/2/</url>
        </repository>
        <!-- Repositorios de Geotools ... -->
    </repositories>

    <dependencies>
        <!-- Dependencia de Swing-layaout (Matisse de Netbeans) -->
        <dependency>
            <groupId>org.swinglabs</groupId>
            <artifactId>swing-layout</artifactId>
            <version>1.0.3</version>
        </dependency>
        <!-- Dependencia de Java Help -->
        <dependency>
            <groupId>javax.help</groupId>
            <artifactId>javahelp</artifactId>
            <version>2.0.05</version>
        </dependency>
    </dependencies>

    <!-- Otras dependencias de Geo Tools ...-->
</project>
```

REFERENCIAS

https://maven.apache.org/guides/getting-started/

https://www.jc-mouse.net/java/aplicaciones-java-con-maven

https://maven.apache.org/plugins/maven-compiler-plugin/examples/set-compiler-source-and-target.html

http://panamahitek.com/jar-ejecutable-con-dependencias-incluidas-en-java/

https://es.wikipedia.org/wiki/Maven

https://juandyb.com/crear-un-paquete-jar-ejecutable-con-las-dependencias-incluidas-usando-maven/

https://javiergarciaescobedo.es/programacion-en-java/29-trucos/517-generar-archivo-ejecutable-jar-para-aplicaciones-java-javafx-maven-netbeans

https://www.yoprogramo.com/2019/06/16/subir-a-maven-central-una-libreria-propia/