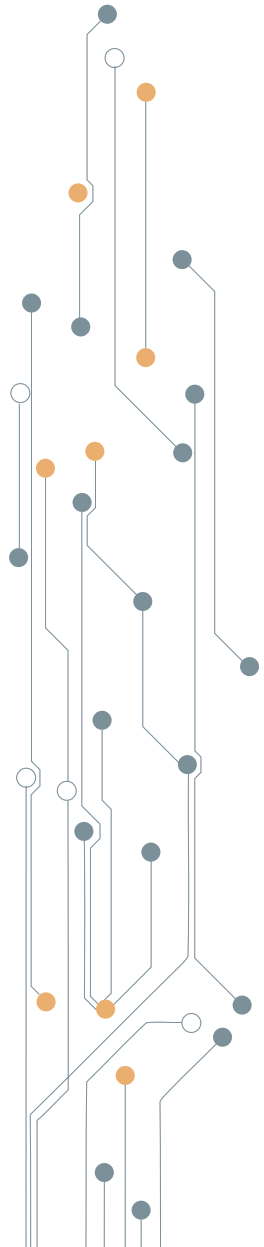
Paquetes y librerías de clases

Telefónica
EDUCACIÓN DIGITAL

Índice



Paquetes y librerías de clases

1. Organización de las clases en paquetes

En Java el paquete es un elemento de organización, que permite agrupar clases, interfaces, enumeraciones y otros paquetes, que tienen propósitos comunes.

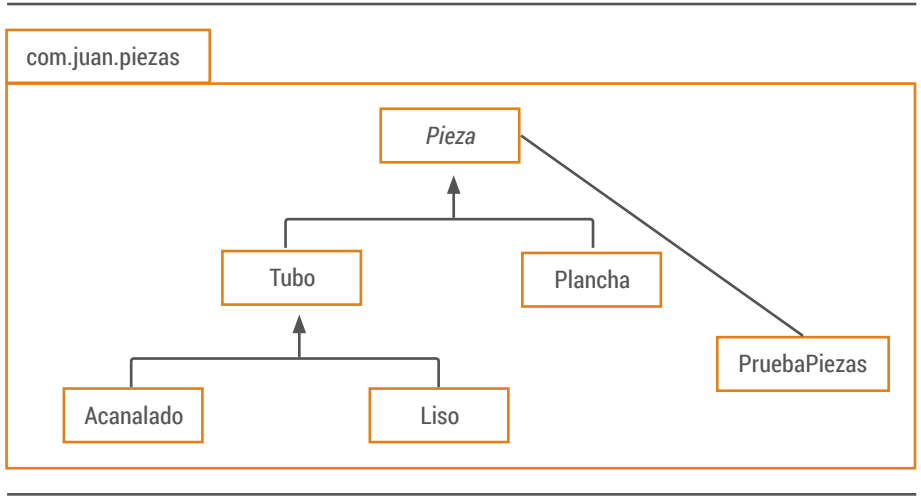


IMAGEN 7.1: ORGANIZACIÓN DE CLASES EN PAQUETES.

La palabra reservada para designar en un fichero de código fuente el paquete en el que esta la clase o clases o enumeraciones definidas en dicho fichero es **package**. Su sintaxis es la del formato siguiente:

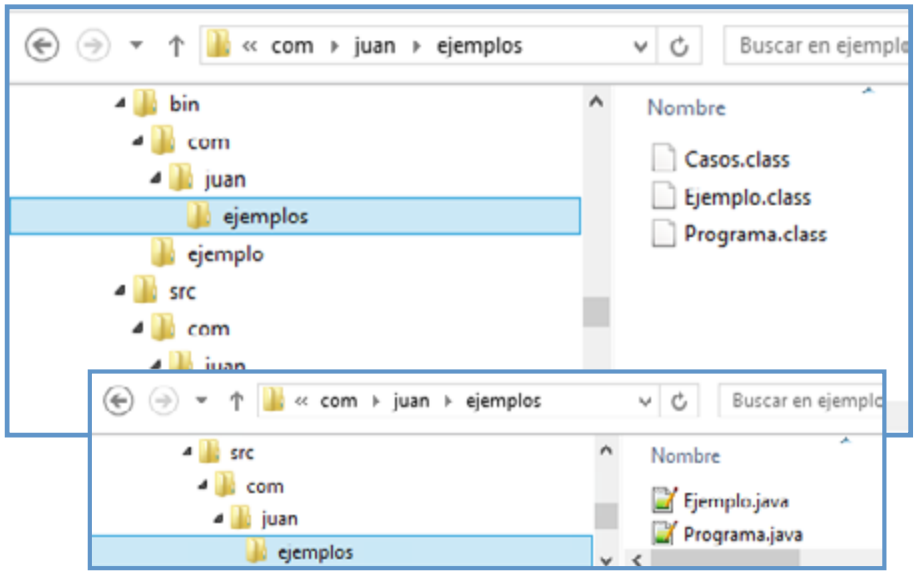
```
package <identificador_paquete>; //un paquete
package <identificador_paquete>[.<identificador_
paquete>]; //paquetes dentro
                    //de paquetes
```

En el ejemplo siguiente se muestra como los elementos definidos en dos archivos fuente están organizados en un paquete:

```
//archivo fuente Ejemplo.java
package com.juan.ejemplos;
//class Ejemplo y enum Casos en archivo Ejemplo.java
public class Ejemplo{
            //variables miembro
            //constructores
            //funciones miembro
}

enum Casos{
            UNO, DOS, TRES, CUATRO
}
//archivo fuente Programa.java
package com.juan.ejemplos;
//class Programa en archivo Programa.java
public class Programa{
            //variables miembro
            //constructores
            //funciones miembro
            public static void main(String [] args){
                        //código de la función main
            }
}
```

Los paquetes son carpetas del sistema de archivos. En los IDEs el paquete esta tanto en el directorio de los archivos fuente (**src**), como en el de los archivos binarios (**bin**), tal y como muestra la imagen.



2. Uso de import

En casi todos los códigos de las clases de Java se necesitan utilizar objetos, clases, interfaces y enumeraciones organizados en otros paquetes diferentes. Una clase puede utilizar todas las clases de su propio paquete además de todas las clases públicas de otros paquetes.

Para acceder a una clase pública de otro paquete, hay que añadir el nombre del paquete completo delante del nombre de la clase, todas las veces que se haga referencia a dicha clase.

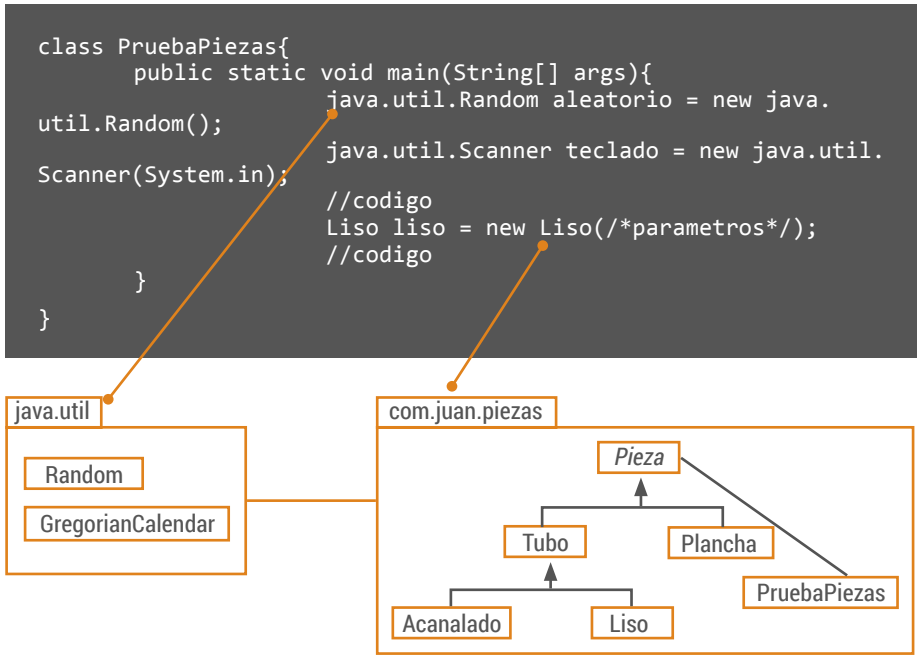


IMAGEN 7.3: UTILIZACIÓN DE CLASES DE OTROS PAQUETES.

La forma más sencilla y frecuente es utilizar la sentencia **import**. Una vez ejecutada esta sentencia ya no hay dar a la clase todo su nombre completo con la referencia al paquete en el que se encuentra.

La sintaxis de utilización de import tiene dos formas:

- Accediendo a la clase, interfaz o enumeración que se tiene que utilizar:

```
import <itinerario_paquete>.<identificador_elemento>;
```

- Accediendo a todos las clases, interfaces o enumeraciones de un paquete, excluyendo los paquetes que contuviera éste.

```
import <itinerario_paquete>.*;
```

La utilización de * (por ejemplo java.util.*) es menos tediosa, no tiene efectos negativos sobre el tamaño del código, pero la desventaja que no se sabe explícitamente que clases, interfaces o enumeraciones se importan.

```
import java.util.Scanner;
import java.util.Random;
//import java.util.*
class PruebaPiezas{
      public static void main(String[] args){
                  Random aleatorio = new Random();
                  Scanner teclado = new Scanner(System.in);
                  //codigo
                  Liso liso = new Liso(/*parametros*/);
                  //codigo
      }
}
```

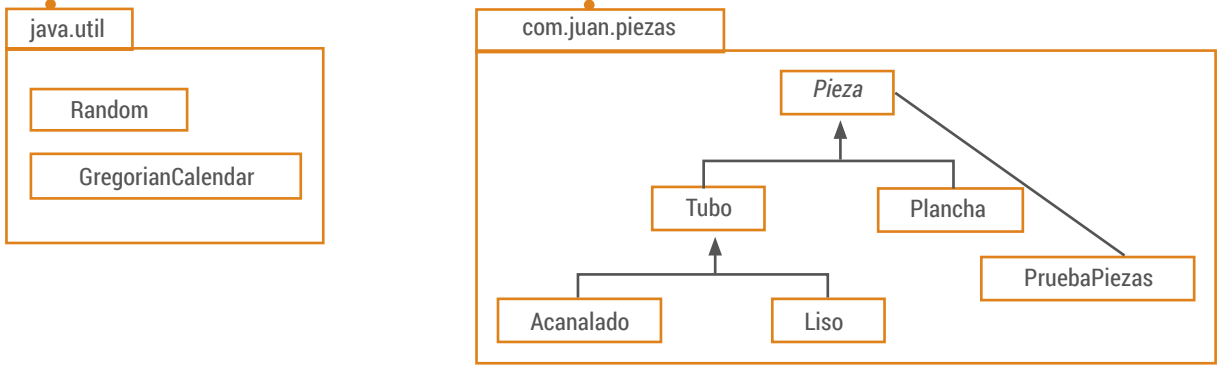


IMAGEN 7.3: UTILIZACIÓN DE IMPORT.

3. Estructura de librerías y paquetes de JSE

Cuando se instala JDK todas las librerías de JSE están disponibles en los archivos JAR instalados en el directorio lib del JRE correspondiente. En la imagen se muestra como desde un IDE como Eclipse, en las propiedades del proyecto se pueden localizar la carpeta con la librería JSE instalada.

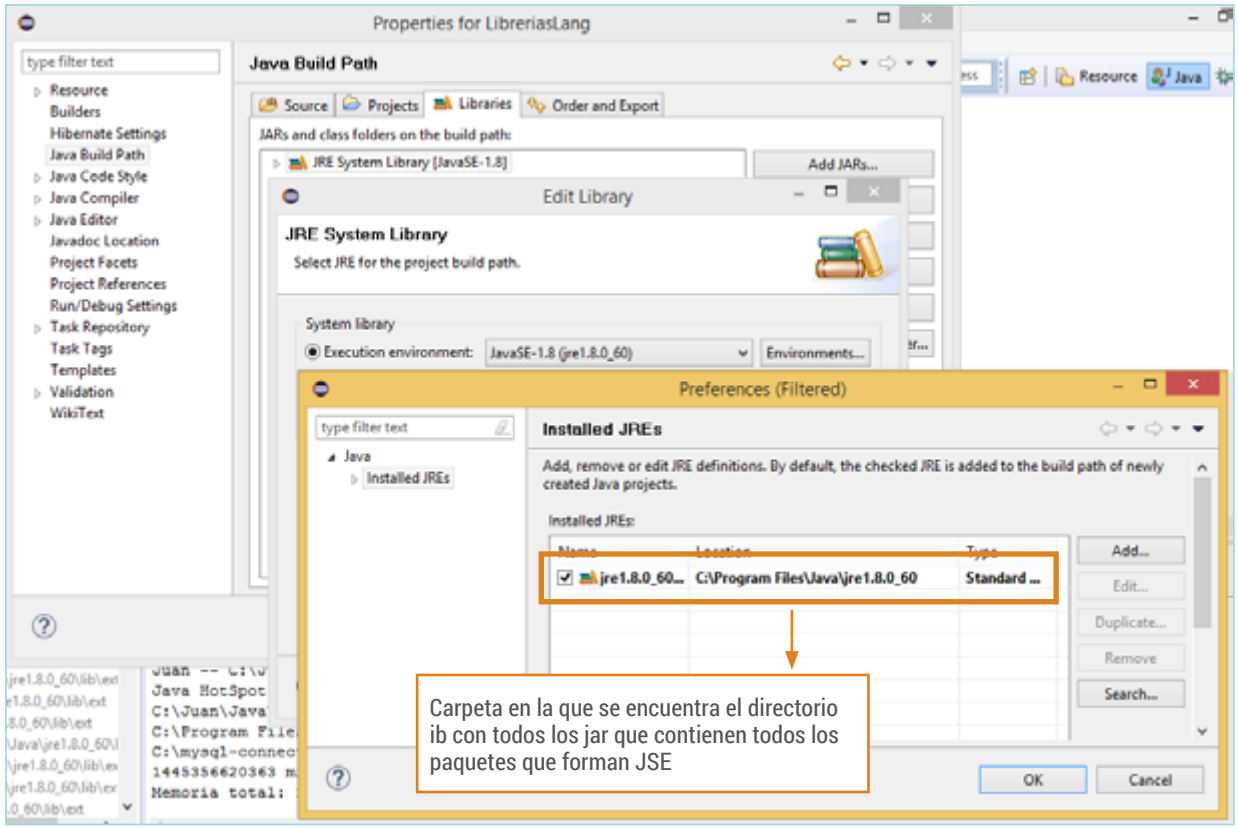


IMAGEN 7.4: DIRECTORIO LIBRERÍA JSE.

En el **directorio lib** están todos los **archivos jar** que componen la **librería JSE**. Un **jar** es un archivo comprimido que contiene el **bytecode** de las clases (los archivos **class**), cada una en su paquete correspondiente.

Por ejemplo en el archivo **rt.jar** están los paquetes más comunes de JSE. En la figura se muestra parte del contenido de este archivo, con el paquete **java.util**, en el que se encuentran entre otras las clase Random y Scanner, ya utilizadas.

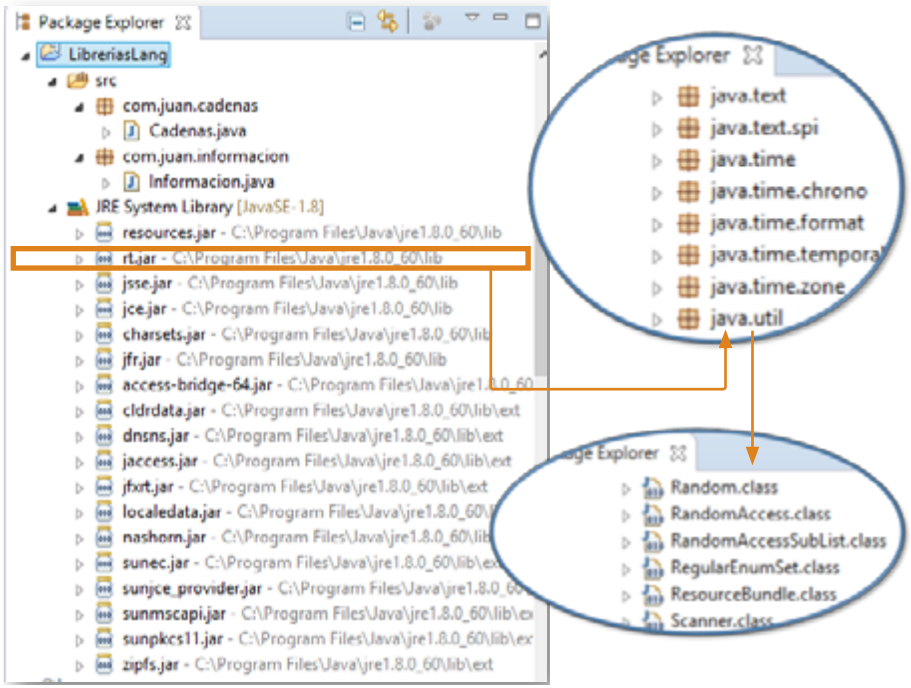


IMAGEN 7.5: CONTENIDO DEL ARCHIVO RT.JAR.

Cuando se ejecuta un código JAVA la máquina virtual buscará las clases de JSE en el directorio **lib**, porque la variable de entorno **JAVA_HOME** tiene el directorio padre:

```
JAVA_HOME = C:\Program Files\Java\jre1.8.0_60
```

Pero para indicar en qué carpeta están las clases que utiliza el programa que se ejecuta es necesario configurar la variable de entorno **CLASSPATH**. Esto se puede hacer de varias formas:

- Especificar el contenido de la variable, como se muestra:

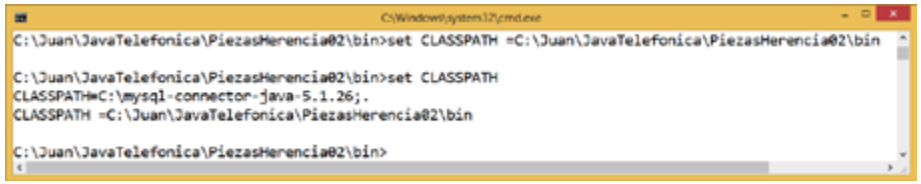


IMAGEN 7.5: VARIABLE DE ENTORNO CLASSPATH.

- Especificar cuando se ejecuta java (especificar a JVM) la opción -cp:

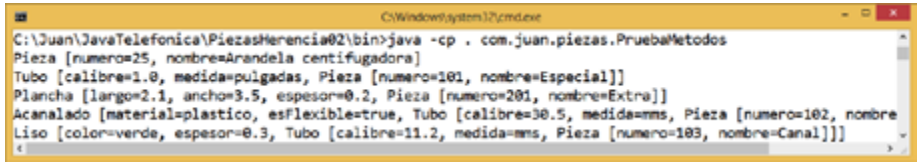


IMAGEN 7.6: ESPECIFICAR A JVM CUAL ES CLASSPATH.

4. java.lang

Proporciona clases que son fundamentales para la programación en Java. Se importa automáticamente, no es necesario poner en la clase donde se vaya a hacer uso de este paquete la sentencia "**import java.lang.***". Dentro de este paquete están gran parte de las clases más utilizadas dentro de los programas. Algunas de las clases de **java.lang** son las que se indican en la tabla siguiente.

<table>
<tr><th>Clases</th><th>Funcionalidades</th></tr>
<tr><td>Object</td><td>La clase raíz de toda la jerarquía de clases</td></tr>
<tr><td colspan="2">Todas las clases envoltorio: Character, Byte, Integer, Long, Float, Double, Boolean</td></tr>
<tr><td colspan="2">Las clases String para cadenas “inmutables” y StringBuffer para cadenas que pueden cambiar su tamaño y contenido, entre sus principales funciones están append, delete, insert, indexOf, replace, reverse.</td></tr>
<tr><td>System</td><td>Es una clase final, no puede tener subclases. Tiene constructor sin parámetros en ámbito private, por tanto no se pueden instanciar objetos. Todos sus miembros son static. Algunas de sus funcione son las que se indican a continuación.
<table>
<tr><th>Objetos</th><th>Funciones</th></tr>
<tr><td>in, out, err: encapsulan los flujos estándar de entrada, salida y salida para error</td><td>exit(int status): terminación de la ejecución del programa.
getenv(): devuelve una colección, en la que cada elemento es el nombre y el contenido de una variable de entorno.
currentTimeMillis(): devuelve el instante actual de tiempo en milisegundos.
gc(): ejecuta el garbage collector, es más recomendable utilizar el que se obtendrá desde la clase Runtime.
getProperty(String key): devuelve el String con el contenido de la propiedad que recibe como parámetro. En la ayuda del API de la función getProperties() esta la tabla de todas las propiedades.</td></tr>
</table></td></tr>
<tr><td>Runtime</td><td>Cada aplicación Java tiene una sola instancia de esta clase en tiempo de ejecución, que es la que permite a la aplicación interactuar con el entorno de ejecución. Tiene constructor sin parámetros en ámbito private, por tanto no se pueden instanciar objetos. Su método más importante es la función static getRuntime(), que retorna el objeto Runtime donde se ejecuta la aplicación. Con este objeto después se invocarán funciones de instancia, entre las que se encuentran las siguientes:

gc(): ejecuta el garbage collector, recomendado utilizar este porque en realidad es el que esta asociado a la aplicación que se esta ejecutando. System.gc() llama a este gc().

exit(int status): como la función de System, termina la ejecución del programa actual.
La diferencia esta en la forma de invocar a esta función: System.exit(n); o Runtime.getRuntime().exit(n);

maxMemory(), totalMemory() y freeMemory(), devuelven el long con la máxima memoria que podría utilizar JVM, la total que tiene a su disposición y la que queda libre.
La memoria usada es totalMemory() - freeMemory().</td></tr>
</table>

Tabla 7.1: Paquete java.lang.

5. java.util

Es junto con java.lang el paquete más utilizado de Java. Contiene las clases que implementan las colecciones en Java, manejo de fechas, y otras utilidades como números aleatorios, localización, operaciones con arrays y temporizador para utilizar en la programación multitarea.

La tabla siguiente muestra un resumen de estas clases.

Clases	Funcionalidad
Random	Para obtener números aleatorios, tanto enteros como de punto flotante.
Arrays	No tiene constructores, no se puede instanciar, todos sus métodos son static. La mayoría de las funciones están sobrecargadas para trabajar con arrays de los tipos primitivos. Algunas de sus funciones son: **binarySearch**, **fill** y **sort**
Date	Representa un "datetime" del instante actual. El único constructor que no esta obsoleto (decrepated) es el que crea un objeto que representa el instante actual.
Calendar	Tiene una gran cantidad de métodos para operar, consultar y modificar las propiedades de una fecha. Es una clase abstracta por lo que algunos de sus métodos deben ser implementados por sus subclases. Se suele instanciar con su método getInstance() el cual crea un objeto de la clase conteniendo la fecha del instante actual. Calendar ahora = Calendar.getInstance();
GregorianCalendar	Es una subclase de Calendar. Utiliza la representación horaria del calendario estándar mundial. Es la que se utiliza para trabajar con fechas, horas e instantes de tiempo, hasta antes de la versión 8 de JDK.
Locale	Representa una región geográfica, política o cultural específica. Los objetos creados son utilizados como parámetros en funciones de otras clases que puedan dar formato fechas, números, etc. Por ejemplo: DateFormat df = DateFormat.getDateInstance(DateFormat.LONG, new Locale("es"));
Colecciones	Listas: ArrayList, LinkedList, ... Conjuntos: **Set**, **HashSet**, **TreeSet**, ... Mapas (asocian un valor a una clave): **Maps**, **HashMap**, **Dictionary**, **Properties**, ...

Tabla 7.2: Paquete java.util.

6. Polimorfismo

La clase **Math** es **final**, no se puede heredar. Tiene constructor private y no se pueden instanciar objetos. Todos sus métodos al igual que sus dos datos (**E** y **PI**) son **static**. Algunas de sus funciones son las de la tabla siguiente:

Método	Funcionalidad
int abs (int num)	Valor absoluto de num
double acos (double num) **double asin (double** num) **double atan (double** num)	Arco coseno, arco seno y el arco tangente de num
double cos (double angulo) **double sin (double** angulo) **double tan (double** angulo)	Coseno, seno y tangente de angulo
double ceil (double num)	Devuelve el double sin decimales más cercano por arriba.
double exp (double pot)	Eleva el número e a una potencia.
double floor (double num)	Devuelve el double sin decimales más cercano por debajo.
double pow (double num, **double** potencia)	Elevar un número a una potencia
double random ()	Número aleatorio entre 0 (inclusive) y 1 (inclusive)
double sqrt(double num)	La raíz cuadrada de un número positivo.
<tipo_numero> max (<tipo_numero> n1, <tipo_numero> n2) **<tipo_numero> min (<tipo_numero> n1, <tipo_numero> n2)**	Devuelve el mayor o el menor de dos números dados. Esta sobrecargada para los tipos int, long, float y double

Tabla 7.3: Clase java.lang.Math.

Como todos los métodos son **static**, para usarlos, siempre habrá que escribir "Math.".Para no tener que escribir siempre el nombre de la clase, precediendo a la llamada de la función, la sentencia **import** tiene la posibilidad de usarla con **static**, en la forma siguiente:

```
import static <paquete>.<clase>;
import static <paquete>.<clase>.*;
```

Un ejemplo de su uso es el siguiente:

```
package com.juan.importestatico;
import static java.lang.System.*;
import static java.lang.Math.*;
import java.util.Scanner;
public class ImportEstatico {
      public static void main(String[] args) {
            Scanner teclado= new Scanner(in);
            double d = teclado.nextDouble();
            out.println("La raíz de " + d + " es " + sqrt(d));
            out.println("Redondeado a menor " + d + " es " + floor(d));
            out.println("Redondeado a mayor " + d + " es " + ceil(d));
            out.println("Rango de byte es: "+
                              MAX_VALUE + " a " + MIN_VALUE);
      }
}
```

Telefónica

EDUCACIÓN DIGITAL