



# ORACLE

## Academy



# Java Foundations

4-2

Declaración `import` y paquetes

**ORACLE**  
Academy



# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Acceder a una clase usando su nombre cualificado completo
  - Describir la función de la sentencia `import`
  - Usar la sentencia `import` para acceder a la clase de un paquete
  - Comprender el objetivo de un asterisco en una sentencia `import`
  - Identificar los paquetes que se importan automáticamente



# ¿Por qué debe reinventar la rueda?

- Con frecuencia, es posible que escriba el mismo código Java para diferentes programas
- Como alternativa a volver a escribir el mismo código, puede utilizar la biblioteca proporcionada por Java, que organiza el código utilizado con frecuencia
- Esta biblioteca se denomina biblioteca de clases Java
- La documentación de la biblioteca de clases Java está disponible aquí:

– <https://docs.oracle.com/en/java/javase/17/docs/api/java.base/module-summary.html>

# Paquetes de la biblioteca de clases Java

- Las clases de la biblioteca de clases Java se organizan por paquetes
- Un paquete contiene un grupo de clases relacionadas
- Con un paquete resulta más fácil localizar las clases relacionadas

# Paquetes de la biblioteca de clases Java

Cuerpo	Objetivo
<code>java.lang</code>	Proporciona clases que son fundamentales para el diseño del lenguaje Java
<code>javax.swing</code>	Proporciona clases para crear componentes de GUI
<code>java.net</code>	Proporciona clases para aplicaciones de red
<code>java.time</code>	Proporciona clases para fechas, horas, instantes y duraciones

# ¿Cómo se organizan los paquetes?


- La inmensa mayoría de la recopilación de clases se organiza en una jerarquía tipo árbol, que permite dividir los paquetes en subpaquetes, del siguiente modo:





# Tutorial de Javadoc

- En Oracle Academy Education Byte – Java – Hands on Lab :
  - Acceda al Java API Documentation (Javadoc) Tutorial y realízelo
  - <https://docs.oracle.com/en/java/javase/15/docs/api/index.html>



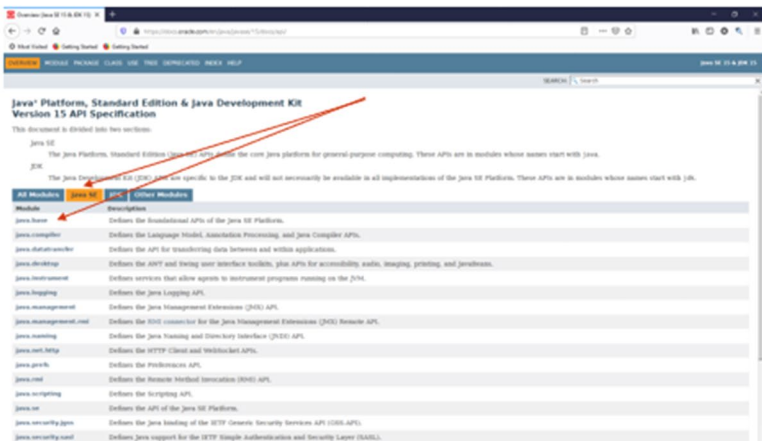
academy.oracle.com

## Javadoc Tutorial

How to access and use the documentation for the Java SE Development Kit, or JDK - Versions 9 and later – for this tutorial, we will use JDK Version 15

Topic	Details
Overview	In this tutorial, you will become familiar with the basic features of the documentation for the JAVA SE Development Kit, or JDK.
Key Concepts	<ul style="list-style-type: none"> <li>Access the Javadocs API</li> <li>Explore Modules</li> <li>Explore Packages, Classes, Constructors, and Methods</li> <li>Utilize the Search feature of Javadocs</li> </ul>
Difficulty	Beginner – This tutorial is appropriate for someone learning Java
Duration	30 minutes
Notes	This tutorial was built using JDK Version 15

- Access and bookmark this link:
  - Java Platform, Standard Edition & JDK Version 15 API Specification: <https://docs.oracle.com/en/java/javase/15/docs/api/index.html>
- documentation we will use in this course is under Java SE – Select the Java SE tab
- the classes are grouped by module – in this course, you will be working with classes in the java.base module – Select the java.base link



Overview

This document is divided into two sections:

- JDK**
  - The Java Platform, Standard Edition (JSE) APIs are used by the core Java platform for general-purpose computing. These APIs are in modules whose names start with `java`.
- JDK**
  - The Java Technology Extension (JTE) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

**All Modules** | **Index** | **API** | **Other Modules**

Module	Description
<a href="#">java.base</a>	Defines the foundational APIs of the Java SE Platform.
<a href="#">java.compiler</a>	Defines the Language Model, Annotation Processing, and Java Compiler APIs.
<a href="#">java.datatransfer</a>	Defines the API for transferring data between and within applications.
<a href="#">java.desktop</a>	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaFX.
<a href="#">java.instrument</a>	Defines services that allow agents to instrument programs running on the JVM.
<a href="#">java.logging</a>	Defines the Java Logging API.
<a href="#">java.management</a>	Defines the Java Management Extensions (JMX) APIs.
<a href="#">java.management.rmi</a>	Defines the RMI support for the Java Management Extensions (JMX) Remote API.
<a href="#">java.naming</a>	Defines the Java Naming and Directory Interface (JNDI) APIs.
<a href="#">java.net.http</a>	Defines the HTTP Client and Websocket APIs.
<a href="#">java.prefs</a>	Defines the Preferences API.
<a href="#">java.rmi</a>	Defines the Remote Method Invocation (RMI) APIs.
<a href="#">java.scripting</a>	Defines the Scripting API.
<a href="#">java.se</a>	Defines the API of the Java SE Platform.
<a href="#">java.security.jgss</a>	Defines the Java binding of the JGSS (Generic Security Services API) (GSS-API).
<a href="#">java.security.sasl</a>	Defines Java support for the SASL (Simple Authentication and Security Layer) (SASL).



# Uso de una clase de un paquete

- Para utilizar una clase de un paquete en su programa, debe especificar su nombre cualificado completo
- Por ejemplo, para usar la clase `Scanner` para leer una entrada de teclado, el nombre cualificado completo de la clase `Scanner`, que se define en el paquete `java.util`, es

`java.util.Scanner`



**Cuerpo**      **Nombre de clase**

# Uso del nombre de clase cualificado completo

- Como puede ver, el uso de un nombre cualificado completo crea nombres muy largos para las clases
- Los nombres largos reducen la legibilidad del código y también dificultan la codificación

```
public static void main(String[] args) { Nombre de clase cualificado completo
    int num;
    java.util.Scanner keyboard = new java.util.Scanner(System.in);
    System.out.print("Enter a number");
    num = keyboard.nextInt();
    System.out.print("The number you entered is" + num);
} //end method main
```

# ¿Existe una alternativa al nombre cualificado completo?

- Suponga que tiene un amigo cuyo nombre es Santi Inez Luis Vidal
  - ¿A que resulta tedioso llamarlo por su nombre completo cada vez?
  - Si pudiera referirse a él como “Santi”, resultaría mucho más cómodo
- Del mismo modo, acceder a las clases Java utilizando nombres cualificados completos es igual de tedioso en los programas
- Veamos si hay alguna forma de especificar únicamente el nombre de la clase en vez de un nombre cualificado completo

# Uso de la sentencia import

- Puede evitar el nombre de clase cualificado completo mediante la sentencia `import`
- Incluya la sentencia `import` sobre la definición de clase Tiene el siguiente aspecto:
  - `import package.className`
  - Ejemplo:

```
import java.util.Scanner;

public class AddNums {
    //class code goes here
} //end class AddNums
```

Cuerpo

Nombre de clase

Sentencia import

Definición de clase

# ¿Cómo se importa una sola clase?

- Puede importar una única clase o un paquete completo
- Para importar una única clase en su programa, escriba una sentencia `import` como la siguiente:

```
import javax.swing.JOptionPane;
```

Nombre de Paquete

Nombre de clase

Palabra clave `import`  
seguida del paquete,  
el punto y el nombre  
de la clase

# Paquete `javax.swing`

- Java tiene una amplia biblioteca para crear diferentes GUI
- Esta biblioteca, denominada `swing`, se puede importar a su programa Java para ofrecer acceso a la funcionalidad de GUI de Java
- La biblioteca `swing` está en el paquete `javax.swing`

# Acceso a una clase del paquete `swing`

- El paquete `swing` tiene una clase `JOptionPane`
- Esta clase crea una ventana emergente que se puede utilizar para mostrar las cadenas de texto al usuario
- Para utilizar la clase `JOptionPane`, primero debe importarla a su clase:

```
import javax.swing.JOptionPane;

public class Welcome {
    //class code to go here
} //end class Welcome
```

Sentencia `import` que importa una sola clase, `JOptionPane`, del paquete `swing`



# Importación de la clase JOptionPane

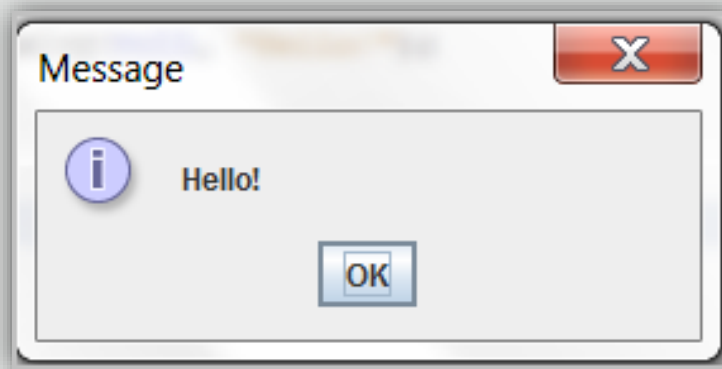
- Puede usar JOptionPane para mostrar texto llamando al método showMessageDialog en la clase JOptionPane

```
import javax.swing.JOptionPane;

public class Welcome {

    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Hello!");
    } //end method main
} //end class Welcome
```

# La salida tiene este aspecto



# ¿Cómo importar todas las clases de un paquete?

- Puede importar todas las clases de un determinado paquete mediante el carácter comodín `*` de la sentencia `import`

# ¿Cómo importar todas las clases de un paquete?

- Suponga que desea ampliar el ejemplo anterior creando una instancia de la clase `JFrame` y agrega su referencia a `JOptionPane`, del siguiente modo:

```
import javax.swing.JOptionPane;
import javax.swing.JFrame;

public class Welcome {

    public static void main(String[] args) {
        JFrame frame = new JFrame();
        JOptionPane.showMessageDialog(frame, "Hello!");
    } //end method main
} //end class Welcome
```

Importación de 2 clases del  
paquete `swing`

# Acceso a todas las clases del paquete `swing`

- A medida que acceda a más clases del paquete `swing` en el programa, también aumenta el número de sentencias `import`

# Acceso a todas las clases del paquete `swing`

- Para evitarlo, puede importar todas las clases del paquete `swing` mediante el carácter comodín `*` en la sentencia `import`, del siguiente modo:

```
import javax.swing.*;
```

Sustituya todas las sentencias `import` del eje anterior por una sola sentencia `import`

```
public class Welcome {  
    public static void main(String[] args) {  
        JFrame frame = new JFrame();  
        JOptionPane.showMessageDialog(frame, "Hello!");  
    } //end method main  
} //end class Welcome
```

# Inclusión de varias sentencias `import`

- Puede incluir varias sentencias `import` en un programa Java para acceder a las clases del mismo paquete o de diferentes paquetes
- Por ejemplo:

```
import java.util.Date;  
import java.util.Calendar; }  
import javax.swing.*;
```

```
public class DisplayDate {  
    //class definition here  
} //end class DisplayDate
```

Importación de clases del mismo paquete

Importación de clases de diferentes paquetes



# Identificación de los paquetes que se importan automáticamente

- Hasta el momento, ha usado `System.out.println()` para imprimir texto en la consola

```
public class DisplayOutput {  
  
    public static void main(String[] args) {  
        System.out.println("Hello, how are you today?");  
    } //end method main  
} //end class DisplayOutput
```

- No obstante, no ha importado un paquete con este método en el programa
- ¿Cómo sabe Java lo que debe hacer cuando se le llama?

# Paquete `java.lang`

- Si examina la biblioteca Java, verá que la clase `System` está organizada en el paquete `java.lang`
- Por defecto, el paquete `java.lang` se importa automáticamente en todos los programas Java

# Ejercicio 1

- Cree un nuevo proyecto y agréguele el archivo `AddImport.java`
- Examine `AddImport.java`
  - Realice lo siguiente:
  - Sustituir el nombre cualificado completo para acceder al componente `JLabel` con una sentencia `import`
  - Para importar las clases del paquete `util`, sustituir varias sentencias `import` por una sola sentencia `import`

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Acceder a una clase usando su nombre cualificado completo
  - Describir la función de la sentencia `import`
  - Usar la sentencia `import` para acceder a la clase de un paquete
  - Comprender el objetivo de un asterisco en una sentencia `import`
  - Identificar los paquetes que se importan automáticamente





# ORACLE

## Academy

