Fundamentos de Java

Paquetes más utilizados del API de Java

Por Ing. Eudris Cabrera Rodríguez

API

Una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Una API representa una interfaz de comunicación entre componentes software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software.

Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla.

De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio. Las APIs asimismo son abstractas: el software que proporciona una cierta API generalmente es llamado la implementación
de esa API.

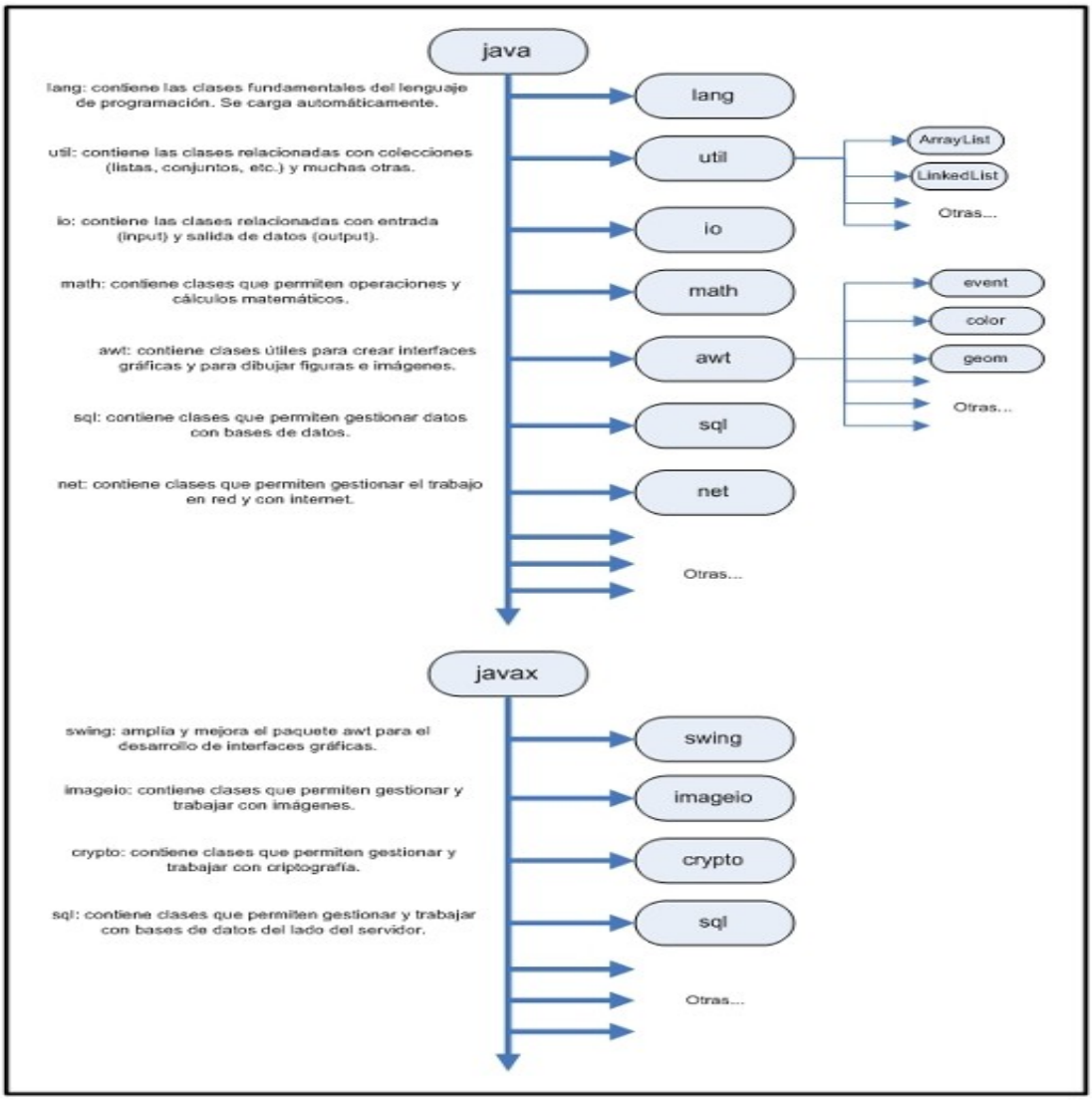
Es común que los programadores dispongan de clases desarrolladas por ellos mismos para ser utilizada en distintos proyectos.

En empresas grandes y hasta en proyectos personales, es frecuente disponer de clases o conjunto de clases (API) desarrolladas por otras empresas o desarrolladores, donde se conoce su interfaz pero no su implementación.

Trabajar con una clase sin ver su código fuente requiere que exista una **buena documentación** que nos sirva de guía.

API Java

La API Java es provista por los creadores del lenguaje de programación Java, ofrece a los programadores los medios para desarrollar aplicaciones Java.



Como el lenguaje Java es un lenguaje orientado a objetos, la API de Java provee un conjunto de clases utilitarias para efectuar toda clase de tareas necesarias dentro de un programa.

La API Java está organizada en paquetes lógicos, donde cada paquete contiene un conjunto de clases relacionadas semánticamente.

La figura anterior trata de mostrar la organización del API de Java de forma resumida.

Los nombres de las librerías responden a este esquema jerárquico y se basan en la notación de punto.

Por ejemplo el nombre completo para la clase **ArrayList** sería **java.util.ArrayList.**
Se permite el uso de ***** para nombrar a un conjunto de clases.
Por ejemplo **java.util.*** hace referencia al conjunto de clases dentro del paquete **java.util**,

donde tenemos **ArrayList, LinkedList** y otras clases.

Para utilizar las librerías del API, existen dos situaciones:

a) Hay librerías o clases que se usan siempre pues constituyen elementos fundamentales del lenguaje Java como la clase **String**.
Esta clase, perteneciente al paquete **java.lang**, se puede utilizar directamente en cualquier programa Java ya que se carga automáticamente.

b) Hay librerías o clases que no siempre se usan. Para usarlas dentro de nuestro código hemos de indicar que requerimos su carga mediante una sentencia **import** incluida en cabecera de clase.

Por ejemplo **import java.util.ArrayList;** es una sentencia que incluida en cabecera de una clase nos permite usar la clase **ArrayList** del API de Java.

Escribir **import java.util.*;** nos permitiría cargar todas las clases del paquete **java.util.**
Algunos paquetes tienen decenas o cientos de clases. Por ello nosotros preferimos en general especificar las clases antes que usar asteriscos ya que evita la carga en memoria de clases que no vamos a usar.

Una clase importada se puede usar de la misma manera que si fuera una clase generada por nosotros: podemos crear objetos de esa clase y llamar a métodos para operar sobre esos objetos.

Para programar en Java tendremos que recurrir continuamente a consultar la documentación del API de Java. Esta documentación está disponible en cds de libros y revistas especializadas o en internet tecleando en un buscador como google o bing el texto **"api java 8"** (o **"api java 6"**, **"api java 10"**, etc.) según la versión que estés utilizando.

La documentación del API de Java en general es correcta y completa. Sin embargo, en casos excepcionales puede estar incompleta o contener erratas.

Podemos obtener información del API completo de la versión actual **javase 11** en https://docs.oracle.com/en/java/javase/11/docs/api/index.html.

Para fines de estudio nos limitaremos a mencionar los más utilizados.

Paquetes de Propósito General

java.lang

El paquete **java.lang** contiene clases fundamentales e interfaces fuertemente relacionadas con el lenguaje y el sistema runtime. Esto incluye las clases raíz que forman la jerarquía de clases, tipos relacionados con la definición del lenguaje, excepciones básicas, funciones matemáticas, Hilos, funciones de seguridad, así como también alguna información sobre el sistema nativo subyacente.

Las principales clases en java.lang son:

- **Object** – la clase que es la raíz de toda la jerarquía de clases.
- **Enum** – la clase base para las clases enumeration.
- **Class** – la clase que es la raíz del sistema de reflexión Java.
- **Throwable** – la clase que es la clase base de la jerarquía de clases de excepciones.
- **Error**, **Exception**, y **RuntimeException** – las clases base de cada tipo de excepción.
- **Thread–** la clase que permite operaciones con hilos.
- **String** – la clase para cadenas String y literales.
- **StringBuffer** y **StringBuilder** – clases para realizar manipulación de cadena de caracteres.
- **Comparable** – la interfaz que permite comparación genérica y ordenamiento de objetos.
- **Iterable** – la interfaz que permite iteración genérica usando el bucle for mejorado.
- **ClassLoader**, **Process**, **Runtime**,**SecurityManager**, y **System** – clases que suministran "operaciones del sistema" que gestionan la enlazado dinámico de clases, creación de procesos externos, investigaciones del entorno del host tales como la hora del día, y refuerzo de políticas de seguridad.
- Las clases **Math** y **StrictMath** – suministran funciones matemáticas básicas tales como seno, coseno, y raíz cuadrada.
- Las clases **wrapper** primitivas encapsulan tipos primitivos como objetos.
- Las clases exception básicas lanzan excepciones del nivel del lenguaje y otras excepciones comunes.

Las clases de **java.lang** son importadas automáticamente a cada Fichero fuente.

java.io

El paquete java.io contiene clases que soportan entrada/salida. Las clases del paquete son principalmente streams; sin embargo, se incluye una clase para archivo de acceso aleatorio. Las clases centrales del paquete son **InputStream** y **OutputStream,** las cuales son clases abstractas, base para leer y escribir a streams de bytes, respectivamente.

Las clases relacionadas **Reader** y **Writer** son clases abstractas base para leer de y escribir a streams de caracteres, respectivamente. El paquete también tiene unas pocas clases misceláneas para soportar la interacción con el Sistema de archivos del computador.

java.nio

El paquete java.nio (NIO o Nueva I/O) fue añadido para soportar I/O mapeada en memoria, facilitando las operaciones I/O cercanas al hardware subyacente con mejor rendimiento.

El paquete **java.nio** suministra soporte para varios tipos de buffer.

El subpaquete **java/nio/charset** suministra soporte para distintas codificaciones de caracteres para datos de tipo carácter.

El subpaquete **java/nio/channels** suministra soporte para **channels**, los cuales representan conexiones a entidades que son capaces de realizar operaciones I/O, tales como archivos y sockets. El paquete **java.nio.channels** también suministra soporte para bloqueo a archivos.

java.math

El paquete **java.math** soporta aritmética multi precisión (incluyendo operaciones aritméticas modulares) y suministra generadores de números primos multi precisión usados para la generación de claves criptográficas. Las clases principales de este paquete son:

- **BigDecimal** – suministra números decimales con signo de precisión arbitraria. **BigDecimal** da al usuario el control sobre el comportamiento de redondeo a través de **RoundingMode**.
- **BigInteger** – suministra enteros de precisión arbitraria. Las Operaciones con **BigInteger** no producen overflow o pérdida de precisión. Además de las operaciones aritméticas estándar, suministra aritmética modular, cálculo de mínimo común múltiplo, pruebas de números primos, generación de números primos, manipulación de bits, y otras operaciones misceláneas.
- **MathContext** – encapsula las configuraciones de contexto las cuales describen ciertas reglas para operadores numéricos.
- **RoundingMode** – una enumeración que suministra ocho comportamientos de redondeo.

java.net

El paquete java.net suministra rutinas especiales IO para redes, permitiendo las peticiones HTTP, así como también otras transacciones comunes.

java.text

El paquete **java.text** proporciona clases e interfaces para el manejo de texto, fechas, números y mensajes de una manera independiente de los lenguajes naturales

java.util

En el paquete está incluida la **API Collections**, una jerarquía organizada de estructura de datos influenciada fuertemente por consideraciones de patrones de diseño. Clases para el manejo de fecha y tiempo y un amplio conjunto de clases utilitarias.

Paquetes de propósito especial

java.applet

Creado para soportar la creación de applet Java, el paquete java.applet permite a las aplicaciones ser descargadas sobre una red y ejecutarse dentro de una sandbox.

Las restricciones de seguridad son impuestas fácilmente en la sandbox. Un desarrollador, por ejemplo, puede aplicar una firma digital a un applet, en consecuencia etiquetando como segura.

Haciéndolo permite al usuario conceder permiso al applet para realizar operaciones restringidas (tales como acceder al disco duro local), y elimina alguna o todas las restricciones de la sandbox.

java.beans

Incluidos en el paquete java.beans hay varias clases para desarrollar y manipular beans, componentes reutilizables definidos por la arquitectura **JavaBeans**.

La arquitectura suministra mecanismos para manipular propiedades de componentes y lanzar eventos cuando esas propiedades cambian.

java.awt

La Abstract Window Toolkit contiene rutinas para soportar operaciones básicas GUI y utiliza ventanas básicas desde el sistema nativo subyacente. Muchas implementaciones independientes de la API Java implementan todo excepto AWT, el cual no es usado por la mayoría de las aplicaciones de lado de servidor.

Este paquete también contiene la API de gráficos Java 2D.

java.rmi

El paquete **java.rmi** suministra invocación a métodos remotos Java para soportar llamadas a procedimientos remotos entre dos aplicaciones Java que se ejecutan en diferentes JVM.

java.security

Soporte para seguridad, incluyendo el algoritmo de resumen de mensaje, está incluido en el paquete java.security.

java.sql

Una implementación de la API JDBC (usada para acceder a bases de datos SQL) se agrupa en el paquete java.sql.

javax.rmi

suministra el soporte para la comunicación remota entre aplicaciones, usando el protocolo RMI sobre IIOP. Este protocolo combina características de RMI y CORBA.

org.omg.CORBA

Suministra el soporte para comunicación remota entre aplicaciones usando general inter ORB protocol y soporta otras características de common object request broker architecture. Igual que RMI y RMI-IIOP, este paquete es para llamar métodos remotos de objetos en otras máquinas virtuales (normalmente por la red).

De todas las posibilidades de comunicación CORBA es la más portable entre varios lenguajes. Sin embargo es también un poco difícil de comprender.

javax.swing

Swing es una colección de rutinas que se construyen sobre **java.awt** para suministrar un toolkit de widgets independiente de plataforma.

Swing usa las rutinas de dibujado 2D para renderizar los componentes de interfaz de usuario en lugar de confiar en el soporte GUI nativo subyacente del Sistema operativo.

Swing es un sistema muy rico por sí mismo, soportando pluggable looks and feels (PLAFs) para que los controles(widgets) en la GUI puedan imitar a aquellos del sistema nativo subyacente.

Los patrones de diseño impregnan el sistema, especialmente una modificación del patrón modelo-vista-controlador, el cual afloja el acoplamiento entre función y apariencia.

Una inconsistencia es que las fuentes son dibujadas por el sistema nativo subyacente, limitando la portabilidad de texto. Mejoras, tales como usar fuentes de mapas de bits, existen.

En general, las layouts(disposiciones de elementos) se usan y mantienen los elementos dentro de una GUI consistente a través de distintas plataformas.

javax.swing.text.html.parser

Suministra el parser de HTML tolerante a errores que se usa para escribir varios navegadores web y web bots.

java.time

La API principal de fechas, horas, instantes, y duraciones.

Eudris Cabrera Rodríguez
Ingeniero Telemático
Consultor / Desarrollador Informático
LinkedIn: http://www.linkedin.com/in/eudriscabrera
Revisado Noviembre 2018, Santiago de los Caballeros, R. D.