La evolución de las bibliotecas de **Java** ha sido crucial para su crecimiento y éxito como lenguaje de programación. Desde su creación, las bibliotecas en Java han evolucionado en términos de funcionalidad, alcance y especialización, lo que ha permitido a los desarrolladores crear aplicaciones robustas y seguras sin tener que reinventar la rueda en cada proyecto.

1. Evolución de las Bibliotecas de Java

Desde su lanzamiento en 1995, Java ha desarrollado un extenso ecosistema de bibliotecas estándar, conocidas como el **Java Standard Library** o **API de Java** (Application Programming Interface). Estas bibliotecas han ido creciendo con cada nueva versión de Java, incorporando características para hacer frente a nuevos desafíos de desarrollo. Algunas etapas importantes de su evolución incluyen:

- **Java 1.0 y 1.1**: Las primeras versiones ofrecían bibliotecas básicas para entrada/salida (I/O), gestión de hilos (threads), manipulación de cadenas (Strings), y gráficos mediante AWT (Abstract Window Toolkit).
- **Java 2 (JDK 1.2)**: Introducción de **Swing** (una biblioteca avanzada para interfaces gráficas), **JDBC** (Java Database Connectivity) para bases de datos, **RMI** (Remote Method Invocation), y un marco de trabajo para colecciones (Collection Framework).
- **Java 5 (JDK 1.5)**: Incorporación de **genéricos**, **autoboxing**, **annotations**, y un gran impulso a la programación concurrente con la **java.util.concurrent**.
- **Java 8**: Introducción de las **expresiones lambda** y el soporte para **Stream API**, lo que mejoró significativamente la programación funcional y el procesamiento de grandes conjuntos de datos.
- **Java 9 en adelante**: Modularización del JDK con el **Proyecto Jigsaw** y la creación de **Jigsaw Modules**, facilitando la creación de aplicaciones más ligeras y especializadas.

Con el tiempo, se ha ampliado la **API estándar de Java** (JDK), cubriendo cada vez más áreas como redes, seguridad, manipulación de fechas, criptografía, gráficos 2D/3D, servicios web, multihilos y más.

2. ¿Se sigue hablando de rutinas?

El término **"rutina"** fue más común en lenguajes más antiguos o de bajo nivel (como C o ensamblador) para referirse a bloques de código que realizan una tarea específica. Hoy en día, este término ha sido sustituido por conceptos más modernos como **métodos**, **funciones** o **procedimientos** en lenguajes de alto nivel como Java.

En Java, en lugar de hablar de "rutinas", hablamos de:

- **Métodos**: Son las unidades básicas de trabajo en una clase. Un método es una función que puede realizar una acción y devolver un valor.
- **Clases y Objetos**: Estos son los bloques fundamentales en los lenguajes orientados a objetos. Los métodos y atributos se agrupan dentro de clases que modelan el comportamiento y el estado de los objetos.

Aunque el concepto de "rutina" todavía existe en su sentido general (como un fragmento de código reutilizable), en el contexto moderno de Java y la programación orientada a objetos, nos referimos más a **métodos** y **API**.

3. ¿Qué es la API de Java? ¿Es la evolución de las rutinas?

La **API de Java (Application Programming Interface)** es el conjunto completo de bibliotecas, clases, interfaces y paquetes que los desarrolladores pueden usar para programar en Java. Podría considerarse la evolución de las "rutinas" en el sentido de que proporciona un conjunto de herramientas predefinidas que realizan funciones comunes, pero mucho más estructuradas y moduladas.

La API de Java incluye todo lo necesario para desarrollar aplicaciones en casi cualquier campo, desde sistemas básicos hasta aplicaciones web avanzadas, con módulos preconstruidos para:

- **Entrada/Salida (I/O)**: Leer y escribir archivos.
- **Colecciones**: Manipulación de conjuntos de datos (listas, colas, pilas, etc.).
- **Interfaz Gráfica (GUI)**: Creación de interfaces gráficas con Swing, AWT y JavaFX.
- **Redes**: Programación de sockets y servicios en red.
- **Seguridad**: Encriptación, manejo de certificados, autenticación.
- **Concurrencia**: Gestión de hilos y tareas paralelas.
- **Bases de Datos**: Conexión y operaciones sobre bases de datos (JDBC).
- **Streams y Funciones Lambda**: Manipulación eficiente de colecciones de datos en Java 8 y posteriores.

Evolución de las "rutinas" a la **API de Java**:

En lugar de trabajar con "rutinas" individuales como en los lenguajes más antiguos, la API de Java encapsula estas "rutinas" dentro de clases y métodos organizados en paquetes. Cada método tiene una función específica, y los desarrolladores pueden invocar estos métodos para realizar

acciones complejas, como leer un archivo, manejar excepciones o dibujar gráficos.

4. ¿Qué debo conocer sobre la API de Java?

Para trabajar eficientemente con la API de Java, es importante comprender los siguientes conceptos:

a. Paquetes y módulos:

- Los paquetes son colecciones de clases relacionadas. La API de Java está organizada en paquetes, como `java.util` (colecciones y utilidades), `java.io` (entrada/salida), `java.nio` (I/O no bloqueante) y `java.net` (redes).
- Los **módulos** (introducidos en Java 9) son un conjunto de paquetes agrupados para dividir la API en partes más manejables y modulares.

b. Clases y Métodos:

- Una **clase** es una plantilla o blueprint que define los atributos y comportamientos de un tipo particular de objeto.
- Un **método** es una función definida dentro de una clase que realiza una tarea específica. Muchos métodos en la API de Java son reutilizables para resolver problemas comunes.

c. Documentación de la API de Java:

- Java proporciona una **documentación oficial** muy detallada para cada clase, método e interfaz dentro de su API. Es fundamental aprender a consultar y navegar esta documentación para entender cómo usar las herramientas que Java ofrece.
- La documentación te muestra qué parámetros recibe un método, qué devuelve, y qué excepciones puede lanzar.

d. Uso de bibliotecas adicionales:

- Aparte de la API estándar, hay una gran cantidad de bibliotecas de terceros que extienden Java. Por ejemplo, **Apache Commons** ofrece soluciones a problemas comunes, y **Spring Framework** es ampliamente utilizado para el desarrollo web.

5. Conclusión: ¿Es la API de Java la evolución de las rutinas?

Podríamos decir que la **API de Java** es la evolución de las "rutinas" porque encapsula todas las funcionalidades comunes en una estructura ordenada, modular y orientada a objetos. Donde antes se usaban "rutinas" sueltas en lenguajes de bajo nivel, en Java utilizamos métodos bien definidos dentro de clases y bibliotecas para realizar tareas complejas de manera eficiente.

Conocer la API de Java es fundamental para desarrollar en este lenguaje. Familiarizarse con los paquetes clave y aprender a leer la documentación oficial es esencial para aprovechar al máximo todas las herramientas que Java ofrece.