I. **INTRODUCCIÓN A JAVA**

1.1 Origen: El lenguaje de programación Java fue inventado por Sun Microsystem (empresa líder en servidores para Internet) en 1991, se le llamó originalmente “OAK” (significa roble), después fue renombrado como Java en 1995 y logrando así la madurez del prototipo original.
El impulso inicial para Java fue la necesidad de un lenguaje de programación que fuera *independiente de plataforma*, y que pudiera ser utilizado para crear software para diversos dispositivos electrónicos con mandos a distancia (hornos microondas, televisores interactivos, etc.)
Mientras se trabajaban distintos aspectos de Java, surgió la internet que fue un importante factor en el mundo. Si el internet no se hubiese desarrollado al mismo tiempo que Java estaba siendo implementado, Java podría haber sido simplemente un lenguaje de utilidad para la programación de dispositivos electrónicos de consumo. Java es para la programación en Internet lo que fue C para la programación de sistemas.
La compañía *tecnológica estadounidense* Oracle *reclamó el lenguaje* Java*, y fue adquirida en enero del* **año** *2010.*

1.2 Características de Java

Las características del lenguaje Java son:

Características	Descripción
Orientado a objetos	Permite las características de la POO: abstracción, encapsulamiento y ocultamiento, polimorfismo y herencia.
Distribuido.	Permite la construcción de aplicaciones distribuidas (en capas). El diseño de aplicaciones modernas involucra la división de una aplicación en múltiples capas; la interface de usuario, la capa negocios, y la capa de acceso a datos. Logrando que varias computadoras trabajen juntas en la red o accediendo un servidor o clientes remotos, *para enviar* o *recibir datos.* Por ejemplo, correo, páginas web, telefonía desarrollado a través de Aplets, Servlet, Jsp, etc.
Robusto, seguro y alto performance	Diseñado para la creación de software altamente confiable. Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. Provee un modelo de manejo de memoria extremadamente simple (no hay punteros definidos explícitamente por el programador) y una recolección de basura automática (Garbage Collection automático que corre como un thread de baja prioridad, aprovechando los tiempos muertos del usuario).
Arquitectura neutral, interpretado y portable	Las aplicaciones escritas en Java se ejecutan en una amplia variedad de arquitecturas de hardware y sobre múltiples sistemas operativos, mediante el **byte-code (**es un formato intermedio de arquitectura neutral que permite la portabilidad de los programas y que es interpretado por la JVM (máquina virtual de Java).
Multithreaded (multihilos)	Es la capacidad de un programa de ejecutar varias tareas simultáneamente a través de la sincronización de hilos.

1.3 Ambiente de desarrollo de java

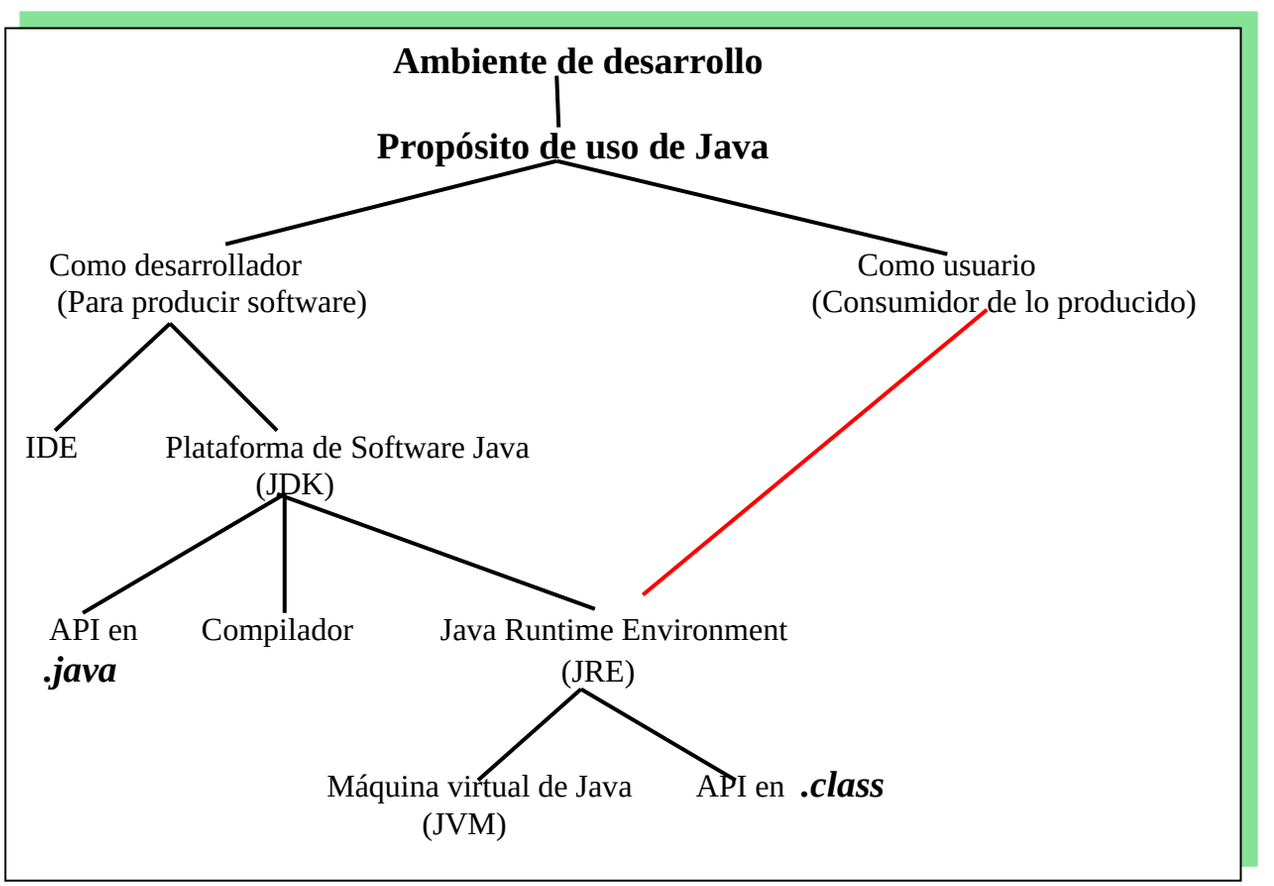
Java es una tecnología para programar y es libre de plataforma, basada en el poder de las redes y la idea de que el mismo software debe correr en muchas clases diferentes de computadoras, teléfonos, televisores, etc y diferentes sistemas operativos.

Existen dos propósitos del uso de Java: Como desarrollador y como usuario.

El desarrollador es el que crea aplicaciones y el usuario el que consume lo producido, por lo que solamente requiere ejecutar los programas. Es por esto por lo que el usuario solamente requiere del **JRE** (Java Runtime), que es el que contiene la **JVM** (*máquina virtual de Java*) + Api (librerías de .class).

Mientras que el desarrollador para generar aplicaciones en dicho lenguaje es necesario un entorno de desarrollo que tenga un **IDE y el JDK.** El JDK contiene la **Api (librerías en .java**, compilador y el **JRE, etc**).

Diagrama que resume la explicación sobre el ambiente de desarrollo y propósito de uso de Java:



1.3.1 Entornos para crear aplicaciones en el lenguaje Java.

El desarrollador requiere de un entorno para crear aplicaciones, en donde requiere de una plataforma. Una plataforma es un ambiente de software y hardware sobre el que se ejecuta un programa. La plataforma Java es sólo una plataforma de Software, que se ejecuta encima de otras plataformas de Sistemas Operativos y Hardware. Es por esto por lo que el desarrollador requiere de un: IDE y el JDK (que incluye el compilador y el JRE).

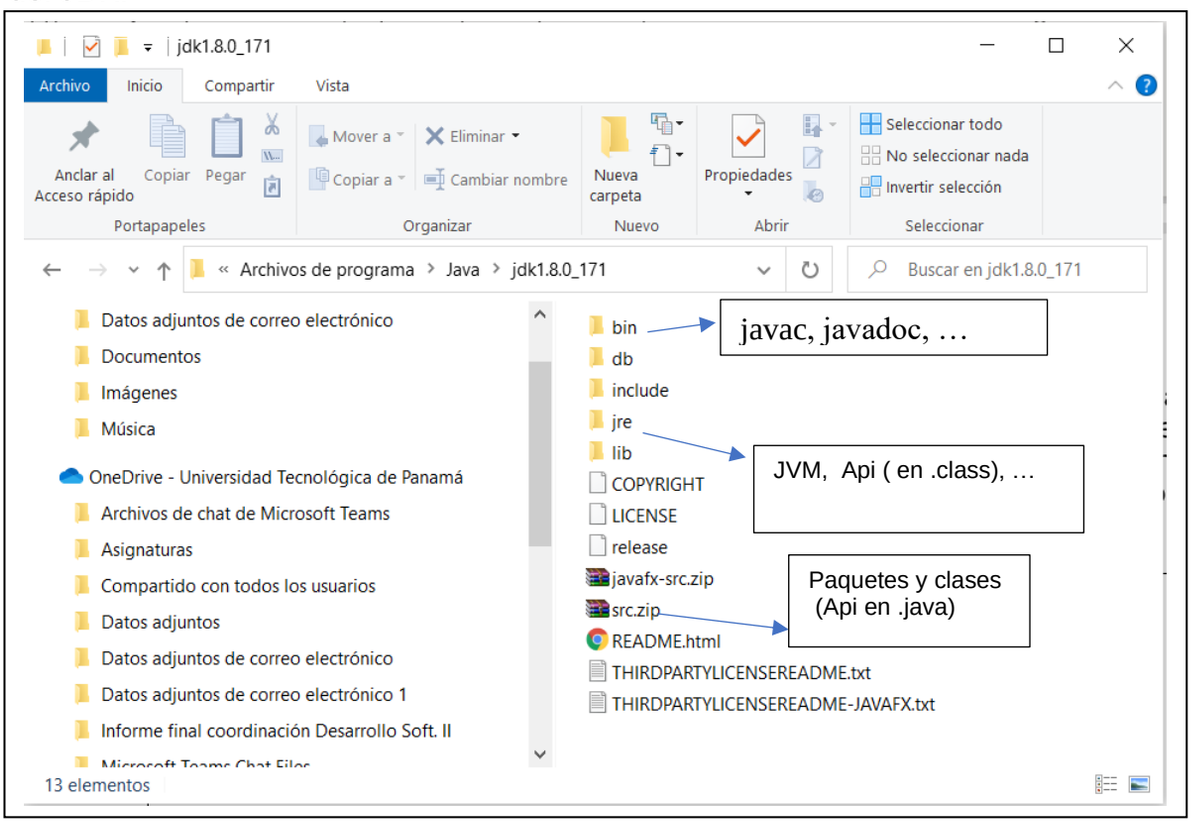
1.3.1.1 **IDE:** (*Integrated Development Environment*)

Entorno de desarrollo integrado, que proporciona un editor con un menú de opciones para escribir programas, compilar, ejecutar, depurar programas, etc.
Eclipse, Neatbeans y JCreator son ejemplos de IDE que permiten trabajar en Java.

1.3.1.2 JDK (*Java Development Kit*)

Es una plataforma de software que contiene un conjunto de librerías y archivos con programas que permiten desarrollar programas en lenguaje Java. El JDK deberá instalarse en una computadora para poder escribir programas, compilar, ejecutar, etc. Tenga presente que al instalarlo tendrá que seleccionar el JDK correspondiente al *sistema operativo de su computadora.*

A continuación, muestro el conjunto de librerías y archivos de programas más importantes del del JDK:



El javac es el ejecutable para compilar un programa escrito en Java.
El java es el ejecutable para ejecutar un programa ya compilado.

1.3.1.2.1 API (*Application Programming Interface*)

Java es un lenguaje que adoptó una sintaxis similar a la de C/ C++ tomando en cuenta sus ventajas y eliminó aquellas características que son fuentes de confusión.
Las características de C/ C++ eliminadas en Java:

- No más preprocesador (macros)
- No más estructuras y uniones
- No más tipos enumerativos
- No más funciones simples
- No hay herencia múltiple de clases
- No más go to
- No más operadores sobrecargados
- No más punteros

La API es una colección de paquetes (Librerías o bibliotecas) provista por los creadores o fabricantes del lenguaje de programación Java, para que los programadores desarrollen aplicaciones o programas Java.
Un paquete contiene todos los archivos con extensión **.java** correspondiente a las clases escritas en lenguaje Java para efectuar todo tipo de tareas necesarias dentro de un programa.

La API contiene los siguientes paquetes:

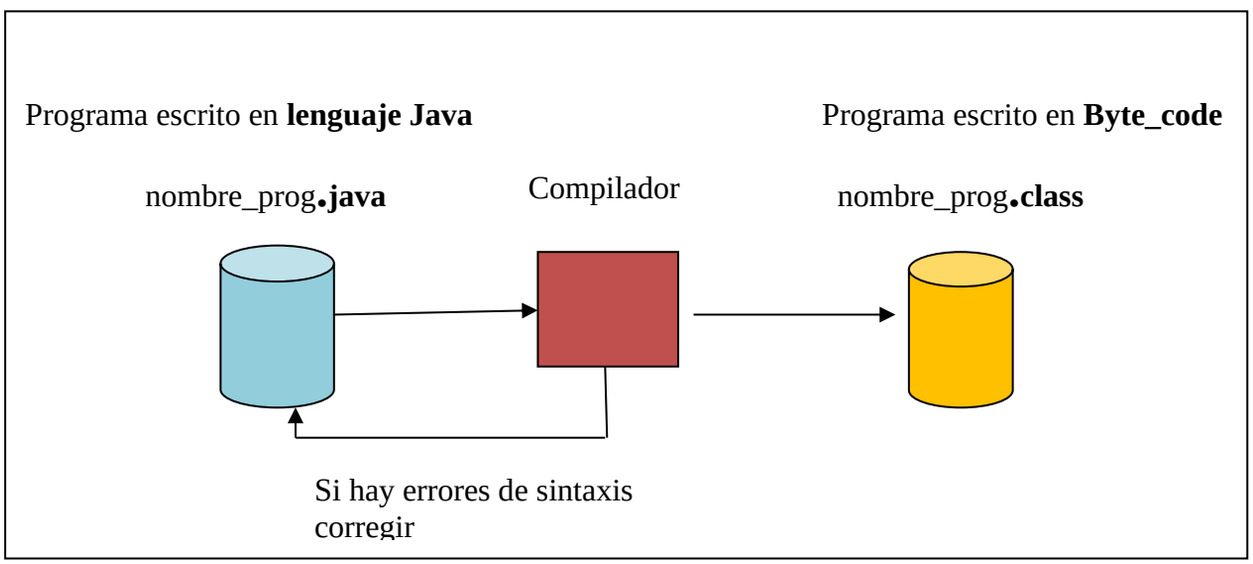
Paquetes de la API básica	Descripción
java.lang	Contiene las clases que son el corazón del lenguaje Java. Es el único paquete que se incluye automáticamente en todos los programas Java.
java.util	Contiene clases que permiten el acceso a recursos del sistema, como Date (fecha), Time (hora), etc.
java.io	Contiene las **clases** de entrada/salida de datos.
java.awt	Clases para implementar una interfaz gráfica del usuario (GUI); incluyendo clases para window, Menu, Button, Font, CheckBox, pintar gráficos e imágenes.
java.applet	Fue la primera tecnología Java para desarrollar aplicaciones web.
java.net	Contiene clases para soportar aplicaciones que acceden a redes TCP/IP (protocolos de comunicaciones de internet) que permiten implementar aplicaciones distribuidas.
java.sql	Contiene clases (JDBC) para el manejo de base de datos relacionales.
javax.swing	Contiene clases para crear interfaces de usuario mejorando el paquete awt.

1.3.1.2.2 Compilador Java:

El compilador traduce el programa fuente (con extensión .**java**), que es el código escrito en lenguaje Java, a Bytecode. El compilador lo almacena con extensión .**class** .

Byte_code (.class): Es código de máquina de bajo nivel. Es un código intermedio entre el lenguaje de máquina, procesador y Java. Evidentemente este código no es ejecutable por sí mismo en ninguna plataforma, pues no corresponde con el lenguaje de ninguno de los procesadores que actualmente se conocen.

Diagrama del compilador Java



1.3.1.2.3 JRE (*Java Runtime Environment*)

JRE es el software necesario para interpretar y ejecutar cualquier aplicación desarrollada en Java. Los archivos interpretados y ejecutados son los generados del proceso de compilación con extensión ***.class*.**

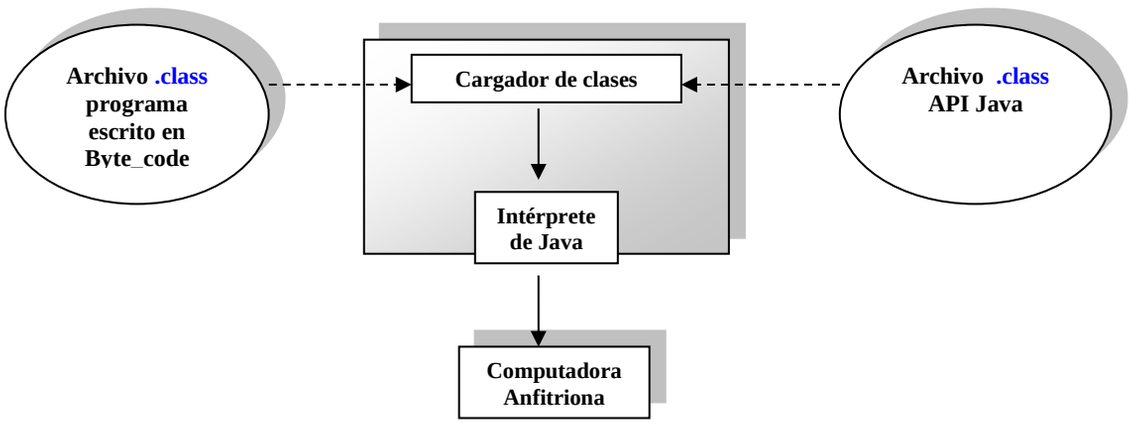
El JRE está conformado por una Máquina Virtual de Java o JVM , la API(en Byte_code) y otros.

1.3.1.2.3.1 JVM (Java Virtual Machine)

Es la base de la plataforma Java y puede ser incorporada en la mayoría de las plataformas hardware y sistemas operativos. Su misión principal es la de garantizar la portabilidad de las aplicaciones Java.

La máquina virtual Java es el entorno en el que se interpreta el Bytecode y ejecuta cada una de estas instrucciones del archivo con extensión .class, el cual es generado por el compilador del lenguaje Java.

La máquina virtual Java



La JVM consta de un cargador de clases y un intérprete de Java que ejecuta los códigos de bytes independientes de la arquitectura. El cargador de clases carga archivos .**class tanto** del programa Java que se está ejecutando como los de la ***API***, para su interpretación y ejecución.

Principales tareas de JVM:

- Verifica el código ByteCode.
- Reservar espacio en memoria para variables y objetos creados sin dejar huecos.
- Liberar memoria no usada (garbage collection).

1.3.1.2.3.2 API de Java con extensión .class(Byte_code)

Es realmente el código generado por el compilador al tratar la API fuente de Java o con extensión .java.

1.3.2 Tipos de plataformas de desarrollo en Java

Existen diferentes plataformas Java tales como: Java SE, Java EE y Java ME. Para entender las diferencias entre las plataformas va a depender del tipo de proyecto que se desea desarrollar.

Cada plataforma tiene su JDK en versiones diferentes.

A continuación, las plataformas de Java.

1.3.2.1 Java SE (Java Platform Standard Edition)

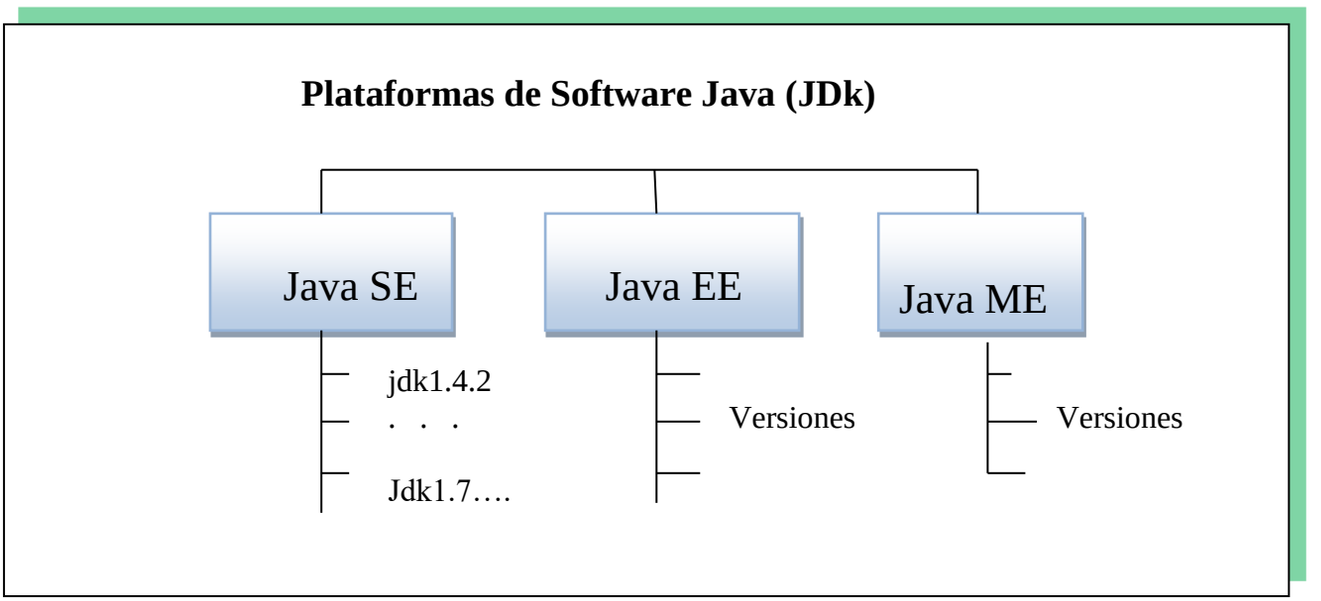
Esta es la versión estándar de la plataforma, es decir, la que la mayoría usa para desarrollar sus aplicaciones de escritorio o de la web, fue la que originalmente fue desarrollada por Sun.

1.3.2.2 **Java EE (**Java Platform Enterprise Edition)

Es la versión más grande de JAVA y se utiliza por lo general para crear aplicaciones grandes orientadas a empresas.

1.3.2.3 **Java ME** (Java Platform Micro Edition)

Esta es una plataforma de desarrollo de aplicaciones para dispositivos, como por ejemplo: teléfonos, tabletas, etc.



1.3.3 Tipos de aplicaciones que permite desarrollar el lenguaje Java

El entorno de desarrollo Java permite desarrollar aplicaciones.
Las aplicaciones no son más que programas autónomos (independientes).
Tipos de aplicaciones: Programas modo consola, programas modo gráfico, Aplets, Servlets y JSP.

Programa modo consola: Deben contener el método"main()". La interacción mediante teclado e interfaz basada en texto. Se ejecutan en modo consola.
Ejemplo:

```
public class Hola
{       public static void main (String [ ] args)
        {   System.out.println("Hola, mundo Java desde una aplicación en modo
            consola!");
        }
}
```

Programas modo gráfico (GUI): Se desarrollan ventanas graficas para la entrada y salida de datos interactivamente como interfaz para el usuario, con botones, menús, casillas de selección , etc.

Ejemplo:

```
import javax.swing.*;
import java.awt.*;
public class MiFrame extends JFrame {
public MiFrame() {
    super("¡Hola mundo con ventanas!");
    setBackground(Color.orange);
    setSize(300,300);
    setVisible(true);
}
public void paint(Graphics g) {
   g.drawString("Hola , Mundo Java desde una aplicación en modo gráfico!",40,160);}
public static void main(String args[]) {
    MiFrame mf = new MiFrame();
   mf.addWindowListener( new WindowAdapter() {
   public void windowClosing( WindowEvent evt ){
         System.exit( 0 );}
   });
}}
```

Applets: Son programas empotrados en HTML. Son programas que pueden reaccionar dinámicamente a entradas y cambios de usuario. A menudo los applets se descargan junto con una página HTML desde un Servidor Web y se ejecutan en la máquina cliente.
Es la tecnología Java más antigua para desarrollar páginas web.
Ejemplo:

```
import java.awt.Graphics;
import java.applet.Applet;

public class HolaMundo extends Applet {
public void paint( Graphics g ) {
      g.drawString( "Hola,  Mundo Java desde un  Applets !",25,25 ) ;
}
}
```

Servlets:
Los servlets al contrario de los applets son programas que están pensados para trabajar en el lado del servidor y desarrollar aplicaciones Web que interactúen con los clientes.

Ejemplo:

```
package test;
import java.io.*;
import javax.servlet.http.*;
import javax.servlet.*;
public class HelloServlet extends HttpServlet {
  public void doGet (HttpServletRequest req,
                     HttpServletResponse res)
    throws ServletException, IOException
```

```
 {
   PrintWriter out = res.getWriter();

   out.println("Hello, Mundo Java desde un Servlet!");
   out.close();
 }
}
```

JSP: (JavaServer Pages), es una tecnología Java que permite a los desarrolladores de software generar dinámicamente HTML, XML u otros tipos de documentos, en respuesta al requerimiento de un cliente web. La principal ventaja de JSP es que permite integrarse con clases Java **(.class**) lo que permite separar en niveles las aplicaciones (diseño, lógica y data) , proporcionando mayor seguridad y reutilización de código.

Ejemplo:

```
<HTML>
<HEAD>
<TITLE>Hola Mundo!</TITLE>
</HEAD>
<BODY>
<% // Hola mundo se imprime 10 veces
for ( int i = 0 ; i < 10 ; i ++ )
{
%>
Hola,  Mundo Java desde un JSP!<br>
<%
}
%>
</BODY>

</HTML>
```

JavaBeans: Son componentes software Java, que se puedan incorporar a otras aplicaciones y tienen la particularidad de ser reutilizable.

Ejemplo:

```
public class PersonaBean   implements java.io.Serializable {
    private String nombre;
    private int edad;

   public void setNombre(String n)
  {     this.nombre = n;    }

   public void setEdad(int e)
  {     this.edad = e;   }

   public String getNombre()  { return (this.nombre); }

   public int getEdad()  { return (this.edad); }
```

}

Aplicaciones móviles: Son aplicaciones que corren en dispositivos móviles (celulares, etc).

Cabe destacar que al desarrollar aplicaciones Java se pueden desarrollar por componentes o módulos y estos pueden ser implementados a través de clases, Applets , Servlets, JSP y JavaBeans, etc.