CFGS Desarrollo De Aplicaciones Multiplataforma

1º DAM

UT 1. Identificación de los elementos de un programa Informático. El lenguaje Java.

Módulo: Programación

Centro de Enseñanza
Gregorio Fernández

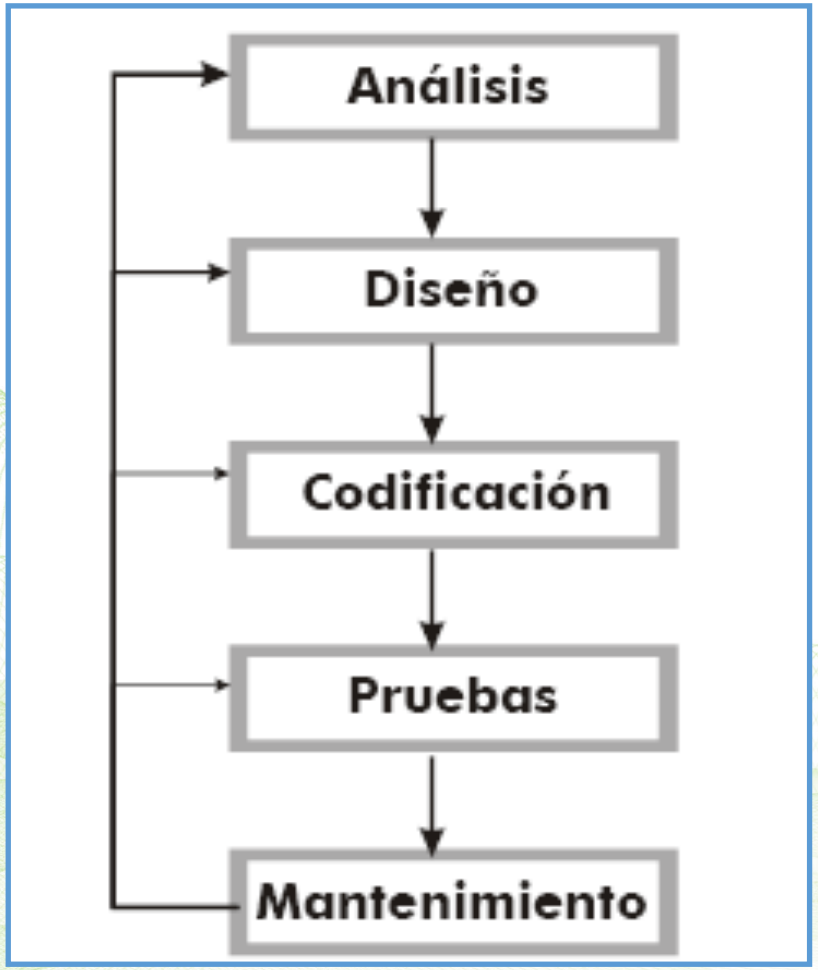
Macarena Cuenca Carbajo

Programas y aplicaciones

- Programa: **conjunto de instrucciones** ejecutables por un ordenador.
- Aplicación: software formado por **uno o más programas**, la **documentación** de los mismos y los **archivos** necesarios para su funcionamiento, de modo que el conjunto completo forman una herramienta de trabajo en el ordenador.

Macarena Cuenca Carbajo
Centro de Enseñanza
Gregorio Fernández

Ciclo de vida de una aplicación



Errores

- Salida inesperada de un programa
- Tipos:
  - Errores de usuario.
  - Errores de programador.
  - Errores de documentación.
  - Errores de interfaz.
  - Errores de E/S o de comunicaciones.
  - Errores fatales.
  - Errores de ejecución.

Lenguajes de programación
I-Historia

1ª generación: Lenguaje máquina.

- **Código máquina**: unos y ceros.
- Durante mucho tiempo esa fue la forma de programar.
- Los ordenadores es el único código que entienden, con lo cual, cualquier forma de programar debe de ser convertida a código máquina.
- Su incomodidad de trabajo hace que sea impensable para ser utilizado hoy en día.
- El código máquina es distinto para cada tipo de procesador.
  Lo que hace que los programas en código máquina **no sean portables** entre distintas máquinas.

Macarena Cuenca Carbajo
Centro de Enseñanza
Gregorio Fernández

Lenguajes de programación
I-Historia

2ª generación: Lenguaje ensamblador.

- **Lenguaje ensamblador:** traducción del código máquina a una forma más textual.
- Cada tipo de instrucción se asocia a una palabra mnemotécnica (como SUM para sumar), de forma que cada palabra tiene traducción directa en el código máquina.
- El programa en código ensamblador, es traducido a código máquina.
- Esta traducción es rápida puesto que cada línea en ensamblador tiene equivalencia directa en código máquina (en los lenguajes modernos no ocurre esto).

Lenguajes de programación
I-Historia

2ª generación: Lenguaje ensamblador.

- **Programas no portables**: cada programa sólo funcionará para la máquina en la que fue concebido el programa.
- Las líneas requeridas para realizar una tarea se disparan ya que las instrucciones de la máquina son excesivamente simples.
- La ventaja de este lenguaje es que se puede controlar absolutamente el funcionamiento de la máquina, lo que permite crear **programas muy eficientes**.

Macarena Cuenca Carbajo
Centro de Enseñanza
Gregorio Fernández

Lenguajes de programación
I-Historia

3ª generación: Lenguajes de alto nivel.

- Solucionan los problemas del lenguaje ensamblador.
- El código vale para cualquier máquina, pero deberá ser traducido a código máquina mediante software especial.
- En 1957 aparece el que se considera el primer lenguaje de alto nivel, el **FORTRAN** (FORmula TRANslation), lenguaje orientado a resolver fórmulas matemáticas.
- En 1958 se crea **LISP** como lenguaje para expresiones matemáticas.
- En 1960 se creó **COBOL** como lenguaje de gestión.
- En 1963 se crea **PL/I** el primer lenguaje que admitía la multitarea y la programación modular.

Macarena Cuenca Carbajo

Lenguajes de programación
I-Historia

3ª generación: Lenguajes de alto nivel.

- **BASIC** se creo en el año 1964 como lenguaje de programación sencillo de aprender, ha sido uno de los lenguajes más populares.
- En 1968 se crea **LOGO** para enseñar a programar a los niños.
- **Pascal** se creó con la misma idea académica, pero siendo ejemplo de lenguaje estructurado para programadores avanzados.
- El creador del Pascal (Niklaus Wirdth) creó **Modula** en 1977 siendo un lenguaje estructurado para la programación de sistemas (intentando sustituir al C).

Lenguajes de programación
I-Historia

4ª generación

- En los años 70 se empezó a utilizar éste término para hablar de lenguajes en los que apenas hay código y en su lugar aparecen indicaciones.
- Lenguaje **SQL** (muy utilizado en las bases de datos) y sus derivados.
- Lenguajes que se programan sin escribir casi código (**lenguajes visuales**).
- También se consideran dentro de esta generación a los **lenguajes orientados a objetos**.

Macarena Cuenca Carbajo
Centro de Enseñanza
Gregorio Fernández

Lenguajes de programación
I-Historia

Lenguajes orientados a objetos.

- Llegan en los 80.
- Destacó inmediatamente **C++**.
- A partir de entonces, numerosos lenguajes clásicos se convirtieron en lenguajes orientados a objetos: **Visual Basic**, **Delphi** (versión orientada a objetos de Pascal), **Visual C++**, ...
- En 1995 aparece **Java** como lenguaje totalmente orientado a objetos.
- En el año 2000 aparece **C#** un lenguaje que toma la forma de trabajar de C++ y del propio Java.

Lenguajes de programación
I-Historia

Lenguajes para la Web.

- La popularidad de Internet ha producido lenguajes híbridos que se mezclan con el código HTML.
- HTML no es un lenguaje en sí sino un formato de texto pensado para crear páginas web.
- Lenguajes interpretados como **JavaScript, VBScript** o **TypeScript**
- Lenguajes para uso en servidores como **ASP**, **JSP** o **PHP**.

Actividad multimedia

Visionado de vídeos:

- “Historia de los lenguajes de programación”

Lenguajes de programación
II-Intérpretes y Compiladores

- A la hora de convertir un programa en código máquina, se pueden utilizar dos tipos de software:
  - Intérpretes
  - Compiladores

Macarena Cuenca Carbajo

Lenguajes de programación
II-Intérpretes

- Se convierte cada línea a código máquina y se ejecuta antes de convertir la siguiente línea.
- La mayoría de los lenguajes para la creación de **páginas web** son interpretados.
- **JavaScript** o incluso, en parte, **Java** son lenguajes interpretados.
- Pasos para convertir un programa a código máquina mediante un intérprete:
    1. Lee la primera instrucción
    2. Comprueba si es correcta
    3. Convierte esa instrucción al código máquina equivalente
    4. Lee la siguiente instrucción
    5. Vuelve al paso 2 hasta terminar con todas las instrucciones

Macarena Cuenca Carbajo

Lenguajes de programación
II-Intérpretes

+ Adecuado para programas multiplataforma.
- Código máquina no optimizado.
- Todos los errores son en tiempo de ejecución.
- Ejecución más lenta de los programas.
- Código fuente “accesible”.

Macarena Cuenca Carbajo
Centro de Enseñanza
Gregorio Fernández

Lenguajes de programación II-Compiladores

- Traducen las instrucciones de un lenguaje de programación de alto nivel a código máquina.
- **Analizan todas las líneas** antes de empezar la traducción.
- Durante muchos años, los lenguajes potentes han sido compilados.
- El uso masivo de Internet ha propiciado que esta técnica a veces no sea adecuada y haya lenguajes modernos **interpretados** o **semi-interpretados**.
- En ellos, primero se compila hacia un código intermedio y luego se interpreta línea a línea.
- Esta técnica la siguen **Java** y los lenguajes de la plataforma **.NET** de Microsoft.

Lenguajes de programación
II-Compiladores

+ Se detectan errores antes de ejecutar el programa: errores de compilación.
+ Código máquina generado optimizado.
+ Es más fácil el proceso de depuración.
+ Programas más rápidos.
+ Código fuente protegido.
- Proceso de compilación lento.
- No es adecuado para programas Web.

Macarena Cuenca Carbajo

Programación

Codificar algoritmos en algún lenguaje de programación para que puedan ser entendidos por el ordenador.

Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

Programación
I-Técnicas

Programación convencional.

- Programación desordenada.
- Predomina el instinto del programador por encima del uso de cualquier método.

- La corrección y entendimiento de este tipo de programas sea casi ininteligible.

Macarena Cuenca Carbajo

Programación
I-Técnicas

Programación estructurada.

Técnica que utiliza únicamente tres estructuras de control:

- **Secuencias** (instrucciones que se generan secuencialmente)
- **Estructuras alternativas** (sentencias if)
- **Estructuras Iterativas** (bucles)

Macarena Cuenca Carbajo
Centro de Enseñanza
Gregorio Fernández

Programación
I-Técnicas

Programación estructurada.

+ Los programas son más fáciles de entender.
+ Reducción del esfuerzo en las pruebas.
+ Reducción de los costos de mantenimiento.
+ Programas más sencillos y más rápidos.
+ Aumento en la productividad del programador.

Macarena Cuenca Carbajo

Programación I-Técnicas

Programación modular.

- Descomposición del problema (programa) en partes denominadas **módulos**, cada uno de los cuales realiza una tarea concreta del programa.
- Un módulo principal (**programa principal**) llama al resto de los módulos, devolviendo el control al módulo principal cuando finalizan.
- Si la tarea asignada a cada módulo es demasiado compleja éste deberá romperse en otros módulos más pequeños.
- Este proceso de división continuará hasta que cada módulo tenga solamente una tarea específica que ejecutar.
- Los módulos son independientes, con lo cual, diferentes programadores pueden trabajar simultáneamente en diferentes partes del mismo programa.
- Esto reduce el tiempo de diseño del algoritmo y codificación del programa.
- Un módulo se puede modificar sin afectar a otros módulos.

Programación
I-Técnicas

Programación orientada a objetos.

- Es la tendencia actual.
- Las aplicaciones se representan como una serie de objetos independientes que se comunican entre sí.
- Cada objeto posee datos y operaciones propias.
- Los programadores se concentran en programar independientemente cada objeto y luego generar el código que inicia la comunicación entre ellos.
- Revoluciona las técnicas de programación.
- Su éxito radica, en la facilidad que posee esta técnica para encontrar fallos, reutilizar el código y documentarlo fácilmente.

Programación
I-Técnicas

Otras:

- Programación orientada a aspectos (POA).
- Programación orientada a eventos.

Macarena Cuenca Carbajo

Actividad multimedia

Visionado de vídeos:

- “8 consejos para programar”
- “’Hola mundo”
- “Top 10 de los lenguajes de programación”

El lenguaje Java
I-Historia

- A principios de los 90 **Sun Microsystems** pensó que el futuro estaba en la electrónica de consumo.
- El primer proyecto en este campo fue el proyecto **Green**. Formado por un pequeño equipo de empleados de Sun que utilizan C++.
- Pronto se dieron cuenta que para programar los chips de los aparatos electrónicos necesitarían un lenguaje mas portable y fiable que **C++** ya que iban a requerir continuas actualizaciones.
- Deciden utilizar el lenguaje **Oak**.
- Este lenguaje fue desarrollado por **James Gosling** uno de los componentes principales del proyecto Green.
- Finalmente y tras varios prototipos fallidos no se tuvo éxito en la empresa y se cerró la iniciativa.

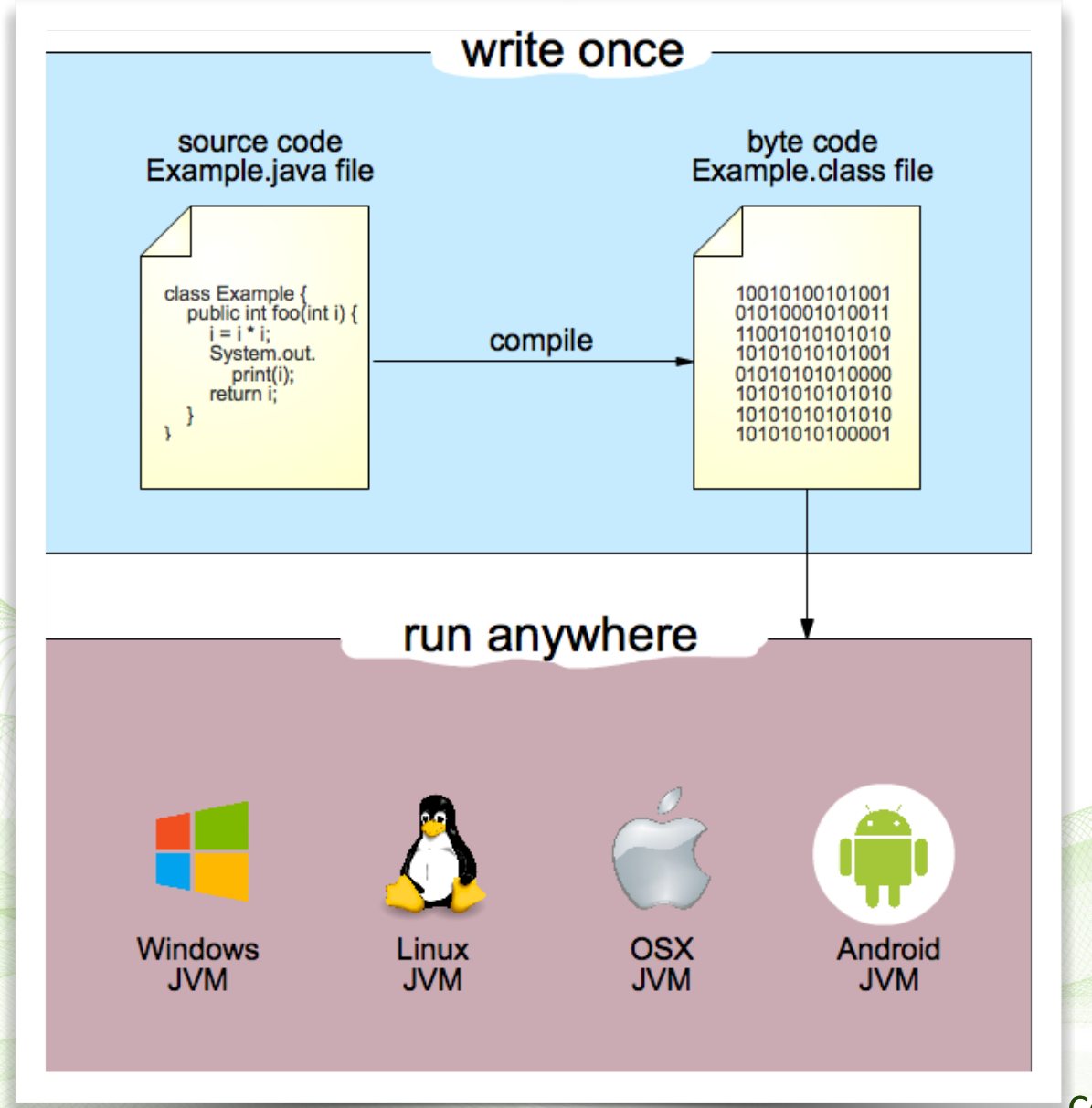
El lenguaje Java
I-Historia

- A mediados de 1993 comenzó el boom de Internet y es entonces cuando se recuperó el lenguaje **Oak**.
- Este nuevo proyecto va tomando forma y en enero de 1995 Oak se renombra a **Java**, un tipo de café que tomaban los empleados de Sun en sus reuniones.
- En agosto de 1995 el escaparate de Internet era el navegador **Netscape** y Sun firma un acuerdo con Netscape para incluir Java en todos sus navegadores.
- En ese momento nace oficialmente **Java**.
- En 1996 aparece la primera especificación formal de la plataforma conocida como **JDK 1.0**.
- A principios de 1997 aparece la primera versión comercial, la **JDK 1.1.**
- En diciembre de 1998 aparece **Java 2**.

El lenguaje Java
II-¿Qué es Java?

- **Lenguaje de programación** .
- **Entorno de ejecución:** conocido como **Java Virtual Machine**.
- El compilador de Java traduce el código fuente en instrucciones de bajo nivel, pero éstas son interpretadas por la Java Virtual Machine (JVM) en vez de por el procesador del ordenador.
- Java es por tanto un lenguaje **compilado** e **interpretado**.

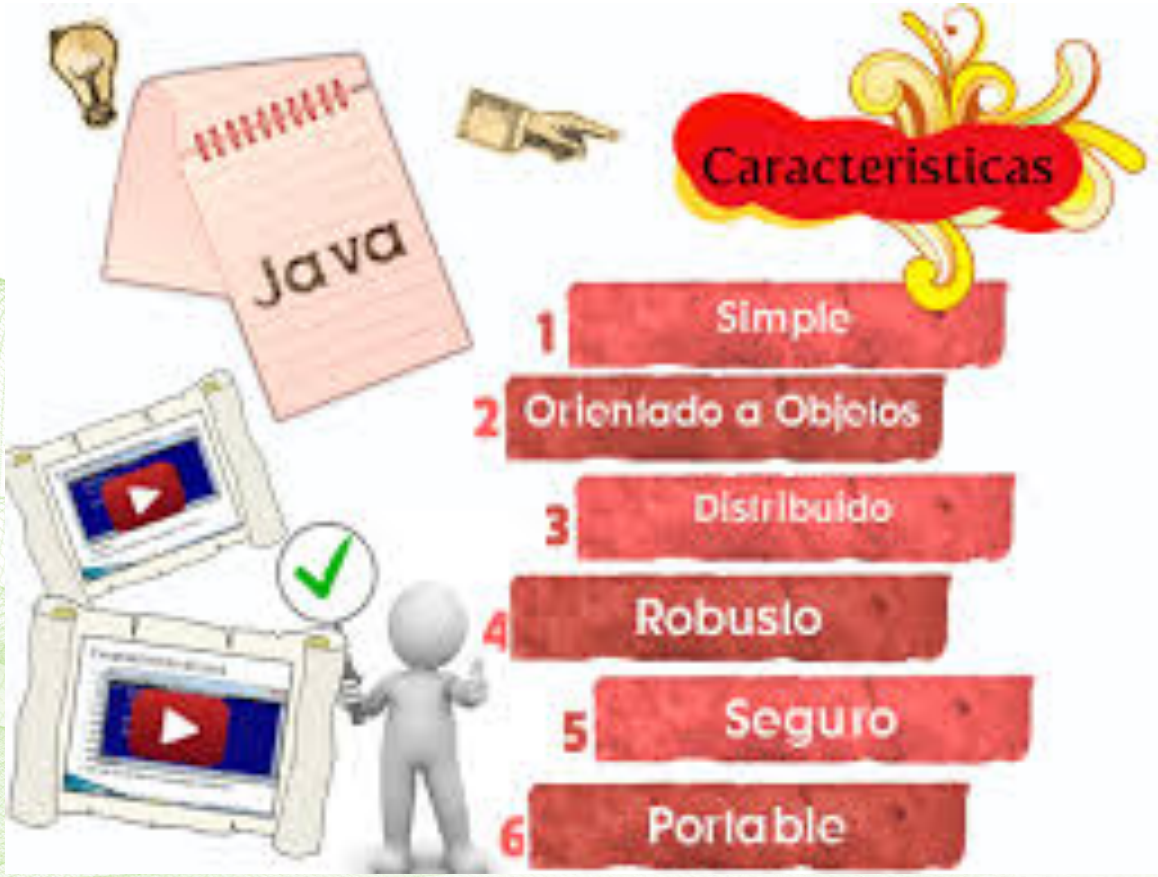
El lenguaje Java
II-¿Qué es Java?



Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

El lenguaje Java
III-Características



Macarena Cuenca Carbajo

El lenguaje Java
IV- Versiones

- **Java 1**:
  - **Java 1.0** (1996). Constaba de sólo 12 paquetes.
  - **Java 1.1** (1997). Constaba de 23 paquetes.
- **Java 2**:
  - **Java 2 1.2** (1998). Constaba de 59 paquetes. Se producen notables mejoras a todos los niveles, por eso Sun lo renombra como "Java 2" y divide la plataforma en tres grandes bloques:
    - **J2SE (Java 2 Standard Edition).** Plataforma para desarrollo de aplicaciones de escritorio o pequeños servidores.
    - **J2EE (Java 2 Enterprise Edition).** Plataforma para desarrollo de aplicaciones en servidores dentro de un entorno empresarial.
    - **J2ME (Java 2 Micro Edition).** Plataforma para desarrollo de aplicaciones en dispositivos móviles.
  - **Java 2 1.3** (2000). Consta de 76 paquetes.
  - **Java 2 1.4** (2001). Consta de 103 paquetes.
  - **Java 2 1.5** (2004). Consta de 131 paquetes. Renombrado por motivos de marketing como **Java 5.0**.

El lenguaje Java
IV- Versiones

* Versiones *LTS (Long Term Support)*: ORACLE proporciona 3 años de soporte en lugar de solo 6 meses.

Lanzamientos cada 6 meses

- **Java SE 6** (diciembre 2006). En esta versión, Sun cambió el nombre "J2SE" por Java SE y eliminó el ".0" del número de versión. Incluye mejoras muy avanzadas y específicas.
- **Java SE 7**. Su lanzamiento fue en julio de 2011.
- **Java SE 8** (marzo 2014). Entre las características que tiene, cabe destacar el soporte para nuevas técnicas de programación y funciones (como las **lambda**), así como una mayor integración con **Java FX**.
- **Java SE 9** (septiembre 2017).
- **Java SE 10** (marzo 2018).
- **Java SE 11** (septiembre 2018).
- **Java SE 12** (marzo 2019).
- ...
- **Java SE 20** (marzo 2023).
- **Java SE 21** (septiembre 2023).
- **Java SE 22** (marzo 2024).
- ...
- **Java SE 25** (septiembre 2025).

El lenguaje Java
V- Herramientas de desarrollo

- El **JDK** (Java Development Kit) es el conjunto de herramientas para desarrollar en Java.
- Puede descargarse gratuitamente en la página oficial de Oracle: http://www.oracle.com.
- La instalación por defecto se hace en un directorio bajo la raíz del sistema.
- El compilador y el intérprete (la JVM) se sitúan en la carpeta **bin**
- El compilador es el programa llamado *javac.exe* y la máquina virtual es el programa *java.exe*.
- De manera independiente al JDK se puede descargar la documentación de Java desde la página oficial. La documentación (API) es donde se describen todas las librerías Java. Esta documentación suele instalarse en un directorio **docs** bajo el directorio donde se haya instalado el JDK para poder ser consultada "offline".

El lenguaje Java
V- Herramientas de desarrollo

- Un modo de trabajo usual en Windows sería tener abiertas las siguientes ventanas:
  - Un editor de textos (por ejemplo el Bloc de notas).
  - Una ventana MS-DOS para compilar y ejecutar el programa.
  - Un navegador con el API de Java.
- La desventaja del **JDK** es que es no es un entorno de programación muy "amigable" ya que está basado en comandos (carece de interfaz gráfica).
- Por ello, existen entornos gráficos de programación, denominados **IDE** (*Integrated Development Environment*). Los IDE comerciales más utilizados son:
  - Eclipse
  - IntelliJ
  - NetBeans (Oracle)
  - Visual Studio Code *

El lenguaje Java
VI - Primer programa

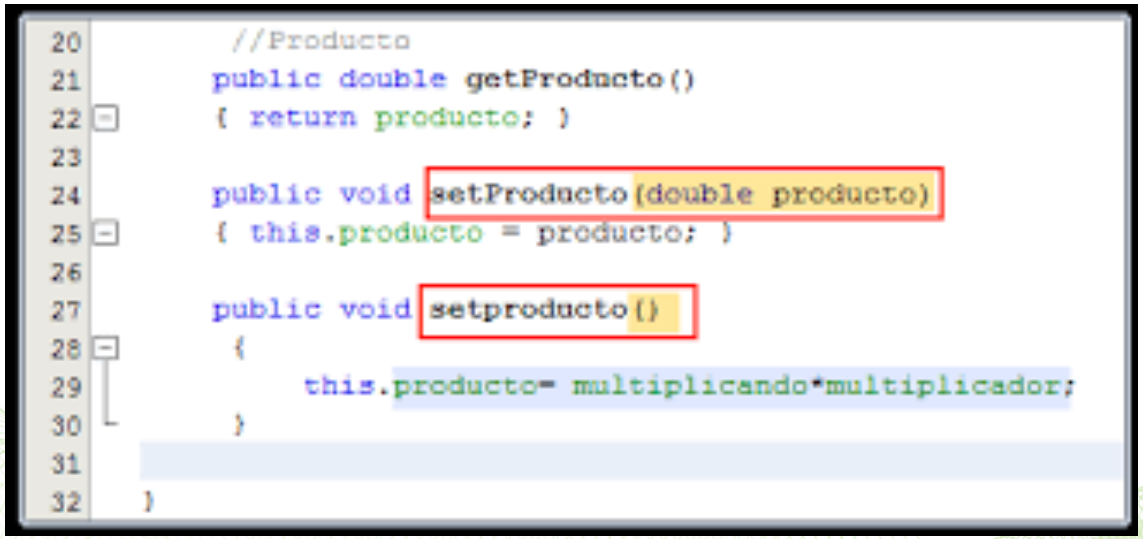

```
public class HolaMundo {
  public static void main( String[] _args ) {
    System.out.println("Hola Mundo!!!");
  }
}
```


Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

Componentes de un programa Java

- Comentarios
- Palabras reservadas
- Modificadores
- Sentencias
- Bloques
- Clases
- Métodos (“main”)



```
20          //Producto
21          public double getProducto()
22          { return producto; }
23
24          public void setProducto(double producto)
25          { this.producto = producto; }
26
27          public void setproducto()
28          {
29              this.producto= multiplicando*multiplicador;
30          }
31
32      }
```


Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

Componentes
I - Comentarios

- Documentan el programa.
- No son sentencias de programación, (no “engordan” el programa compilado).
- Dos tipos de comentarios:
  - **Comentarios de una línea** (//):

```
int i = 0; //contador de letras
//Aquí sumamos 1
i++;
```

  - **Comentarios de varias líneas: /* … */:**

```
/*
* Comentario de
* varias
* líneas
*/
```

```
/**
* Método: getImage
* @param url una dirección absoluta
* name la localización relativa
* @return la imagen especificada
*/
```


Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

Componentes
II – Palabras reservadas

- Son palabras especiales para el compilador que no pueden ser utilizadas por el programador para otros fines diferentes a aquellos para los que fueron diseñadas.
- Ejemplos de palabras reservadas son:

```
class, while, if, static, public
```


Macarena Cuenca Carbajo

Componentes
III – Modificadores

- Son un tipo de palabras reservadas que especifican de qué tipos son los datos, los métodos y las clases y por tanto de qué modo pueden ser utilizados.
- Ejemplos de modificadores son:

```
public, static, private, final
```

Componentes
IV – Sentencias

- Representan una **acción** o **secuencia de acciones**.
- Terminan en punto y coma (**;**).
- Ejemplos:

```
i++;
z = z + 10;
System.out.println("La fecha es: " + new Date());
```

V – Bloques

- Estructuras que agrupan sentencias.
- Comienzan por "{" y terminan por "}".
- Las clases, sentencias de selección y estructuras de control (bucles) encierran su código entre bloques.
- Ejemplo:

```
if (z < 10) {
        System.out.println("z es menor que 10 ");
}
```

Componentes
VI – Clases

- **Plantilla para crear objetos**.
- La definición de una clase comienza con la palabra reservada **class**.
- Las clases compiladas son grabadas en ficheros de extensión class.

VII – Métodos

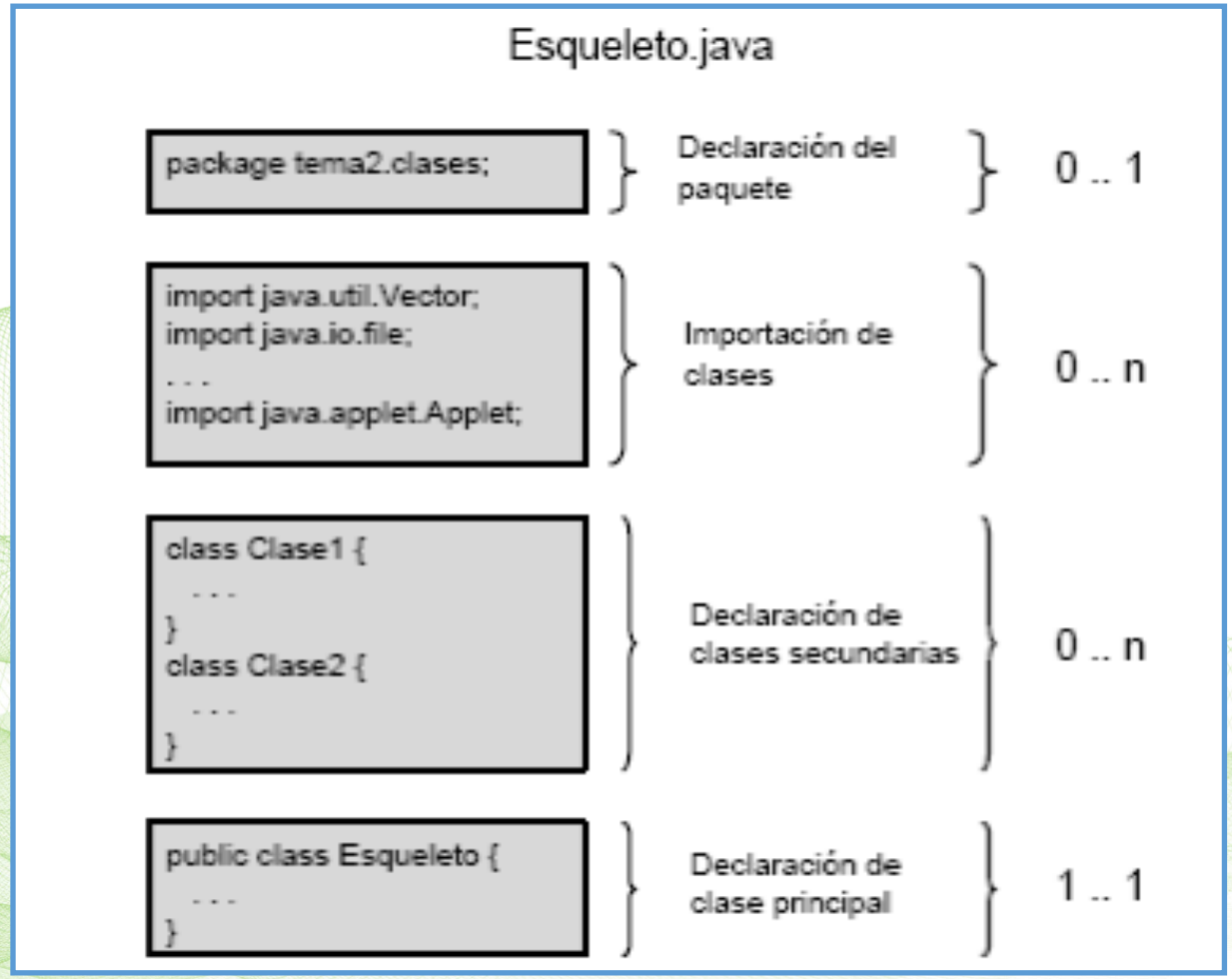
- Colección de sentencias que realizan una serie de operaciones.
- Existen métodos predeterminados de Java y métodos escritos por el programador, como podría ser por ejemplo un método para calcular la superficie de un círculo:

```
calculaSuperficie()
```

- Método *main()*:

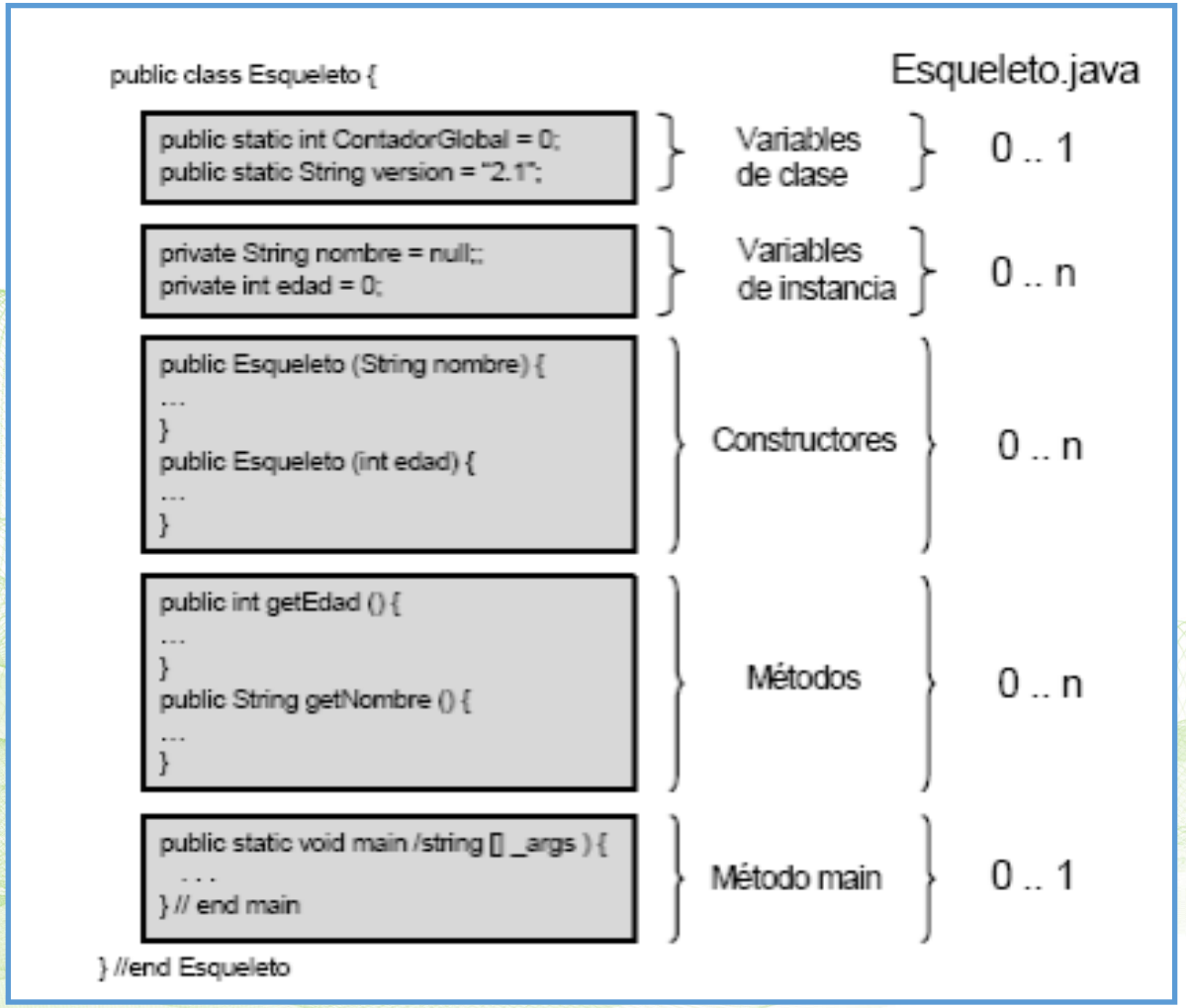
```
public static void main(String[] _args) {
    // sentencias y llamadas a métodos
}
```

Estructura de un programa Java



Macarena Cuenca Carbajo

Estructura de un programa Java
I – Clase principal



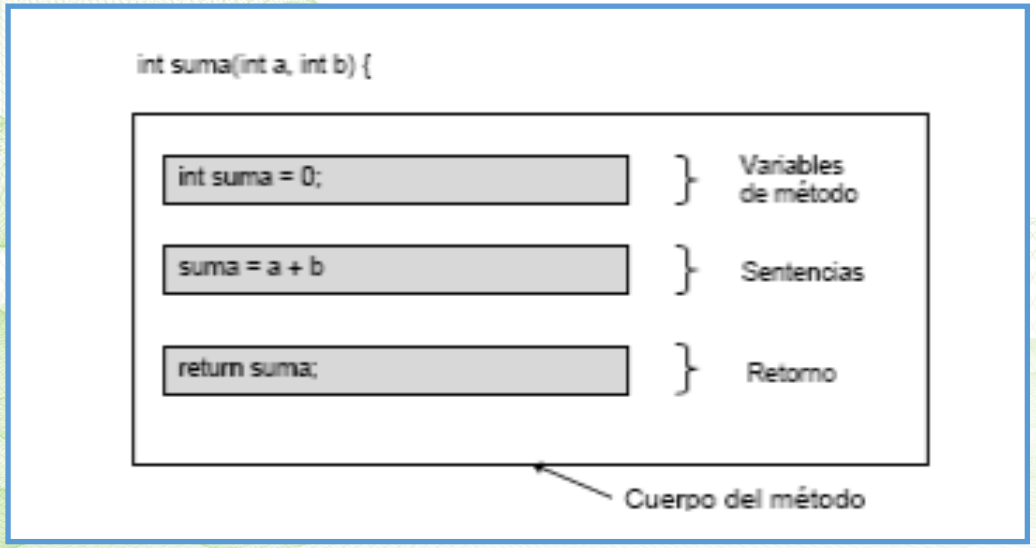
Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

Estructura de un programa Java
II – Clases secundarias

- Misma estructura que la clase principal, a excepción de que no tienen método **main().**

III – Métodos



Macarena Cuenca Carbajo
Centro de Enseñanza
Gregorio Fernández

Nomenclatura Java

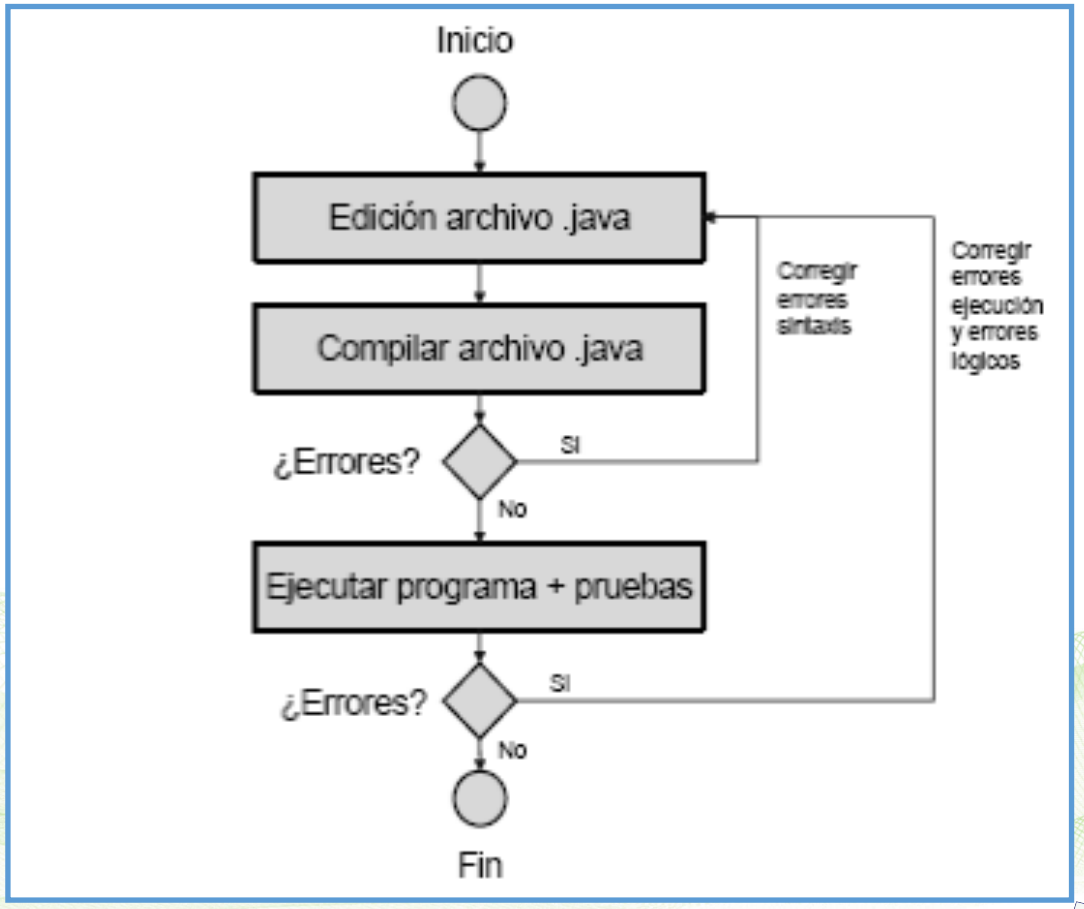
- Case-sensitive. Las variables **num**, **Num** y **NUM** son diferentes.
- Reglas respecto a los nombres que facilitan la lectura y el mantenimiento de los programas:
  - Habitual utilizar nombres con **minúsculas**, con las excepciones siguientes.
  - Cuando un nombre consta de varias palabras poner una a continuación de otra, poniendo con **mayúscula la primera letra** de la palabra que sigue: `elMayor()`, `RectanguloGrafico`
  - Los nombres de **clases** e **interfaces** comienzan siempre por **mayúscula**: `Rectangulo`, `Dibujable`, `Graphics`
  - Los nombres de **objetos**, los nombres de **métodos** y **variables miembro**, y los nombres de las **variables locales**, comienzan siempre por **minúscula**: `main()`, `dibujar()`, `numRectangulos`, `x`, `y`, `r`
  - Los nombres de las **constantes** en mayúsculas: `PI`

Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

Errores de programación

- **Bugs** (bichos/insectos).
- El proceso para corregir un error se denomina **debugging**.
- El ciclo de construcción de un programa es un ciclo iterativo:



Macarena Cuenca Carbajo

Tipos de errores

I – Errores de compilación

- **Errores de sintaxis**: violan la gramática del lenguaje Java.
- Son detectados por el compilador.
- Ejemplos típicos:
  - Llaves no cerradas
  - Variables no declaradas
  - Palabras reservadas mal escritas
  - Signos de puntuación olvidados (sentencias no finalizadas con “;”)
- Fáciles de detectar y de corregir.
- Aconsejable compilar bastante a menudo (cada 10 sentencias nuevas por ejemplo).

Macarena Cuenca Carbajo

Tipos de errores

I – Errores de compilación

```
1. public class Prueba {
2.    public static void main(String [] args) {
3.       z = 50;
4.       System.out.println(z+10);
5.    }
6. }
```

- La salida del compilador es bastante autoexplicativa:

```
Prueba.java:3: cannot resolve symbol
symbol : variable z
location: class Prueba
z = 50;
^
Prueba.java:4: cannot resolve symbol
symbol : variable z
location: class Prueba
System.out.println(z+10);
^
2 errors
```

Tipos de errores

II – Errores de ejecución

- Se producen en **tiempo de ejecución** y pueden dar lugar a graves consecuencias.
- Las instrucciones que generan este tipo de errores son sintácticamente correctas por lo que son traducidas a código máquina, pero cuando el ordenador intenta ejecutarlas, no puede, y se produce la excepción.
- Tipos:
  - **Errores de operación**: aparecen cuando se hacen operaciones no permitidas.
  - **Errores de datos**: debido a que se han recibido datos incorrectos, como por ejemplo de un tipo diferente al esperado.
- Solución: **excepciones**

Tipos de errores

II – Errores de ejecución

```
1. public class Prueba {
2.   private static int z;
3.
4.   private static void divide() {
5.     z = 10/z;
6.   }
7.
8.   public static void main(String[] _arg) {
9.      divide();
10.  }
11.
12. }
```

- El error devuelto en tiempo de ejecución es el siguiente::

```
Exception in thread "main" java.lang.ArithmeticException:
/by zero
at Prueba.divide(Prueba.java:6)
at Prueba.main(Prueba.java:10)
```

Tipos de errores

III – Errores lógicos

- Son los errores más difíciles de detectar ya que el programa finaliza su ejecución y puede parecer que todo ha funcionado correctamente ya que no se muestran mensajes de error.
- Se producen cuando los resultados obtenidos no son los esperados.
- Debidos a un **error en el diseño del algoritmo**.
- Solución: **baterías de pruebas** muy detalladas que cubran el mayor número posible de casos de prueba.

Actividades

Actividades Tema1

Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández