La Sénia
Institut d'Educació Secundària
Paiporta

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Introducción al Java

Curso 2023 - 2024

1. Java Development Kit

El **Java Development Kit** (JDK) es un entorno de desarrollo de software utilizado para desarrollar aplicaciones y applets de Java.
Incluye:

- Incluye Java Runtime Environment (JRE).
- Un intérprete/cargador (Java).
- Un compilador (javac).
- Un archivador (jar).
- Un generador de documentación (Javadoc).
- Otras herramientas necesarias para el desarrollo de Java.

1. Java Development Kit

JRE significa “**Java Runtime Environment**”.

Java Runtime Environment proporciona los requisitos mínimos para ejecutar una aplicación Java.

Los elementos que incluye son:

- Java Virtual Machine (JVM)
- clases principales
- archivos auxiliares.

1. Java Development Kit

JVM significa “**Java Virtual Machine**”.

Su implementación ha sido proporcionada por Sun y otras compañías.

- Una implementación es un programa de computadora que cumple con los requisitos de la especificación JVM.
- Es una instancia en tiempo de ejecución. Siempre que escriba el comando **`java`** en el símbolo del sistema para

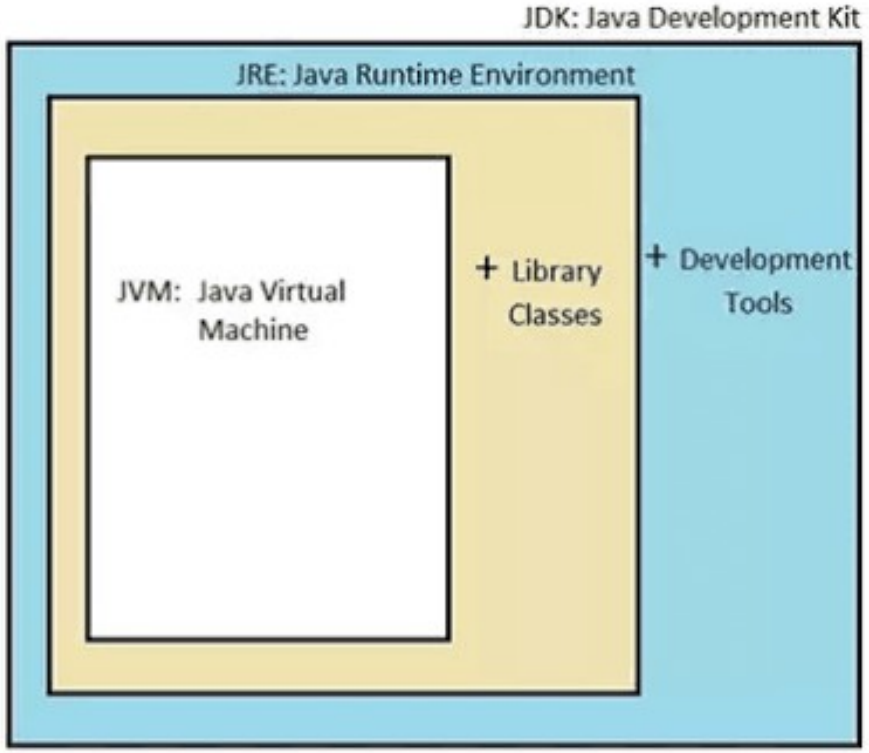
1. Java Development Kit

JDK – Incluye dos cosas:
1) JRE
2)Herramientas de desarrollo

JDK sólo lo necesitan los desarrolladores del Java.

JRE - Proporciona un entorno para ejecutar (no desarrollar) el programa Java. Para usuarios finales.

JVM - responsable de ejecutar el programa Java línea por línea.



JDK = JRE + Development Tools
JRE = JVM + Library Classes

2. Proyectos en Java

Un **proyecto Java** podemos considerarlo como una serie de carpetas ordenadas y organizadas de acuerdo con una lógica para mantener organizado el código.
Un proyecto suele constar de archivos .java, archivos .class entre otros.

- Los archivos **.java** contienen el código fuente y suelen encontrarse en carpetas de nombre **src** (source).
- Los archivos **.class** contienen el bytecode (código binario ejecutable por la máquina virtual Java) y suelen encontrarse en carpetas de nombre **bin** (binary).

2. Proyectos en Java

- Los **paquetes** (**`package`**) en Java es un concepto similar al de carpeta de usuario: un contenedor donde mantenemos cosas relacionadas entre sí.
- La organización del proyecto será similar a la organización de archivos: en un paquete podemos tener clases de tipo A, en otro clases de tipo B, etc. A su vez, un paquete puede contener subpaquetes: el paquete A puede contener a los subpaquetes A.1, A.2 y A.3, etc.
- Así pues, un package es una agrupación de clases afines, similar al concepto de **librería** en otros lenguajes.

Nota: para usar archivos de otro package se usa la palabra reservada **`import`**

2. Proyectos en Java

```
//package mipaquete;

public class Hola
{
    public static void main(String[] args)
    {
        System.out.println("Hola mundo");
    }
}
```

3. Variables

Una variable es un espacio de la memoria del ordenador a la que asignamos un valor.

Para declararlas, debemos indicar:

- **`tipo_dato`** de la variable: valor numérico, carácter o cadena de caracteres, etc.
- **`nombre`**: representa a la variable y permite acceder a ella.
- **`valor_inicial`** (opcional): es posible asignarle un valor inicial cuando se crea la variable, lo que se conoce como ***inicializar*** la variable

```
tipo_dato nombre = valor_inicial
```

4. Tipos de datos

Enteros (**`int`**): Son los números enteros. Como horas exactas o unidades de producto
Reales (**`double`**): Son los números con decimales. Como precios de Productos.
Lógicos (**`boolean`**): Tienen dos valores posibles que son Verdadero o Falso.
Carácter (**`char`**): Son las letras del alfabeto.
Cadena de caracteres (**`String`**): Son un conjunto de caracteres como el nombre y apellidos de una persona
Nota: String es el único tipo con mayúscula ya que en realidad se trata de una clase con métodos (wrapped) y no de un tipo simple

4. Tipos de datos

```
public class Ejemplo02 {
   private double precio;
   private String profesor;
   private String aula;
   private int unidades;
   private boolean funciona;
   private boolean esVisible;
   private float diametro;
   private short edad;
   private long masa;
   private char letra1;
}
```

```
precio = 42; // Tipo double
profesor = “Angel Berlanas”; //String
aula = “INFO4”; //String
unidades = 1500; // Entero tipo int
funciona = true; // Tipo boolean
esVisible = false; // Tipo boolean
diametro = 34.25f; // Tipo float. Una f o F
final indica que es float.
edad = 19; // Entero tipo short
masa = 178823411L; // Entero tipo long. Una
l o L final indica que es long.
letra1 = ‘s’; // Tipo char (carácter)
```

Nota: el tipo char se escribe con comillas simples y el String con comillas dobles

5. Literales y Constantes

- Los valores **literales** son aquellos que podemos asignar a las variables. Dependiendo del tipo de variable podremos asignar unos valores u otros, como hemos visto en el ejemplo anterior
- Una **constante** es una variable del sistema que mantiene un valor inmutable en todo el programa. Las constantes en Java se definen mediante el modificador ***final*** y se utilizan con su nombre como una variable más:

```
static final int DIAS_SEMANA = 7;
System.out.println("El número de días de la semana son " + DIAS_SEMANA);
```

6. Operadores y expresiones

En Java podemos usar los siguientes **operadores**:

- **Aritméticos**: + - * / % (resto de división entera)
- **Relacionales o de comparación:** == != < <= > >=
- **Lógicos**: && (AND) || (OR) ! (NOT)

Los operadores se aplican a los **operandos** (constante o variable, un elemento que contiene un valor a operar)

Una **expresión** es una combinación de operandos y operadores para obtener un resultado, como por ejemplo:

```
X=5+9-2;
x=y*z;
EsMayor=12>49;
esMayor=a>b;
puedo_jugar_online = tengo_PC && tengoWiFi
```

7. Conversiones de tipo

Cuando se asigna el valor de un tipo de datos a otro, los dos tipos pueden no ser compatibles entre sí. Si los tipos de datos son compatibles, entonces Java realizará la conversión (automáticamente) que se conoce como ***Conversión automática de tipos*** y, en caso negativo, deberá hacer un ***casting*** o convertirlo explícitamente.

Por ejemplo, la conversión de un int a float se puede realizar automáticamente pero si convertimos un float a entero, se pierden los datos decimales

byte ← short ← int ← long ← float ← doble

7. Conversiones de tipo

Conversión automática de tipos:

```
class Test
{
    public static void main(String[] args)
    {
        int i = 100;
        long l = i; //conversion automatica de tipo
        float f = l; //conversion automatica de tipo
        System.out.println("Valor Int "+i);
        System.out.println("Valor Long "+l);
        System.out.println("Valor Float "+f);
    }
}
```

Salida:
Valor Int 100
Valor Long 100
Valor Float 100.0

7. Conversiones de tipo

Sin embargo, hay conversiones que no pueden hacerse automáticamente y dan error:

```
class Test
{
    public static void main(String[] args)
    {
         char ch = 'c';
    int num = 88;
    ch = num;
    }
}
```

Salida:

9: error: incompatible types: possible lossy conversion from int to char
 ch = num;
 ^
1 error

7. Conversiones de tipo

Conversión explícita de tipos o casting:

```
class Test
{
    public static void main(String[] args)
    {
       double d = 100.04;
       long l = (long)d; //casting de tipo
       int i = (int)l;  //casting de tipo
       System.out.println("Valor Double "+d);
       System.out.println("Valor Long "+l);  //parte fraccionaria perdida
       System.out.println("Valor Int "+i);  //parte fraccionaria perdida
    }
}
```

Salida:
Valor Double 100.04
Valor Long 100
Valor Int 100

8. Comentarios

Los comentarios sirven para apoyar la documentación interna de los programas que desarrollamos. Si comentamos código, éste será ignorado y no se ejecutará. Podemos aplicar los siguientes comentarios:

- Comentarios de línea (***//***): sirven para comentar una única línea y se ha usado en los ejemplos anteriores
- Comentarios de bloque (***/* */***): sirven para comentar múltiples líneas de código (pueden haber comentarios de línea dentro de los de bloque):

```
/*
        Esta línea no se ejecuta ---- for(int i = 0; i <= 0; i++)
        Esto aun sigue siendo un comentario
        System.out.print("Hola"); //Este es otro comentario
*/
```

8. Comentarios

Además disponemos de comentarios de documentación en Java (**JavaDoc**) que además de documentar el código fuente sirven para documentar el proyecto.

Estos comentarios van entre ***/** y ***/*** y cada línea entre apertura y cierre del comentario se inicia con ***

Además estos comentarios incluyen opciones precedidas con @ para indicar los componentes del código fuente como @***param*** para indicar los parámetros de una función, @***return*** para indicar los valores de retorno, @***author*** para indicar el desarrollador del código, @***version*** para indicar la versión, etc.

8. Comentarios

```
package figuras;
/**
 * Una clase para representar círculos situados sobre el plano. Cada círculo queda
 * determinado por su radio junto con las coordenadas de su  centro.
 * @version 1.2, 24/12/20
 */
public class Círculo {
    protected double x,y; // coordenadas del centro
    protected double r;  // radio del círculo
    /**
     * Crea un círculo a partir de su origen su radio.
     * @param x La coordenada x del centro del círculo.
     * @param y La coordenada y del centro del círculo.
     * @param r El radio del círculo. Debe ser mayor o igual a 0.
     */
    public Circulo(double x, double y, double r) {
        this.x=x; this.y = y; this.r = r;
    }
     /**
     * Cálculo del área de este círculo.
     * @return El área (mayor o igual que 0) del círculo.
     */
     public double área() {
        return Math.PI*r*r;
     }
}
```

Ejercicios

- **Actividad propuesta para trabajar con variables y operadores (puedes usar un IDE online como https://www.online-java.com/ si no tienes otro disponible de momento en la clase)**