

Lenguaje JAVA

ALGORITMOS Y
ESTRUCTURAS
DE DATOS

1991 - Comienza Green Project:

Idea: desarrollar tecnología con facilidades de seguridad, programación distribuida, portable

1995 - Se cambia el nombre de OAK a JAVA

Surge oficialmente la tecnología JAVA

“Write once - run anywhere”

Características

- ❑ LENGUAJE DE ALTO NIVEL
- ❑ SIMPLE (Estricto)
- ❑ ORIENTADO A OBJETOS
- ❑ ROBUSTO y SEGURO
- ❑ ARQUITECTURA NEUTRAL
- ❑ MULTITHILO

JRE (Java Runtime Environment)

Entorno de ejecución que brinda un conjunto de componentes para ejecutar un programa Java que fue previamente compilado.

Contiene:

- JVM (Java Virtual Machine)
- Bibliotecas de integración
- Bibliotecas base
- Otros

JVM (Java Virtual Machine)

Los programas en JAVA se compilan e interpretan.

Cuando compilamos un archivo .java, el compilador Java genera un archivo .class (byte-code) con el mismo nombre del archivo. Este archivo .class entra en varios pasos cuando lo ejecutamos (JVM).

API (Application Programming Interface)

Bibliotecas de clases que facilitan el desarrollo en el lenguaje. La JVM y API deben ser consistentes entre sí y por este motivo, son distribuidas de modo conjunto.

Las clases en la Java API están separadas en packages

Hay clases que se pueden utilizar directamente en cualquier programa Java ya que se carga automáticamente (ej: String, System). Hay otras, que debemos indicar que requerimos su carga previa mediante una sentencia `import`.

API (Application Programming Interface)

Ejemplos de packages:

- java.lang
- java.util
- java.io
- javax.swing
- java.math

JDK (Java Development Kit)

Paquete de software que contiene todo lo necesario para desarrollar programas en JAVA y ejecutarlos. Es de distribución libre.

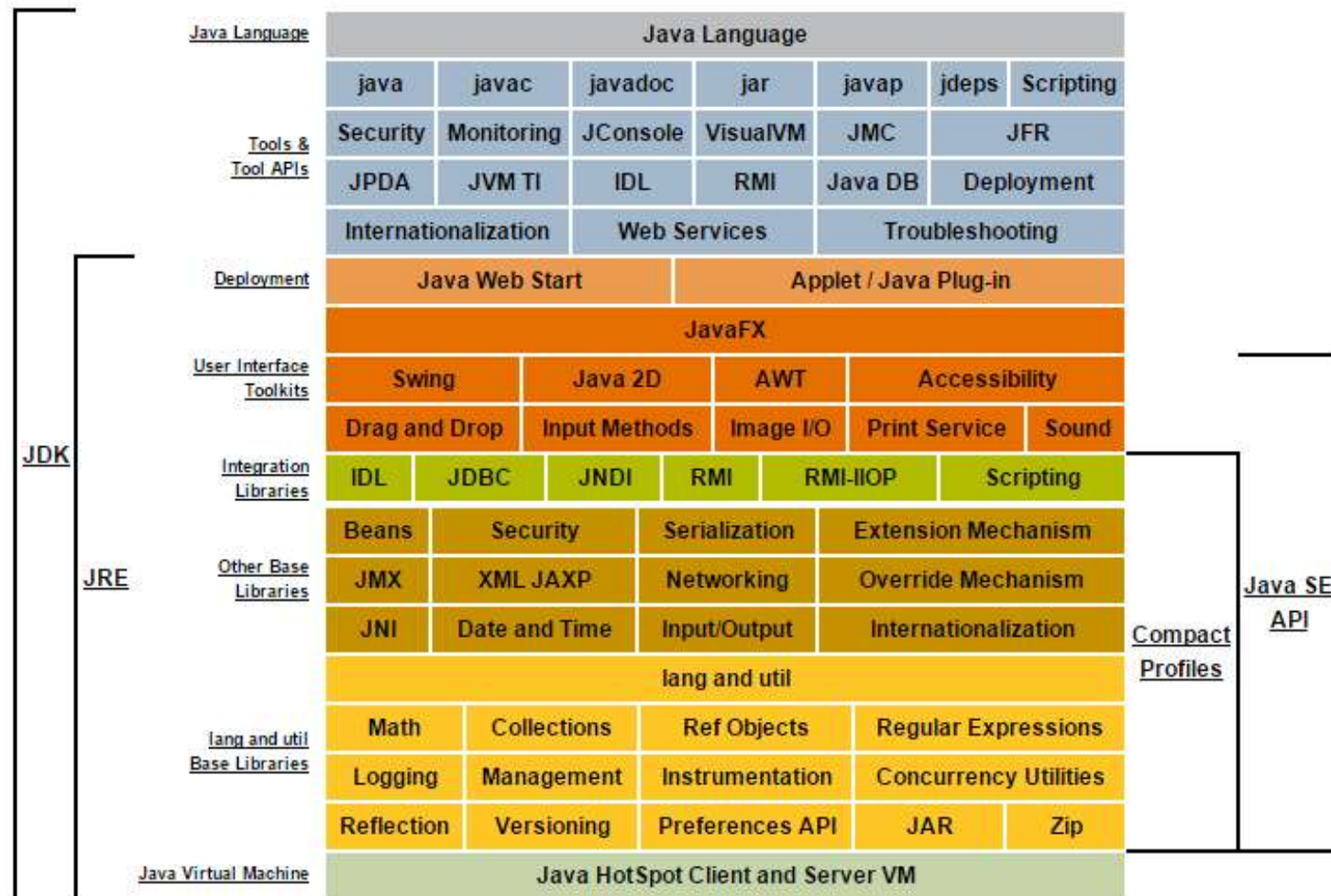
Descargas: <https://www.oracle.com/java/technologies/javase-downloads.html>

IDE (Integrated Development Environment)

Entorno de desarrollo que permite desarrollar software. En nuestro curso utilizaremos NetBeans.

Descarga: <https://netbeans.apache.org/>

Description of Java Conceptual Diagram



Estructura de un programa

- Un programa consiste en una o más clases
- Una clase contiene uno o más métodos
- Un método contiene sentencias
- Uno de los métodos debe ser el llamado main(), que es donde comienza la ejecución del programa

Escribir un programa que muestre en consola todos los números impares entre 1 y 50



Tipos de errores

- En tiempo de compilación
(compile-time errors)
- En tiempo de ejecución
(run-time errors)
- Lógicos
(logical errors)

Comentarios

- También llamados documentación en línea
- Los comentarios en Java pueden ser:
 - // Este comentario va hasta el final de la línea
 - /* El comentario va hasta el símbolo de terminación, pueden ser varias líneas
 - */

Principales tipos primitivos:

- int
- boolean
- char
- float
- double

El código de un programa está formado por:

- Identificadores
- Palabras reservadas
- Literales

Identificadores

- Son palabras usadas por el programador (por ejemplo, nombres de variables)
- Casi todos los identificadores no tienen significado predefinido
- Puede formarse de letras, dígitos, underscore (`_`), y `$`
- Java es *case sensitive*, por ej. `Total` y `total` son diferentes identificadores

Palabras reservadas

- Son identificadores que tienen significado especial en Java y no se pueden usar para otra cosa
- Ejemplos:
for, if, while, public, private, protected, class, int, long, double, boolean, etc

Literales

- Es un valor explícito usado en el programa

- Literales enteros:

25 69 -4288

- Literales con decimales (punto flotante):

3.14159 42.075 -0.5

- Literales String:

"El resultado es: "

"Este texto sale tal cual en pantalla"

Uso del operador +

El operador + tiene diferentes propósitos:

- Si se aplica a strings, da uno nuevo
“esto es una” + “prueba” → “esto es una prueba”
- Si se aplica a un string y otro valor, el valor es convertido a string y se unen
“El máximo valor es ”+max → El máximo valor es 10
- Si son dos números, suma
34 + 45 → 79

Operadores lógicos

!	NOT
&&	AND
	OR

- Toman operandos booleanos (verdadero o falso)
- El resultado es también booleano

Tablas de verdad

- Operador NOT

a	!a
true	false
false	true

- Operador AND

El resultado es verdadero SOLO cuando ambos operadores lo son

a	b	a&& b
true	true	true
true	false	false
false	true	false
false	false	false

- Operador OR

El resultado es falso SOLO cuando ambos operadores lo son

a	b	a b
true	true	true
true	false	true
false	true	true
false	false	false

Operadores de asignación

Operador	Ejemplo	Equivalente a
<code>+=</code>	<code>x+=y</code>	<code>x=x+y</code>
<code>-=</code>	<code>x-=y</code>	<code>x=x-y</code>
<code>*=</code>	<code>x*=y</code>	<code>x=x*y</code>
<code>/=</code>	<code>x/=y</code>	<code>x=x/y</code>
<code>%=</code>	<code>x%=y</code>	<code>x=x%y</code>

Incremento y decremento

- Operador incremental (++)
 - Suma uno al operador int o float
- Operador decremental (--)
 - Resta uno
- Ejemplo
 - La sentencia:
 - `cont++;`
 - Es equivalente a
 - `cont = cont + 1;`
- Se puede usar en prefix (antes de la variable) o postfix (luego de la variable)

Incremento y decremento

- Cuando se usan solas en una sentencia son equivalentes:
 - `cont++;`
 - `++cont;`
- Cuando se usan en una expresión mayor, tienen diferente efecto.
- En ambos casos, la variable se incrementa o decrementa.
- Pero el valor usado en la expresión mayor depende:

Expresión	Operación	Valor en la Expresión
<code>cont++</code>	Suma 1	Valor anterior
<code>++cont</code>	Suma 1	Valor nuevo
<code>cont--</code>	Resta 1	Valor anterior
<code>--cont</code>	Resta 1	Valor nuevo

Estructuras

IF

- Sintaxis

```
if (condición){  
    sentencias;  
} else {  
    sentencias;  
}
```

- Ejemplos

```
if (a>0){  
    a=a+3;  
}
```

```
if(a==b){  
    System.out.println("Son iguales");  
} else {  
    System.out.println("Son distintos");  
}
```

= asigna
== compara

Estructuras iterativas

FOR

- Sintaxis

```
for (inicialización; condición; incremento){  
    sentencias;  
}
```

- Ejemplos

```
for (int cont=1; cont<75; cont=cont+1){  
    System.out.println(cont);  
}
```

```
for (int num=5; num<=100; num=num*2){  
    if(num%3==0){  
        System.out.println(num + " es multiplo de 3");  
    }else{  
        if(num%7==0)  
            System.out.println(num + " es multiplo de 7");  
    }  
}
```

No es necesarios los corchetes
(llaves) ya que es una
sentencia sola

Estructuras iterativas

WHILE

- Sintaxis

```
inicialización;  
while (condición){  
    sentencias;  
    incremento;  
}
```

- Ejemplos

```
int num = 0;  
int porcentaje = 100;  
  
while (num < porcentaje && porcentaje !=0){  
    ...  
    num++;  
    porcentaje=porcentaje-num;  
}
```

!= compara si son distintos

Estructuras iterativas

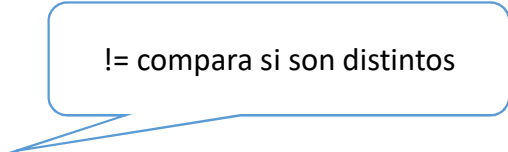
- WHILE

- Sintaxis

```
inicialización;  
while (condición){  
    sentencias;  
    incremento;  
}
```

- Ejemplos

```
int num = 0;  
int porcentaje = 100;  
  
while (num < porcentaje && porcentaje !=0){  
    ...  
    num++;  
    porcentaje=porcentaje-num;  
}
```



!= compara si son distintos

String

Es una clase del lenguaje

Un string es una secuencia de caracteres (letras, números, símbolos)

NO es un tipo primitivo de Java, es tipo Class

La clase String se encuentra en el package java.lang (ya vimos que este package se importa automáticamente)

Son especiales porque se pueden crear de dos formas:

```
String s1 = "Ana";// igual a los tipos básicos del lenguaje
```

```
String s1 = new String ("Ana");
```

String

Comparaciones: `s1 == s2`

- compara referencias, no valor
- En objetos, no se puede usar `==`, se debe usar el método `equals()`
- En String, la equivalencia se puede preguntar:
 - `s1.equals(s2)`
 - `s1.equalsIgnoreCase(s2)`
 - `s1.compareTo(s2)`
 - compara lexicográficamente:
 - `<0`: `s1` va antes `s2`
 - `=0`: `s1` igual `s2`
 - `>0`: `s1` va después `s2`

String: Ejemplo

Definiendo	Comparaciones	Resultado
String s1, s2, s3;	s1 == s2	TRUE
s1 = "Ana";	s1 == s3	FALSE
s2 = "Ana";	s1.equals(s2)	TRUE
s3 = new String ("Ana");	s1.equals(s3)	TRUE
String s4;	s1.compareTo(s2)	0
s4 = s3;	s1.compareTo(s3)	0
s5 = "Ana";	s4 == s3	TRUE
	s5 == s3	FALSE
	s5 == s1	TRUE

s3 y s4 son alias

Visibilidad

- `public`

los métodos y variables declarados públicos pueden ser accedidos desde cualquier lugar.

- `private`

los métodos y variables declarados privados solo pueden ser accedidos en la clase en que se declararon.

- `protected`

son accesibles desde todas las clases del mismo package.

Práctico 1

