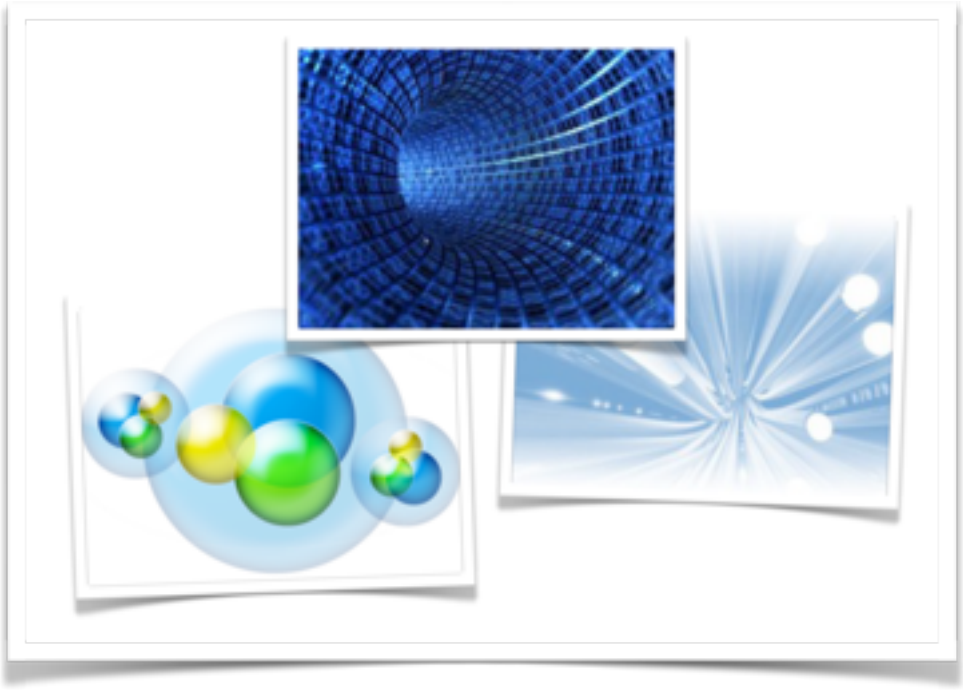
Taller

Programación Orientada a Objetos

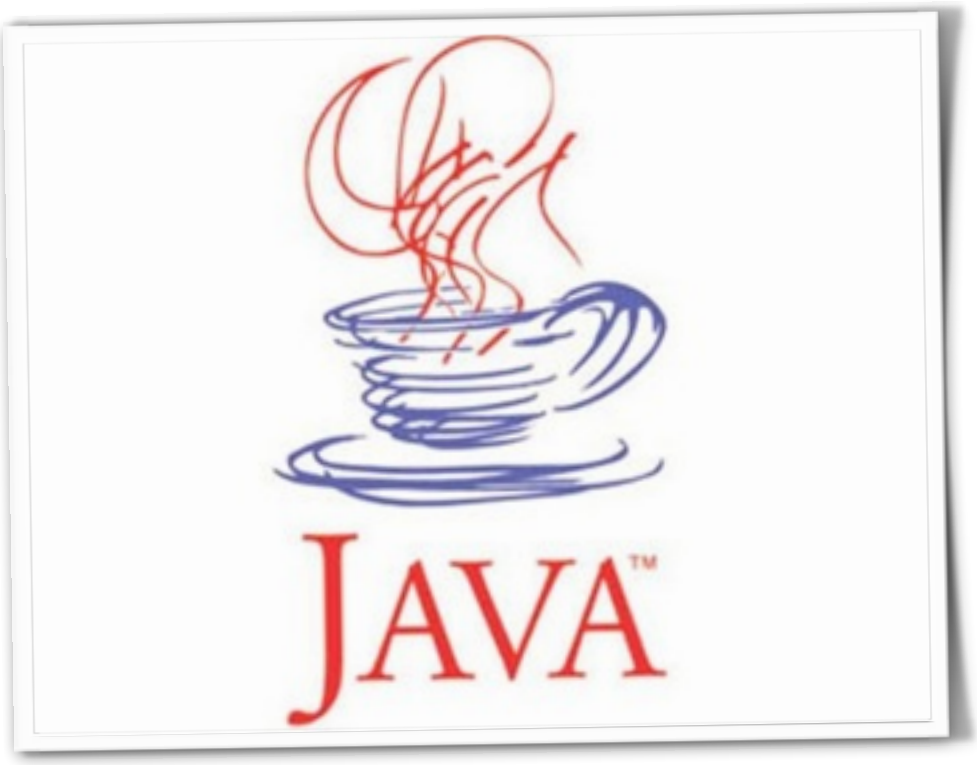


Un lenguaje orientado a objetos
Julio, 2014
Guadalajara, Jalisco. México

UDGVIRTUAL

Prof. Vega Maldonado Sabrina Lizbeth

Lenguaje Java



El lenguaje de Java nos permite trabajar con la programación orientada a objetos desde un inicio ya que es un lenguaje 100% orientado a objetos. Este lenguaje es multi plataforma, es decir nos permite trabajar en cualquier sistema operativo (por supuesto que tenga instalado el lenguaje jdk) ya que tiene una máquina virtual, la cual es un interprete.

La máquina virtual, es una máquina abstracta de computo. Tiene su propio conjunto de instrucciones y su propia memoria en tiempo de ejecución.

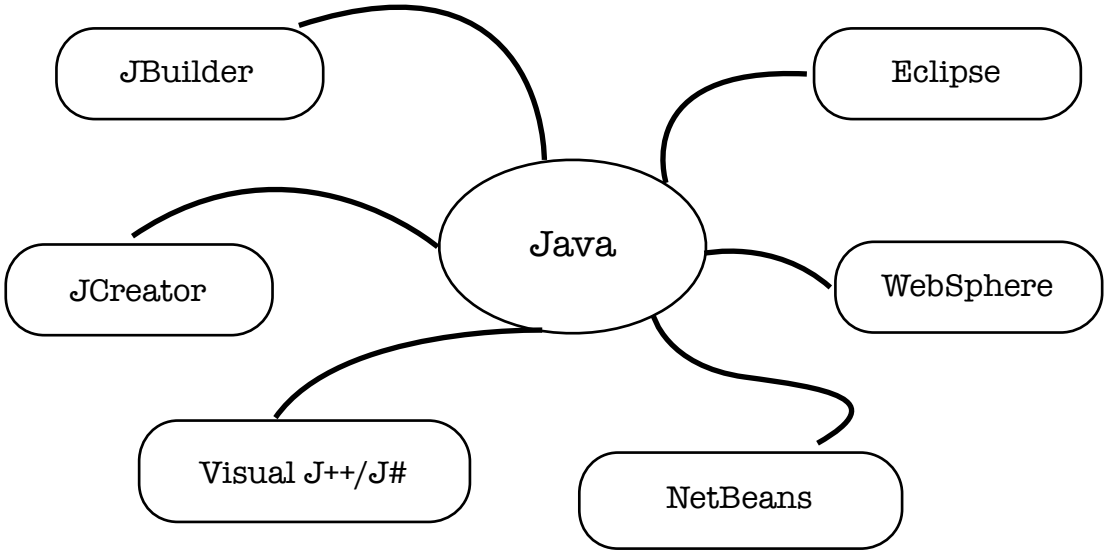
Por lo tanto java es un lenguaje interpretado, es decir, crea un programa en un formato especial conocido como bytecode (código binario) para que el procesador lo entienda y a su vez este formato es ejecutado por un interprete (la máquina virtual).

Este lenguaje está basado en C++, es simple y claro en el manejo de conceptos orientados a objetos. A crecido en numerosas tecnologías tales como servers, programación para móviles, programación para internet etc.

Un ambiente integrado de programación en java debe tener:

- Un editor
- Un cargador de programa (interprete)
- Un ejecutor de clases

Existen editores que se les puede configurar el path para acceder tanto al compilador como al interprete, algunos son:



Trabajando desde Consola

En consola se deberá acceder a donde está el compilador, generalmente se encuentra en el siguiente path:

c:\Archivos de programa\java\jdk1.6.0_07\bin

En esta ruta guardar el archivo con la extensión *.java por ejemplo Mensaje.java.

Dicho archivo se puede escribir en cualquier editor (cerciorandose que se grabe únicamente con la extensión .java)

Del editor pasar a consola y en la ruta descrita anteriormente ejecutar el comando para compilar de la siguiente manera:

c:\Archivos de programa\java\jdk1.6.0_07\bin> **javac Mensaje.java**

Si hay errores aparecerá el posible error y la línea donde se encuentra (regresar al editor y corregir). Si no hay errores aparecerá de nuevo el path, entonces activar el comando de ejecución de la siguiente manera:

c:\Archivos de programa \java\jdk1.6.0_07\bin> **java Mensaje**

El archivo a escribir en el editor es el siguiente:

```
class Mensaje {
    public static void main (String arg[]){
        System.out.println("Mi mensaje");
    }
}
```

Trabajando con JCreator

Para trabajar en JCreator es necesario abrir el programa desde el escritorio del sistema operativo en el que te encuentres.

Ir al menú File, opción New, opción File (notese que también se pueden crear proyecto o áreas de trabajo, en donde un proyecto es un conjunto de archivos relacionados entre sí y un área de trabajo es un conjunto de proyectos).

Una vez que se tiene un nuevo archivo se escribe el código descrito anteriormente.

Se compila con el ícono correspondiente (build)

Se ejecuta con los ícono correspondiente (run).

Recordando que previamente se ha configurado en dicho editor el path para acceder al compilador y traductor.

Nota: Para trabajar con eclipse el procedimiento es el mismo, tomando en cuenta que lo que se debe crear es un nuevo proyecto aunque se tenga un solo archivo (en eclipse no se pueden crear archivos independientes como en JCreator).

Mi primer programa

```
class Mensaje {
        public static void main (String arg[]) {
                System.out.println("mi mensaje");
        }
}
```

Paso 1: Escribir en un editor el programa y guardarlo con el nombre de:

Mensaje.java

Paso 2: en consola compilar con el comando javac:

javac Mensaje.java

Paso 3: en consola (si no hay errores) ejecutar con el comando java:

java Mensaje

Glosario

Abstracción: La abstracción en la programación orientada a objetos consiste en separar las características de los objetos para poder analizar su escénica y determinar lo que se va a programar del objeto dependiendo de los requerimientos de la solución del problema computable

Encapsulamiento: Se puede definir como encerrar las características del objeto tanto físicas como de comportamiento en una clase, la finalidad es poder ocultar los detalles de la implementación y de esta manera programar un objeto.

Objeto: entidad de la realidad con características que lo diferencian de otras entidades.

Clase: conjunto de objetos con las mismas características.

Atributos: son las características físicas del objeto, las cuales se obtienen de una abstracción el mismo en un primer nivel.

Métodos: son las características de comportamiento del objeto, éstas se obtienen con una abstracción detallada.

Esquema preliminar de clases: es el símbolo que se utiliza al estar diseñando una clase para programar un objeto en cualquier lenguaje de programación orientad a objetos.

Sintaxis

La sintaxis de un lenguaje es la forma general de escribir las sentencias, entendiendo como sentencia el elemento más simple de la programación. Una sentencia se convertirá en una instrucción específica en el lenguaje. Todas las instrucciones en un lenguaje tienen elementos que no cambiarán y elementos que cambiarán dependiendo de su aplicación. En este caso los elementos que cambian de las instrucciones se representarán dentro de los símbolos de menor que y mayor qué; los elementos que no cambian se escribirán tal como aparecen en la sintaxis. Los corchetes representan que ese elemento es opcional.

Sintaxis de clase en java:

[<tipoAcceso>] class <Identificador > {

<sentencia 1> ;

...

<sentencia n> ;

}

Sintaxis de un atributo en java:

<tipoAcceso > <tipoDato> <identificador> ;

Sintaxis de un método que regresa valor en java:

<tipoAcceso> <tipoDato> <identificador> ([<parámetros>])

{

<sentencia 1> ;

...

<sentencia n> ;

}

Sintaxis de la llamada de un método que regresa valor en java:

<variable> = <instancia> . <identificador> ([<argumentos>]) ;

Sintaxis de un método que no regresa valor en java:

<tipoAcceso> void <identificador> ([<parámetros>])

{

<sentencia 1> ;

...

<sentencia n> ;

}

Sintaxis de la llamada de un método que no regresa valor en java:

<instancia> . <identificador> ([<argumentos>]) ;

Esquema preliminar de clases:

<Identificador>
<atributo 1> <atributo n>
<método 1> <método n>

Tipos de Acceso para una clase en el lenguaje de java

Tipo	descripción	código java

publica	Es la clase que contiene a la función principal	public class <Identificador> {
		
		}

abstracta	Es la clase que contiene métodos abstractos y se usa como molde para las subclases	abstract class <Identificador> {
		
		}

final	Es la clase que no puede contener subclases	final class <Identificador> {
		
		}

Nota: aquí no existe un tipo de acceso por default y las clases pueden no tener tipo de acceso

Tipos de Acceso para atributos y métodos en el lenguaje de java

Tipo	descripción	símbolo	código java
publico	desde cualquier lugar puedo mandar llamar al atributo y/o método público	+	public
privado	sólo dentro de la clase a la que pertenece (miembro)pude mandar llamar el atributo y/o método privado	-	private
protegido	se puede mandar llamar desde la misma clase a la que pertenece y desde sus subclases (herencia) los atributos y/o métodos protegidos	#	protected
amigable	es el tipo de acceso por defaúlt, es decir si no se pone ningun tipo de acceso el atributo o método se vuelve a migable puede ser accedido desde cualquier clase que este dentro del mismo paquete (conjunto de clases)		

Tipos de Datos en el lenguaje de java

Tipo	Requisito de almacenamiento	Rango
int	4 bytes (con signo)	-2.147.483.648 a 2.147.483.647
short	2 bytes (con signo)	-32.768 a 32.757
long	8 bytes (con signo)	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
byte	1 byte (con signo)	-128 a 127
float	4 bytes (con signo)	3.4 e^{-308}
dobule	8 bytes (con signo)	1.7 e^{-308}
char	2 bytes (sin signo)	\u0000 a \uFFFF (caracter Unicode)

Fuente: “Programación en C, C++, Java y UML” por Luis Joyanes Aguilar e Ignacio Zahonero Martínez, Ed. McGrawHill, 2010

Explicación de cada parte del primer programa en java:

```
1. class Mensaje {
2.       public static void main(String arg[] ) {
3.                System.out.println(“Mi mensaje”);
4.       }
5. }
```

1. definición de la clase
2. definición de la función principal
3. llamada de la función println a través de la instancia System.out que se encuentra en la clase PrintStream
4. fin de la función principal
5. fin de la clase

Elementos de la función principal en java:

```
public static void main (String arg[]){
      System.out.println("Mi mensaje");
}
```

public: es necesario que la función principal pueda ser vista y accedida por todos.

static: significa que se va a memoria desde tiempo de compilación, esto es necesario porque al ejecutarse lo primero que busca en la memoria es la función principal.

void: no regresa valor, en java la función principal siempre es de tipo de dato void

main: es el identificador de la función principal

(String arg[]): es el parámetro de la función (por fuerza debe declararse aunque no se use), y siempre tiene que ser un arreglo de tipo de dato cadena, el identificador del parámetro el programador lo puede elegir.

Nota: en el lenguaje de java todo es dinámico por default, es decir, se almacena en memoria hasta tiempo de ejecución.