INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA Y CIENCIAS SOCIALES Y ADMINISTRATIVAS

Material para el curso de LENGUAJE JAVA

Contenido

Objetivo General

Los participantes conocerán el lenguaje Java, su ambiente y herramientas de desarrollo, para su utilización en aplicaciones sencillas.

Asimismo, se familiarizaran con el paradigma de la Programación Orientada a Objetos.

Prerrequisitos

- Experiencia en Programación
- Conocimiento de algún lenguaje, tal como C o C++

Introducción

¿Qué es Java?

Es un Lenguaje de Programación Orientado a Objetos desarrollado por Sun Microsystems.

Los navegadores o browsers utilizan documentos escritos en HTML. La combinación actual de navegadores HTML/WWW está limitada a texto y gráficos, si se quiere reproducir un sonido o ejecutar un programa hay que descargarlo primero en el ordenador y posteriormente con ayuda de algún programa ejecutarlo lo cual no le da dinamismo al documento.

El lenguaje Java y los navegadores que lo soportan, dan la posibilidad de ejecutar programas, reproducir sonido directamente desde el navegador, etc. Java se basa en la filosofía NC (Network Computer), esto es, los programas residen en un servidor que los envía cuando se los solicitan.

Características de Java

- Simple. Java se diseño con base en C++ que es un lenguaje que se ha difundido mucho, lo cual facilita un rápido y fácil aprendizaje. Sin embrago, no es una extensión de él, sino que fue escrito completamente desde cero. Java elimina de las especificaciones del lenguaje algunas características como:
  - ✓ Apuntadores explícitos ni aritmética de punteros.
  - ✓ Registros (*struct*) ni *union*.
  - ✓ Definición de Tipos (*typedef*).
  - ✓ Macros ni preprocesador de código (*#include, #define*, etc.)
  - ✓ Necesidad de liberar memoria. (existe el *garbage collector* o reciclador de memoria dinámica)
  - ✓ Variables o funciones globales. Todas ellas pertenecen a algún objeto
  - ✓ Dependencias de la implementación (plataforma)
  - ✓ *goto*…
- Orientado a Objetos. Soporta las características propias del paradigma de la orientación a objetos: Encapsulación, herencia y polimorfismo. Las elementos con las que cuenta son: clases y sus instancias (objetos), estas últimas necesitan ser construidas y destruidas como en C++.
- Distribuido. Java se ha construido con extensas capacidades de interconexión TCP/IP. Accede e interactúa con protocolos como http y ftp. Esto permite accesar a la información a través de la red con tanta facilidad como archivos locales. Java en si no es distribuido, sino que proporciona librerías y herramientas para que los programas puedan ser distribuidos.
- Robusto. Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como tiempo de ejecución. Además para asegurar el funcionamiento de la aplicación, realiza una verificación de los ByteCodes, que son el resultado de la compilación de un programa Java.
- Arquitectura Neutral. El compilador Java compila su código a un archivo objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, independientemente de la máquina en que ha sido generado.
- Seguro. La seguridad en Java tiene dos facetas, en el lenguaje se eliminan punteros para prevenir el acceso ilegal a memoria. El código pasa muchas comprobaciones antes de ejecutarse. El Cargador de Clases separa el espacio de nombres del sistema de archivos local del de los recursos procedentes de la red.

- Portable. Además de la portabilidad básica por ser de arquitectura independiente, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que estás puedan ser implantadas en entornos Unix, PC o Mac. También implementa estándares de portabilidad para facilitar el desarrollo.
- Interpretado. El intérprete Java (sistema *run-time*) puede ejecutar directamente el código objeto. Enlazar (linkear) un programa normalmente consume menos recursos que compilarlo, por lo que los desarrolladores con Java pasarán más tiempo desarrollando y menos esperando por el computador.
- Multitarea. Al ser multitarea o multihilo, Java permite realizar muchas actividades simultáneas en un programa. Las tareas son básicamente pequeños procesos o piezas independientes de un gran proceso, al estar construidas en el mismo lenguaje, son más fáciles de usar y más robustos que sus homólogas en C o C++.

Máquina Virtual Java

El compilador de Java únicamente genera el denominado ByteCode. Este código es un código intermedio entre el lenguaje máquina del procesador y Java. Evidentemente este código no es ejecutable por sí mismo en ninguna plataforma hardware, pues no se corresponde con el lenguaje de ninguno de los procesadores que actualmente se conocen. El mecanismo que permite ejecutar el ByteCode es denominada Máquina Virtual Java (JVM).

Herramientas JDK

El *Java Development Kit* no es un IDE sino un conjunto de utilerías:

- ✓ javac. Compila código fuente de Java.
- ✓ java. Interprete, ejecuta aplicaciones en forma local.
- ✓ appletviewer. Despliega applets de java localmente usando un archivo HTML.
- ✓ javap. Descompila archivos .class.
- ✓ javadoc. Produce documentación en HTML.
- ✓ jdb. Es un rudimentario debbuger.
- ✓ javah. Utilería para generar archivos de interfase con lenguaje C.

Los archivos fuente en Java terminan con la extensión ".java". Estos pueden ser creados utilizando cualquier editor que tenga salida a archivo de texto plano o ASCII.

Compilación

El compilador de Java traslada el código fuente Java a byte-codes. Al compilar un archivo con extensión .java se genera al menos un archivo con el mismo nombre pero con extensión .class. Esto es posible con la siguiente línea de comando:

% javac NombreArchivo.java

Para ejecutar una aplicación se recurre al intérprete java, con la siguiente línea de comando:

% java NombreArchivo

El primer programa

```
public class HolaMundo {
        public static void main (String args [ ])    {
                System.out.println (“Hola Mundo”);
        }
}
```

Debe capturarse exactamente como está escrito, y guardarse en un archivo llamado HolaMundo.java

Comentarios

En Java hay tres tipos de comentarios:

1. Comentarios para una sola línea. //
2. Comentarios de una o más líneas. /* */
3. Comentario de documentación, de una o más líneas. /** */

Tipos de Datos Primitivos

Tipo	Precisión	Valor por Defecto
byte	8 bits	0
short	16 bits	0
int	32 bits	0
long	64 bits	0
char	16 bits	\u0000
float	32 bits	+0.0f
double	64 bits	+0.0d
boolean	64 bits	false

Identificadores

Los identificadores nombran variables, funciones, clases y objetos; cualquier cosa que el programador necesite identificar o usar.

Sintaxis.

tipo identificador [= valor][,identificador [= valor] ...];

Palabras Reservadas

abstract	continue	for	new	switch
boolean	default	goto	null	synchronized
break	do	if	package	this
byte	double	implements	private	threadsafe
byvalue	else	import	protected	throw
case	extends	instanceof	public	transient
catch	false	int	return	true
char	final	interface	short	try
class	finally	long	static	void
const	float	native	super	while

Operadores y Precedencia

Operadores postfijos	[ ]	.	(parámetros)			
		exp++	exp--			
Operadores unarios	++exp		–exp	+exp	–exp	~
		!				
Creación o cast	new		(tipo) exp			
Aritméticos	*	/	%			
	+	-				
Cambio	<<	>>	>>>			
Relacionales	<	>	<=	>=	instanceof	
Igualdad	==	!=				
Nivel de Bits	&	∧	|			
Lógicos	&&	||				
Condicional	?	:				
Asignación	=	*=	/=	%=	+=	-=
		>>=	<<=	>>>=	&=	∧=
		!=				

CONTROL DE FLUJO

Decisiones

if/else

if(expresión-boolean) {

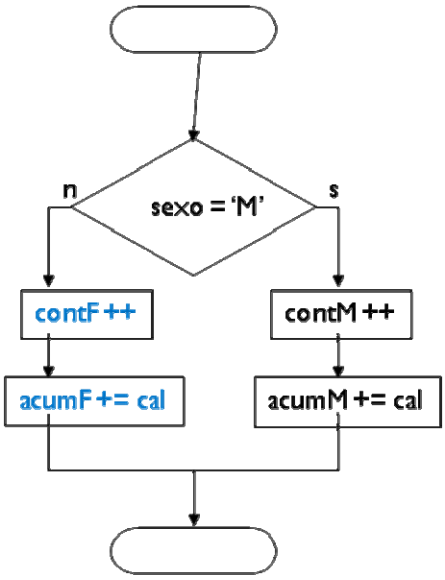
sentencias;

}

[else {

sentencias;

}]




```
{
        if (sexo == 'M') {
                contM++;
                acumM += cal;
        }
        else {
                contF++;
                acumF += cal;
        }
}
```

switch

```
switch( expresión ) {
        case valor1:
                sentencias;
        break;
        case valor2:
                sentencias;
        break;
        [default:
                sentencias;]
}
```

```
{ switch (edo_civil) {
        case 'C': case 'c':
                 System.out.println("Casado");
                break;
        case 'S': case 's':
                 System.out.println("Soltero");
                break;
        case 'V': case 'v':
                 System.out.println("Viudo");
                break;
        default: System.out.println("Opción invalida");
        }
}
```

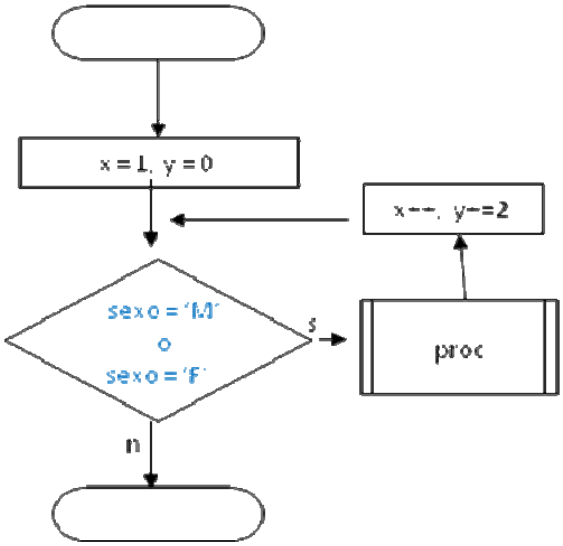
Ciclos

for

for(inicialización; terminación; incremento) {

sentencias;

}



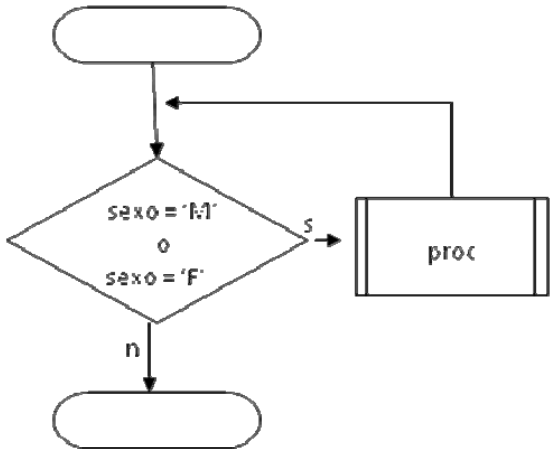

```
{
        for (x = 1, y = 0; sexo == 'M' || sexo == 'F';
                x++,  y+=2 )
                proc();
}
```

while

while(terminación-expresión-booleana) {

sentencias;

}



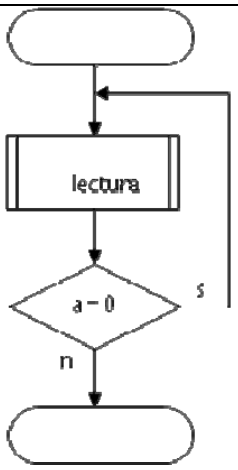

```
{
while (sexo == 'M' || sexo == 'F')
        proc();
}
```

do/while

do{

sentencias;

} while(terminación-expresión-booleana);




```
{
        do
                lectura();
        while (a == 0);
}
```

Objetos

Un objeto es la instanciación de una clase y tiene un estado y un funcionamiento. El estado está contenido en sus variables miembro, y el funcionamiento viene determinado por sus métodos. Las variables miembro pueden ser variables de instancia o variables de clase.

Sintaxis general para crear un objeto:

Clase NombreDelObjeto = new constructor ();

Se activa el funcionamiento de un objeto invocando a uno de sus métodos, que realizará una acción. La creación de un objeto se realiza en los siguientes pasos:

- Declaración, proporcionar un nombre al objeto.
- Instanciación, asignar memoria al objeto.
- Inicialización, opcionalmente se pueden proporcionar valores iniciales a las variables de instancia del objeto.

Para acceder a las variables o a los métodos de un objeto, se especifica el nombre del objeto y el nombre de la variable, o método, separados por un punto (.). Por ejemplo:

miObjeto.variable;

miObjeto.metodo ();

Clases

Son lo más simple de Java. Todas las acciones de los programas Java se colocan dentro del bloque de una clase o un Objeto. La sintaxis de una clase es:

[modificador] class <NombreClase> [extends < NombreClase 1>] [implements <interfaceNames*>] {*

. . . }

Modificadores – Una clase puede ser: abstract, final, public (o sin modificador se le llama package)

extends – Nombre de la superclase.

implements - Nombre de las interfaces implementadas por esta clase.

Un objeto es una instancia de una clase. Todos los métodos se definen dentro del bloque de la clase. Todo en Java forma parte de una clase, es una clase o describe cómo funciona una clase.

Java no soporta funciones o variables globales. Todos los datos básicos se deben declarar en las clases antes de hacer uso de ellos. Pocas sentencias se pueden colocar fuera del bloque de una clase, la palabra clave import es una de ellas.

La declaración de la clase indica al compilador el nombre de la clase, la clase de la que deriva (su superclase), los privilegios de acceso a la clase y si la clase implementa o no, una o varias interfaces.

Por convención los nombres de las clases empiezan con una letra mayúscula. Cada clase Java deriva, directa o indirectamente, de la clase Object.

Los miembros de una clase son los atributos y los métodos. Los primeros son las variables que almacenan información sobre el estado y otras propiedades de cada objeto.

Los métodos son funciones que pueden ser llamadas dentro de la clase o por otras clases. La implementación de un método consta de dos partes, una declaración y un cuerpo. Sintaxis:

```
especificadorAcceso tipoRetorno nombreMetodo (lista_de_argumentos) {
        instrucciones;
}
```

A la hora de la declaración de un método, hay que indicar el tipo de dato que ha de devolver. Si no devuelve ningún valor, se debe indicar el tipo void como retorno.

Los argumentos son como variables locales declaradas en el cuerpo del método que están inicializadas al valor que se pasa como parámetro en la invocación del método.

En Java todos los argumentos de tipos primitivos se pasan por valor, mientras que los objetos se pasan por referencia.

Abstracción

Consiste en ver las cosas desde una perspectiva más amplia y general, que ignora los elementos no esenciales para nosotros. En otras palabras, es el proceso de encontrar patrones.
En programación, quiere decir ignorar los aspectos de los objetos del mundo real que no nos conciernen y enfocarnos en aquellos que si lo hacen.
Esto nos permite escribir código más simple y genérico, y por tanto, reutilizable.

Encapsulamiento

Es la técnica de ocultar o empaquetar los datos de un objeto, permitiendo el acceso a ellos solamente a través de las funciones o métodos del propio objeto, a efecto de protegerlos de ser vistos o alterados desde el exterior.

Herencia

Es el mecanismo por el que se crean nuevos objetos definidos en términos de objetos ya existentes.

La palabra clave *extends* se usa para generar una subclase (especialización) de un objeto. Se pueden sustituir los métodos proporcionados por la clase base. Java se diseño con la idea de que fuera un lenguaje sencillo y, por tanto, se le denegó la capacidad de la herencia múltiple. Sintaxis:

```
class nombreDeLaClase extends SuperClase {

}
```

Subclases.

Son aquellas que se crean a partir de otras existentes. Las clases de las que se hereda se llaman Superclases.

Cualquier objeto de la subclase contiene todas las variables y todos los métodos de la superclase y sus antecesores.

Todas las clases en Java derivan de alguna clase anterior. La clase raíz del árbol de la jerarquía de clases de Java es la clase *Object*. Cada vez que se desciende en el árbol de jerarquía, las clases van siendo más especializadas.

Arreglos

Son un tipo de datos homogéneo, de posiciones secuenciales.

Su sintaxis general es:

tipoDeElementos nombreDelArray [ ] =

new tipoDeElementos [tamañoDelArray]

Todos los arrays son objetos y tienen una variable miembro: length, que se utiliza para conocer la longitud del array. Su sintaxis es:

array.length;

Java dispone de comprobaciones exhaustivas para la correcta manipulación de los arrays, por ejemplo, de la comprobación de sobrepasar los límites definidos para el array. Un array hay que declararlo antes de poder utilizarlo. En la declaración hay que incluir el nombre del array y el tipo de datos que se van a almacenar en él. Se pueden declarar en Java arrays de cualquier tipo:

char s [ ];

int arreglo [ ];

Incluso se pueden construir arrays de arrays:

int tabla [ ] [ ] = new int [4] [5];

Para crear un array en Java hay dos métodos básicos:

1. Array vacío: *int lista[ ] = new int[50];*
2. Array con valores iniciales: *String nombres[ ] = {"Juan", "Pepe", "Pedro"};*

Bibliografía

"Java Básico"
Capacitación Integral Y Servicios Avanzados de Cómputo

"Thinking in Java"
Bruce Eckel
Prentice Hall

"Java 2, Manual de Programación"
Luis Joyanes Aguilar, Matilde Fernández Azuela
Osborne Mc Graw Hill

"El lenguaje de programación JAVA"
Francisco Javier Ceballos
Alfaomega RAMA

http://java.sun.com/j2se/1.6.0/docs/api/

http://www.javasoft.com

http://www.sun.com/sunworldonline