Elementos básicos de un lenguaje Orientado Objetos

UACM SLT

M.C. José Castillejos López

El lenguaje Java

- Como cualquier lenguaje de programación, el lenguaje Java tiene su propia estructura, reglas de sintaxis y paradigma de programación. El paradigma de programación del lenguaje Java se basa en el concepto de programación orientada a objetos (POO), que las funciones del lenguaje soportan.
- El lenguaje Java es un derivado del lenguaje C, por lo que sus reglas de sintaxis se parecen mucho a C: por ejemplo, los bloques de códigos se modularizan en métodos y se delimitan con llaves ({ y }) y las variables se declaran antes de que se usen.

- Estructuralmente, el lenguaje Java comienza con *paquetes*. Un paquete es el mecanismo de espacio de nombres del lenguaje Java. Dentro de los paquetes se encuentran las clases y dentro de las clases se encuentran métodos, variables, constantes, entre otros

El compilador Java

- Cuando usted programa para la plataforma Java, escribe el código de origen en archivos .java y luego los compila. El compilador verifica su código con las reglas de sintaxis del lenguaje, luego escribe los *códigos byte* en archivos .class. Los códigos byte son instrucciones estándar destinadas a ejecutarse en una Java Virtual Machine (JVM). Al agregar este nivel de abstracción, el compilador Java difiere de los otros compiladores de lenguaje, que escriben instrucciones apropiadas para el chipset de la CPU en el que el programa se ejecutará.

La JVM(Java Virtual Machine)

- Al momento de la ejecución, la JVM lee e interpreta archivos .class y ejecuta las instrucciones del programa en la plataforma de hardware nativo para la que se escribió la JVM. La JVM interpreta los códigos byte del mismo modo en que una CPU interpretaría las instrucciones del lenguaje del conjunto. La diferencia es que la JVM es un software escrito específicamente para una plataforma particular. La JVM es el corazón del principio "escrito una vez, ejecutado en cualquier lugar" del lenguaje Java. Su código se puede ejecutar en cualquier chipset para el cual una implementación apropiada de la JVM está disponible. Las JVM están disponibles para plataformas principales como Linux y Windows y se han implementado subconjuntos del lenguaje Java en las JVM para teléfonos móviles y aficionados de chips.

El recolector de basura

- En lugar de forzarlo a mantenerse a la par con la asignación de memoria (o usar una biblioteca de terceros para hacer esto), la plataforma Java proporciona una gestión de memoria lista para usar. Cuando su aplicación Java crea una instancia de objeto al momento de ejecución, la JVM asigna automáticamente espacio de memoria para ese objeto desde el *almacenamiento dinámico*, que es una agrupación de memoria reservada para que use su programa. El *recolector de basura* Java se ejecuta en segundo plano y realiza un seguimiento de cuáles son los objetos que la aplicación ya no necesita y recupera la memoria que ellos ocupan. Este abordaje al manejo de la memoria se llama *gestión de la memoria implícita*porque no le exige que escriba cualquier código de manejo de la memoria. La recogida de basura es una de las funciones esenciales del rendimiento de la plataforma Java.

El kit de desarrollo de Java

- Cuando usted descarga un kit de desarrollo de Java (JDK), obtiene, — además del compilador y otras herramientas, — una librería de clase completa de programas de utilidad preconstruidos que lo ayudan a cumplir cualquier tarea común al desarrollo de aplicaciones. El mejor modo para tener una idea del ámbito de los paquetes y bibliotecas JDK es verificar la documentación API JDK

El Java Runtime Environment

- El Java Runtime Environment (JRE, también conocido como el Java Runtime) incluye las bibliotecas de códigos de la JVM y los componentes que son necesarios para programas en ejecución escritos en el lenguaje Java. Está disponible para múltiples plataformas. Puede redistribuir libremente el JRE con sus aplicaciones, de acuerdo a los términos de la licencia del JRE, para darles a los usuarios de la aplicación una plataforma en la cual ejecutar su software. El JRE se incluye en el JDK.

		NOMBRE	TIPO	OCUPA	RANGO APROXIMADO
TIPOS DE DATOS EN JAVA	**TIPOS PRIMITIVOS** (sin métodos; no son objetos; no necesitan una invocación para ser creados)	byte	Entero	1 byte	-128 a 127
		short	Entero	2 bytes	-32768 a 32767
		int	Entero	4 bytes	$2*10^9$
		long	Entero	8 bytes	Muy grande
		float	Decimal simple	4 bytes	Muy grande
		double	Decimal doble	8 bytes	Muy grande
		char	Carácter simple	2 bytes	---
		boolean	Valor true o false	1 byte	---
	TIPOS OBJETO (con métodos, necesitan una invocación para ser creados)	Tipos de la biblioteca estándar de Java	String (cadenas de texto) Muchos otros (p.ej. Scanner, TreeSet, ArrayList...)		
		Tipos definidos por el programador / usuario	Cualquiera que se nos ocurra, por ejemplo Taxi, Autobus, Tranvia		
		arrays	Serie de elementos o formación tipo vector o matriz. Lo consideraremos un objeto especial que carece de métodos.		
		Tipos envoltorio o wrapper (Equivalentes a los tipos primitivos pero como objetos.)	Byte		
			Short		
			Integer		
			Long		
			Float		
			Double		
			Character		
			Boolean		

Expresiones y Operadores

- Expresión
  - Una expresión es una combinación de variables, operadores y llamadas de métodos construida de acuerdo a la sintaxis del lenguaje que devuelve un valor.
  - El tipo de dato del valor regresado por una expresión depende de los elementos usados en la expresión.
- Operadores
  - Los operadores son símbolos especiales que por lo común se utilizan en expresiones.

Operador	Significado	Ejemplo
	Operadores aritméticos	
+	Suma	a + b
-	Resta	a - b
*	Multiplicación	a * b
/	División	a / b
%	Módulo	a % b
	Operadores de asignación	
=	Asignación	a = b
+=	Suma y asignación	a += b (a=a + b)
-=	Resta y asignación	a -= b (a=a - b)
*=	Multiplicación y asignación	a *= b (a=a * b)
/=	División y asignación	a / b (a=a / b)
%=	Módulo y asignación	a % b (a=a % b)

Operadores relacionales

==	Igualdad	a == b
!=	Distinto	a != b
<	Menor que	a < b
>	Mayor que	a > b
<=	Menor o igual que	a <= b
>=	Mayor o igual que	a >= b

Operadores especiales

++	Incremento	a++ (postincremento) ++a (preincremento)
--	Decremento	a-- (postdecremento) --a (predecremento)
(tipo)expr	Cast	a = (int) b
+	Concatenación de cadenas	a = "cad1" + "cad2"
.	Acceso a variables y métodos	a = obj.var1
( )	Agrupación de expresiones	a = (a + b) * c

Precedencia de operadores

Operador	Notas
. [] ()	Los corchetes se utilizan para los arreglos
++ -- ! ~	! es el NOT lógico y ~ es el complemento de bits
new (tipo)expr	new se utiliza para crear instancias de clases
* / %	Multiplicativos
+ -	Aditivos
<< >> >>>	Corrimiento de bits
< > <= >=	Relacionales
== !=	Igualdad
&	AND (entre bits)
^	OR exclusivo (entre bits)
|	OR inclusivo (entre bits)
&&	AND lógico
||	OR lógico
? :	Condicional
= += -= *= /= %= &= ^= |= <<= >>= >>>=	Asignación

Variables

- Las variables son localidades de memoria en las que pueden almacenarse datos. Cada una tiene un nombre, un tipo y valor. Java tiene tres tipos de variables: de instancia, de clase y locales.

- Variables de instancia.
  - Se utilizan para definir los atributos de un objeto.

- Variables de clase.
  - Son similares a las variables de instancia, con la excepción de que sus valores son los mismos para todas las instancias de la clase.

- Variable locales.
  - Se declaran y se utilizan dentro de las definiciones de los métodos.

* A diferencia de otros lenguajes, Java no tiene variables globales, es decir , variables que son vistas en cualquier parte del programa.

Estructuras de Control

- Las sentencias de control de flujo se pueden utilizar para ejecutar sentencias condicionalmente, para ejecutar de manera repetida un bloque de sentencias y en general para cambiar la secuencia normal de un programa.

Estructura de Control If-else

- **La sentencia if**
- La sentencia **if** permite llevar a cabo la ejecución condicional de sentencias.

```
if ( Expresion )
{
   sentencias;
}
```

Se ejecutan las sentencias si al evaluar la expresión se obtiene un valor booleano true.