1

PROGRAMACIÓN I (JAVA)

1 Programación I (Java)

CONTENIDO

UNIDAD 1.

1. **INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN**

1.1 Características y estructura del lenguaje.
1.2 Operadores
1.3 Declaración de constantes
1.4 Declaración de una variable
1.5 Sentencias de asignación
1.6 Sentencias de Entrada/Salida
1.7 Ejercicios de aplicación

1. INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN

1.1 CARACTERÍSTICAS Y ESTRUCTURA DEL LENGUAJE

HISTORIA Y EVOLUCIÓN DE JAVA

Lenguaje de programación de uso general, aunque ha sido fuertemente ligado a internet.

Su innovación ha ocurrido por 2 razones:

Para adaptarse a los cambios en ambientes y usos

Para implementar refinamientos y mejoras en el arte de la programación

LINAJE DE JAVA

Hereda características de:

Creado en 1972 por Dennis M. Ritchie. Basado en lenguaje B

Creado en 1972 por Dennis M. Ritchie. Basado en lenguaje B

Lenguaje Original

Es una ampliación de C

```
//Fecha:5/4/2010
//suma
//Virginia y Esther

#include<studio.h>
int main (){
        int numero1;
        int numero2;
        int suma;
        suma=0;

        printf ("Introduce el valor del primer sumando:");
        scanf ("%d",&numero1);
        printf ("Introduce el valor del segundo sumando :");
        scanf ("%d",&nuemro2);

        suma=numero1+numero2;

        printf ("La solucion es :%d",suma);
}
```

```
#include <iostream>
using namespace std;
int main(){
  int cont = 1,num;
  cin>>num;

  do{
    cout<<num*cont<<endl;
    cont++;
  }while(cont<=10);

  return 0;
}
```

```
C:\User
10
10
20
30
40
50
60
70
80
90
100

Process
Press ar
```

La creación de C fue el resultado directo de la necesidad de un lenguaje de alto nivel, estructurado, eficiente y que pudiera reemplazar el código ensamblador en la creación de programas.

Lenguaje estructurado: lenguaje natural (palabras y construcciones)

Paradigma de Programación: provee la visión y métodos de un programador en la construcción de un programa o subprograma.

Diversas formas de pensar la solución de un problema.

Lenguaje Ensamblador: lenguaje de programación de bajo nivel (lenguaje máquina). Se programa directamente sobre hardware.

Binario
Hexadecimal

MOV al, 061h
Asigna el valor hexadecimal 61 (97 en decimal) al registro "al"

Antes de C se utilizaron

FORTRAN
BASIC Dependían del GOTO (ir a)
COBOL

Como consecuencia producían "Código Spaghetti" Códigos llenos de saltos, enredados y ramificaciones condicionales.

"Hacen que la comprensión sea imposible"

La complejidad en el tamaño de los programas por tal razón surge…

"La programación orientada a objetos (POO)"

Es una metodología de programación que ayuda a organizar programas complejos mediante el uso de herencia, encapsulación y polimorfismo, principios fundamentales de Java.

CREACIÓN DE JAVA
Creado por:

1. James Gosling
2. Patrick Naughton
3. Chris Warth Para Sun Microsystems 1991.
4. Ed. Frank
5. Mike Sheridan

1

Fue llamado inicialmente **"Oak"** y en 1995 fue nombrado "Java"

Otros contribuyentes:
1. Bill Joy
2. Artur van Huff
3. Jhonathan Payne
4. Frank Yellin
5. Tim Lindholm

"El impulso inicial de Java no fue internet"
Se creó para ser un lenguaje independiente de plataforma (arquitectura neutral). Su utilización en internet se dio por su portabilidad.

Portable: independiente de plataforma.

Dadas las semejanzas entre Java y C++ se puede pensar que Java es simplemente "La versión de C++ para internet", pero NO es así.

Java tiene prácticas y filosofías diferentes.

Multihilo: Biblioteca que simplifica el acceso a internet.

C# creado por Microsoft.

.NET y C# están fuertemente relacionados con Java, ya que ambos comparten la misma sintaxis general y el mismo modelo de objetos.

APPLET: es un tipo especial de programa de Java que es diseñado para ser transmitido por internet y automáticamente ejecutado por un navegador compatible con Java.

Los Applets son dinámicos, ellos mismos se ejecutan.

***Categoría de Objetos.**

Información Pasiva (sin ejecutar)

Programas Activos (cuando se ejecuta)

La creación de los **applets** cambió la programación para internet, porque expandió el universo de objetos que pueden ser movidos libremente en el ciberespacio.

BYTECODE: con esto resuelve Java el problema de seguridad y portabilidad (para no ser confundido como virus)

Es un conjunto de instrucciones altamente optimizado diseñado para ser ejecutado por una máquina virtual "JAVA VIRTUAL MACHINE (JVM)"

JVM: es un intérprete de Bytecode.

Tecnología **HotSpot:** proporciona un compilador de bytecode a código nativo denominado Just In Time (JIT)

SERVLETS: Programa que se ejecuta en el servidor.

CGI (Common Gateway Interface)

1

CUALIDADES DE JAVA

- **SIMPLE**
  - Aprendizaje y utilización sencilla.
- **SEGURO**
  - Por JMV controla el acceso a recursos completos del PC
- **PORTABLE**
  - Sus programas se ejecutan en diversos sistemas
- **DISTRIBUIDO**
  - Ideado para el entorno distribuido de internet (Protocolo TCP/IP).
  - RIM (Invocación remota de métodos) - en internet.
- **ORIENTADO A OBJETOS**
  - Todo es un objeto
- **DINÁMICO**
  - Actualización de bytecode en tiempo real
- **ROBUSTO**
  - Liberación de memoria automática
  - Gestión de memoria. Condicionales de excepción no controlada (errores)
  - Es fiable (sin errores).
  - Es tipificado (comprueba el código mediante compilación y ejecución.
- **ALTO RENDIMIENTO**
  - Los intérpretes de Java proporcionan un alto rendimiento
- **INTERPRETADO**
  - Compilador de bytecode con características JIT (Justin In Time). Lo que da la ejecución en diversas plataformas
- **ARTQUITECTURA NEUTRAL**
  - Que no importe la longevidad y portabilidad. La meta es : **«escribir una vez, ejecutar en cualquier sitio, momento y para siempre»**
- **MULTIHILOS**
  - Permite la escritura de programas que hagan varias cosas simultáneamente.

1

PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

En la programación orientada a objetos existen 2 enfoques de programación:

1. **Modelo Orientado a Procesos.** Lo que está ocurriendo, programa como una serie de pasos lineales, es decir, un código.
2. **Programación Orientada a Objetos.** Se basa en el paradigma de "quién está siendo afectado". Organiza un programa alrededor de sus datos, es decir, objetos.

Un programa orientado a objetos se puede definir como un conjunto de datos que controlan el acceso al código.

ABSTRACCIÓN. La abstracción es un elemento esencial de la POO. Se define como: separar las propiedades de un objeto a través de una operación mental.

Ejemplo:

Vemos un coche o un auto como un objeto, NO como un conjunto de diez mil partes individuales.

En POO cada uno de los objetos describe su comportamiento propio y único. Se puede tratar estos objetos como entidades que responden a los mensajes que les ordenan **"hacer algo".**

TRES PRINCIPIOS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS

1. ENCAPSULAMIENTO. Es el mecanismo que permite unir el código junto con los datos que manipula y mantiene a ambos a salvo de interferencias exteriores y de uso indebido.

Una forma de ver el encapsulado es como una envoltura protectora.

En Java la base de la encapsulación es **La Clase**

Una clase define la estructura y comportamiento **(datos y código)** que serán compartidos por un conjunto de objetos.

Cada **objeto de una clase** contiene la estructura y comportamiento definida por la clase, por esta razón los objetos se denominan **Instancia de una Clase.**

Una clase es una construcción lógica, mientras que un objeto tiene una realidad física.

Cuando se crea una clase, se especifica el código y los datos que constituyen esa clase, lo que se denomina **miembros de la clase** (código y datos)

Código: operan sobre los datos. Son métodos miembros o simplemente métodos.

Datos: son variables miembros o variables de instancia.

MÉTODO = FUNCIÓN C/C++

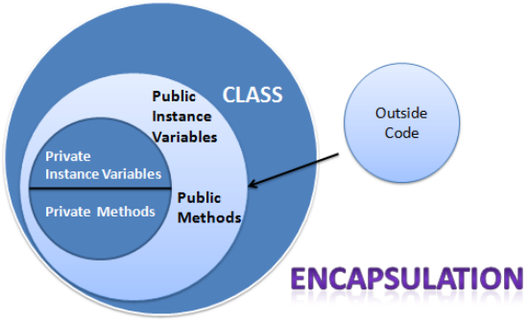
En los programas en Java **los métodos definen cómo se pueden utilizar las variables miembros (datos).** Esto significa que el comportamiento y la interfaz de una clase están definidos por los métodos que operan sobre sus datos de instancia.

Cada método o variable dentro de una clase puede declararse **privada o pública.**

Privada: Sólo se puede acceder por el código miembro de la clase.

Pública: Representa todo lo que el usuario externo necesita o puede conocer.

* Cualquier código que no sea miembro de una clase no puede tener acceso a un método o variable privada.



Fuente: http://skuarch.blogspot.com.co/2013/11/encapsulacion-en-java.html

2. **HERENCIA.** Es el proceso por el cual un objeto adquiere las propiedades de otro objeto. Es la base del concepto de clasificación jerárquica, cada objeto posee las características de su clase.

Ejemplo:

Un labrador es parte de la clasificación de perros, que a su vez es parte de la clasificación de mamíferos que está contenida en una clasificación mayor, que es la clase animal.

Por la clasificación de jerarquías un objeto sólo necesita definir aquellas cualidades que lo hacen único en su clase.

Clase: Descripción de atributos (características) y comportamientos (acciones)

Una **Subclase** hereda todos los atributos de cada uno de sus predecesores en la jerarquía de clases.

3. **POLIMORFISMO** (Muchas formas). Es una característica que permite que una interfaz sea utilizada por una clase general de acciones.

Método: es una clase general de acciones.

Ejemplo:

El sentido del olfato de un perro es **polimórfico,** si el pero huele un gato, ladrará y correrá detrás de él. Si el perro huele comida, producirá saliva y correrá hacia su plato.

El mismo sentido está funcionando en ambas situaciones, la diferencia está en lo que el perro huele, es decir, esto sería el **dato**, sobre la acción que sería el método.

TODO PROGRAMA EN JAVA ESTÁ ORIENTADO A OBJETO, ES DECIR, CADA PROGRAMA EN JAVA IMPLICA ENCAPSULACIÓN, HERENCIA Y POLIMORFISMO.

1.2 OPERADORES

Java proporciona un amplio conjunto de operadores, que se pueden dividir en cuatro grupos:

- ✓ Aritméticos
- ✓ A nivel de bit
- ✓ Relacionales
- ✓ Lógicas

- **Operadores Aritméticos.** Se utilizan en expresiones matemáticas de la misma forma que son utilizados en álgebra.

Operador	Resultado
+	Suma
-	Resta (también es el menos unario)
*	Producto
/	Cociente
%	Módulo
++	Incremento
+=	Suma y asignación
-=	Resta y asignación
*=	Multiplicación y asignación
/=	División y asignación
%=	Módulo y asignación
--	Decremento

Operadores Aritméticos básicos

Las operaciones aritméticas básicas, suma, resta, multiplicación y división, se comportan con todos los tipos numéricos como se espera. El operador menos también tiene una forma unaria que sirve para negar su operador único (valores negativos)

Las operaciones con operadores siguen un orden de prelación o de precedencia que determinan el orden con el que se ejecutan. Si existen expresiones con varios operadores del mismo nivel, la operación se ejecuta de izquierda a derecha. Para evitar resultados no deseados, en casos donde pueda existir duda se recomienda el uso de paréntesis para dejar claro con qué orden deben ejecutarse las operaciones. Por ejemplo, si dudas si la expresión 3 * a / 7 + 2 se ejecutará en el orden que tú deseas, especifica el orden deseado utilizando paréntesis.
Por ejemplo 3 * ((a / 7) + 2).

El operador de Módulo (%). Devuelve el residuo generado por una operación de división. Se puede aplicar tanto a los tipos de punto flotante como a los tipos enteros.

Destacar que el operador % es de uso exclusivo entre enteros. 7%3 devuelve 1 ya que el resto de dividir 7 entre 3 es 1. Al valor obtenido lo denominamos módulo (en otros lenguajes en vez del símbolo % se usa la palabra clave mod) y a este operador a veces se le denomina “operador módulo”.

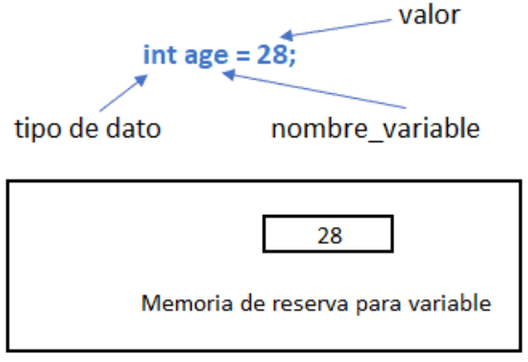
1.3 DECLARACIÓN DE CONSTANTES

Una constante es una variable del sistema que mantiene un valor inmutable a lo largo de toda la vida del programa. Las constantes en Java se definen mediante el modificador final.

```
int num;
num = 10;
```

1.4 DECLARACIÓN DE VARIABLES

Podemos declarar las variables en Java de la siguiente manera[6]:



Fuente: [6]

tipo de dato: tipo de datos que se pueden almacenar en esta variable.
nombre_variable: nombre dado a la variable.
valor: es el valor inicial almacenado en la variable.

Identificadores y Palabras Reservadas en Java

En los lenguajes de programación, los identificadores (como su nombre lo indica) se utilizan con fines de identificación. En Java, un identificador puede ser un nombre de clase, un nombre de método o un nombre de variable.

Existen ciertas reglas para definir un identificador java válido. Estas reglas deben seguirse, de lo contrario obtenemos un error en tiempo de compilación. Estas reglas también son válidas para otros lenguajes como C, C++.

- ✓ Los únicos caracteres permitidos para los identificadores son todos los caracteres alfanuméricos ([AZ], [az], [0-9]), "$" (signo de dólar) y '_' (guión bajo). Por ejemplo, "java@" no es un identificador de Java válido ya que contiene "@" – carácter especial.
- ✓ Los identificadores no deben comenzar con dígitos ([0-9]). Por ejemplo, "123java" no es un identificador de Java válido.
- ✓ Los identificadores de Java distinguen entre mayúsculas y minúsculas.
- ✓ No hay límite en la longitud del identificador, pero es aconsejable usar solamente una longitud óptima de 4 a 15 caracteres.
- ✓ Las palabras reservadas no se pueden usar como un identificador. Por ejemplo, int while = 28; es una declaración inválida ya que while es una palabra reservada. Hay 53 palabras reservadas en Java.
  Las puedes consultar dando clic aquí

1.5 SENTENCIAS DE ASIGNACIÓN

Cómo asignar valores dinámicamente a una variable en Java

Primera Forma:
Importar:

```
import java.util.Scanner;

Scanner Apuntador = new Scanner(System.in);
int num;
System.out.println("Por favor ingrese el número");
num = Apuntador.nextInt();
```

Segunda Forma:

```
int num;
System.out.println("Por favor ingrese el número");
num = new java.util.Scanner(System.in).nextInt();
```

Operadores aritméticos combinados con asignación

Java proporciona operadores especiales que se pueden utilizar para combinar una operación aritmética con una asignación.

Operación	Operación con asignación
a = a + 4;	a += 4;
a = a % 2;	a %= 2;
var = var op expresión; ***var op = expresión;*** ***Donde op es el operador aritmético (+, -, *...)***	

Incremento y Decremento

Los operadores de incremento y decremento de Java son respectivamente, ++ y --. El operador de incremento aumenta en una unidad a su operando, mientras que el operador de decremento reduce una unidad a su operador.
Por ejemplo:
x = x + 1; se puede escribir **x++;**
x = x - 1; se puede escribir **x--;**

En forma prefija, el operando es incrementado o decrementado antes de obtener el valor que se utilizará en la expresión.
Por ejemplo:
x = 42;
y = ++42;

En este caso, se asigna a **y** el valor de 43, ya que el incremento se produce antes de que se asigne a y el valor de x.

Sin embargo, si se escribe de esta otra forma:
x = 42;
y = x++;
El valor de **y** será 42. En ambos casos el valor de **x** es 43.

1.6 SENTENCIAS DE ENTRADA/SALIDA

Operadores relacionales. Los operadores relacionales determinan la relación que un operando tiene con otro.

Operador	Resultado
==	Igual a
!=	Diferente de
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

Operadores lógicos booleanos. Sólo operan sobre operandos del tipo **boolean**. Todos los operadores lógicos binarios combinan dos valores boolean para dar como resultado un valor **boolean.**

Operador	Resultado
&	AND lógico
|	OR lógico
^	XOR lógico (OR exclusivo) (F si ambas F o V)
||	OR en cortocircuito
&&	AND en cortocircuito
!	NOT lógico unario
&=	Asignación AND
|=	Asignación OR
^=	Asignación XOR
==	Igual a
!=	Diferente a
?:	If-then-else ternario

Cuando se utilizan las formas en cortocircuito Java no evalúa el operando de la derecha si el resultado de la operación queda determinado por el operando de la izquierda. Esto es útil cuando el operando de la derecha depende de que el de la izquierda sea true o false.

Java incluye un operador ternario especial que puede sustituir en ciertos casos los tipos de sentencias if-then-else. Su sintaxis es:

expresión 1 ? expresión 2 : expresión 3

1.7 EJERCICIOS DE APLICACIÓN

Aplicación # 1.1

Declare una clase principal en Java, donde:

Precio = preciobase + 0.35
Tenga en cuenta que en toda asignación debe haber coincidencia en el tipo de variables.
Saldo += cantidad
Preciocaja = precio* 2
Preciocaja += 10;

Saldo -= preciocaja*cantidadcaja

Nota: realice la prueba de escritorio, imprima el valor de cada variable utilizada y si requiere el valor de alguna variable, pida al usuario que la ingrese por teclado.

Aplicación # 1.2

Define una clase principal llamada BasicMat, donde declares las variables de tipo entero **a, b, c, d** y **e**. Teniendo en cuenta que el valor de cada variable es el siguiente:
Asignar el valor de la suma de dos unidades a la variable **a**.
El valor de 3 veces **a**, a la variable **b**.
Un cuarto de **b**, a la variable **c**.
El valor de la variable **d**, es **c – a**.
Y el valor de **e**, sería la negación de la variable **d**.

Primero realiza el algoritmo y la prueba de escritorio. Escribe el algoritmo en el MCV.

Luego utiliza el entorno de desarrollo de NetBeans, para realizar la aplicación.

Al ejecutar el programa, la salida debe ser la siguiente:
a = 2; b = 6; c = 1; d = - 1; e= 1
Realiza el mismo procedimiento utilizando el tipo de variable **double**, y observa las diferencias.

Aplicación # 1.3.

En la misma clase, realiza lo siguiente: determinar cuántos años faltan para que la edad de una persona sea múltiplo de 5 y mostrar un mensaje informativo por pantalla. Por ejemplo si la persona tiene 22 años; se deberá en primer lugar obtener el residuo de la división de 22 entre 5, que es 2. Luego, obtendrá los años que faltan para que la persona tenga una edad múltiplo de 5; que será 5-2 = 3 años. Para finalizar deberá mostrar un mensaje por consola del tipo: "La persona tendrá una edad múltiplo de 5 dentro de 3 años". Realice el algoritmo y luego hágalo en Java.

Ejemplo: Si el usuario ingresa que su edad es 31.
Al ejecutar el programa, la salida debe ser la siguiente:
"La persona tendrá una edad múltiplo de 5 dentro de 4 años".

Aplicación # 1.4.

Realice en Java, lo siguiente, utilizando el entorno de desarrollo de Netbeans:

1. Declare las variables enteras x, y, z, w
2. Asigne de forma constante los siguientes valores:
   x = - 2;
   y = - 1;
   z = 0;

3. Halle w, mediante la siguiente ecuación:

$$w = \{x - [(y - z)(x - y)]\} - \{z - (y + x)\}$$

4. Realice:
   x + = - 2 ;
   y - = 3;
   z =++y;
   y = x++;

5. Imprima los valores de x, y, z, w

Nota: Usted debe realizar primero la prueba de escritorio y luego pasar a Java en el entorno Netbeans.

REFERENCIAS BIBLIOGRÁFICAS

Schildt, Herbert. JAVA. Séptima Edición. Mc Graw Hill. 2007. México.
BIBLIOTECA VIRTUAL UNIVERSIDAD POPULAR DEL CESAR
APRENDER A PROGRAMAR. Recuperado de: <http://www.aprenderaprogramar.com>
Toro, José Luis. Estructuras de repetición. Recuperado de: <http://i7exe.blogspot.com.co/2013/04/estructuras-de-repeticion.html>
Garay, Emerson. Guía práctica en java con NetBeans. Recuperado de: <https://es.slideshare.net/emergar/guia-practica-funciones-en-java-con-netbeans?qid=0181f3df-0caf-4cfe-8e94-34982da03e18&v=&b=&from_search=1>