Lenguaje de programación Java

El lenguaje Java se originó con el propósito de conectar diferentes dispositivos electrónicos. Si bien el objetivo del proyecto original no se cumplió, comenzó a utilizarse a partir de 1995 y hoy en día es mundialmente empleado en páginas Web, multimedia y aplicaciones de escritorio.

Características del lenguaje

Es orientado a objetos: esto lo diferencia de los lenguajes procedurales, como el lenguaje C. Los lenguajes orientados a objetos se centran en la creación de objetos y en la interacción de estos entre si mediante mensajes, mientras que los lenguajes procedurales están centrados en la secuencia de pasos que es necesaria para resolver un problema. En este útlimo caso, la abstracción principal es el algoritmo, mientras que en la programación orientada a objetos es el tip de dato abstracto.

Es multiplataforma: cuando el programa fuente se compila, se genera un código denominado ***bytecode*** que es independiente de la plataforma. Cada plataforma posee un programa llamado ***máquina virtual*** que interpreta el ***bytecode*** y lo ejecuta en una máquina en particular.

Se llama plataforma al conjunto de CPU + Sistema Operativo.

La empresa Sun Microsystems desarrolló el entorno de ejecución de Java (JRE), compuesto por la máquina virtual de Java (JVM) y la biblioteca de clases de Java.

Tecnología Java

Hay tres opciones dentro de la tecnología desarrollada por Java, que responden a distintas necesidades de desarrollo.

- Java 2 Platform, Standard Edition (J2SE): permite desarrollar aplicaciones de escritorio y código que se ejecuta dentro de una página Web (applet).
- Java 2 Platform, Enterprice Edition (J2EE): permite desarrollar aplicaciones cliente-servidor, apropiadas para el ámbito empresarial.
- Java 2 Platform, Micro Edition (J2ME): permiten en desarrollo de aplicaciones que se ejecutan en dispositivos con recursos restringidos, como celulares, agendas electrónicas, etc.

El SDK es el kit de desarrollo que permite crear, compilar y ejecutar los programas Java.

El kit de desarrollo puede bajarse libremente de la página de Oracle.

Primer programa en Java

Para comenzar por el clásico “Hola mundo” hay que comprender que en Java, cada archivo fuente es una clase.

Utilizando un editor de textos, como el block de notas, escriban el siguiente código y guárdenlo como ***HolaMundo.java***.

```
public class HolaMundo{

    public static void main(String [] args){

        System.out.println("Hola mundo");

    }//fin de main()

}//fin de HolaMundo
```

Explicación del primer programa en Java

```
public class HolaMundo{
```

- declara una clase pública, o sea accesible a otras clases

- HolaMundo es el nombre de la clase y también el nombre del archivo fuente, el cual debe tener extensión ***java.***

- la llave indica el inicio de la clase y es costumbre colocarla a la derecha del nombre de la clase, aunque puede ir debajo.

```
public static void main(String [] args){
```

- se define el método main(), el cual es público, estático y de tipo void. Un método es similar a la función main del C++. Es el punto de entrada del programa en Java

- entre paréntesis está el argumento de la línea de comando, que es un array de cadenas.

```
System.out.println("Hola mundo");
```

- se muestra la frase “Hola mundo” por la salida estándar, o sea la pantalla.

- si no se desea una nueva línea después de la salida, usar ***print***, ya que la terminación ***ln*** indica ***new line***.

Más adelante se explicará detalladamente cada instrucción, pero para comenzar hay que seguir algunas normas elementales:

1. Cada bloque de código va encerrado entre llaves.
2. Identar (tabular) los bloques para facilitar la lectura del código.

3. El nombre del archivo fuente debe ser el mismo que el de la clase pública. Debe llevar la extensión ***java***.
4. Los identificadores de las clases comienzan con mayúscula.
5. Los comentarios de una sola línea comienzan con ***//.***
6. Los comentarios que abarquen más de una línea comienzan con ***/**** y finalizan con ****/*** (al estilo del lenguaje C).

Sintaxis del lenguaje

- Solamente los nombres de las clases comienzan con mayúscula como ***System*** o ***String***.
- Los demás identificadores comienzan con minúscula.
- Las variables o atributos no llevan paréntesis como ***out***.
- Los metodos llevan paréntesis como ***println().***
- Cuando un identificador está formado por varias palabras, cada inicial comienza con minúscula, como en ***HolaMundo*** (si es una clase) o la primera en minúscula si no es una clase, como en ***precioUnitario (CamelCase)***.

Uso del entorno Netbeans

Netbeans es un entorno de desarrollo de libre distribución que puede bajarse del sitio www.netbeans.org. Una vez instalado el software, sigan los siguientes pasos:

- creen una carpeta con un nombre a elección, por ejemplo ***mistrabajos***.
- ejecuten Netbeans y elijan el menú ***File*** – ***New Project***.
- seleccionen el tipo de proyecto ***Java application***, luego presionen el botón ***Next***.
- elijan como ***Project Location***, la carpeta que crearon y como ***Project Name*** un nombre a elección, por ejemplo ***secuencial***.
- desmarquen la casilla ***Create Main Cl***ass.
- presionen ***Finish***.

Se genera un proyecto con varias carpetas. Presionamos el botón derecho del ***Mouse*** sobre ***Source Packages*** y elegimos ***New Java Class***. Le damos un nombre a la clase, por ejemplo ***HolaMund***o y escribimos el código del ejemplo anterior.

Netbeans posee un conjunto de instrucciones abreviadas que se expanden al pulsar la tecla ***Tabulado***. Por ejemplo ***psvm*** se convierte en ***public static void main(String[] args)*** o ***sout*** en ***System.out.println("")***. Para ver otras opciones, ir al menú ***Tools, Options***, ***Editor***, solapa ***Code Templates***.

Para ejecutar el programa, con el botón derecho del ***Mouse*** sobre la clase recién creada, elegir ***Run File***.

Tipos de Datos Primitivos en Java

El lenguaje Java posee estos tipos primitivos:

Tipo	**Tamaño/Formato**	**Descripción**
byte	8 bits	Entero de un Byte
short	16 bits	Entero corto
int	32 bits	Entero
long	64 bits	Entero largo
float	32 bits	Coma flotante de precisión simple
double	64 bits	Coma flotante de precisión doble
char	16 bits	Un sólo caracter Unicode
boolean	true o false	Un valor booleano o lógico

Para utilizar cadenas de caracteres se emplea la clase ***String*** que no es un tipo de dato primitivo sino una clase.

Declaración e inicialización

Para declarar variables dentro del método main() u otros métodos, se escribe el tipo de dato, el nombre de la variable y se le da un valor inicial.

Para formar el nombre de una variable puede usarse el conjunto de caracteres Unicode, de 16 bits. Los primeros 256 son compatibles con el código ASCII, aunque no todas las plataformas soportan la totalidad de caracteres.

Una variable puede declararse dentro de cualquier bloque de código encerrado entre llaves, pero será visible solamente en ese bloque y en los bloques de nivel inferior. Más adelante se explicará el alcance de las variables.

Las variables se inicializan del mismo modo en que en el lenguaje C. Los números se escriben sin comillas y los caracteres entre comillas simples. A los contenidos de tipo flotante se les pospone la letra F mayúscula;

```
int num=3;

char letra='a';

float num=0.37F;
```

Los literales de tipo ***float*** utilizan el sufijo ***F***; los literales de tipo ***long*** utilizan el sufijo ***L***; otros literales no necesitan sufijo.

Para declarar una cadena se utiliza el tipo de dato ***String*** que no es primitivo, como es una clase se escribe con mayúsculas. Las cadenas se inicializan encerrando el texto entre comillas dobles.

```
String nombre="Pepe";
```

Para declarar una constante se antepone la palabra ***final*** al tipo de dato. Los nombres de constantes se escriben con mayúscula.

```
final float PI=3.14159F;
```

Operadores

Los operadores utilizados en Java son similares a los empleados en el lenguaje C. Se recordarán brevemente.

5.4 Operadores aritméticos

Operador	**Uso**	**Descripción**
+	op1 + op2	Suma op1 y op2
-	op1 - op2	Resta op2 de op1
-	-op1	Cambia el signo de op1
*	op1 * op2	Multiplica op1 y op2
/	op1 / op2	Divide op1 por op2
%	op1 % op2	Obtiene el resto de dividir op1 por op2

Operadores de incremento y decremento

Operador	Uso	Descripción
++	op ++	Incrementa op en 1; evalúa el valor antes de incrementar
++	++ op	Incrementa op en 1; evalúa el valor después de incrementar
--	op --	Decrementa op en 1; evalúa el valor antes de decrementar

--	-- op	Decrementa op en 1; evalúa el valor después de decrementar

Operadores relacionales

Operador	Uso	Devuelve true si
>	op1 > op2	op1 es mayor que op2
>=	op1 >= op2	op1 es mayor o igual que op2
<	op1 < op2	op1 es menor que op2
<=	op1 <= op2	op1 es menor o igual que op2
==	op1 == op2	op1 y op2 son iguales
!=	op1 != op2	op1 y op2 son distintos

Operadores lógicos

Operador	Uso	Devuelve true si
&&	op1 && op2	op1 y op2 son verdaderos
||	op1 || op2	uno de los dos es verdadero
!	! op	op es falso

Operadores de asignación

Operador	Uso	Equivale a
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2

/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2

6 Salida por consola

Para mostrar una salida por pantalla, se utilizan los métodos de la clase ***System***, propia del lenguaje Java.

`System.out.println()`

Muestra una cadena por pantalla y luego agrega un carácter de nueva línea.

`System.out.print()`

Muestra una cadena por pantalla, sin agregar una nueva línea.

Admiten como argumento distintos tipos de datos, como ***int, char, String, float, double, boolean.*** Los datos pueden concatenarse utilizando el signo ***+.***

Entrada por consola

A partir de la versión 5 de Java, el ingreso se facilita utilizando la clase Scanner. Se debe crear un objeto o instancia de esta clase, y utilizar sus métodos para leer distintos tipos de datos.

Ejemplo

```
import java.util.Scanner;

public class EntradaSalida {

    public static void main(String[] args) {

        String nombre="";
```

```
        int edad=0;

        Scanner entrada=new Scanner(System.in);

        System.out.println("Nombre: ");

        nombre=entrada.next();

        System.out.println("Edad: ");

        edad=entrada.nextInt();

        System.out.println("Ud es " + nombre + " y tiene " + edad + " años");

    }

}
```

Explicación

```
import java.util.Scanner;
```

Es similar al ***#include*** de ***C***. Permite utilizar la clase ***Scanner***.

```
Scanner entrada=new Scanner(System.in);
```

Se crea una instancia u objeto de la clase ***Scanner***. El argumento entre paréntesis es la entrada estándar, o sea el teclado.

```
nombre=entrada.next();
```

El método ***next()*** lee un ***String*** desde el teclado.

```
edad=entrada.nextInt();
```

El método ***nextInt()*** lee un entero desde el teclado.

Ejemplo

```
import java.util.*;

public class Longitud {
```

```
    public static void main(String[] args) {

        final float PI=3.14F;

        float radio=0;

        Scanner entrada=new Scanner(System.in);

        System.out.println(entrada);

        System.out.println("Ingrese el radio: ");

        radio=entrada.nextFloat();

        System.out.printf("Longitud: %.2f\n" ,2*PI* radio);

    }

}
```

Explicación

```
import java.util.*;
```

Permite utilizar todas las clases del paquete ***java.util***.

```
final float PI=3.14F;
```

Declara la constante PI, de tipo flotante.

```
radio=entrada.nextFloat();
```

Permite ingresar un número flotante desde el teclado.

Instrucciones de control

Estructura alternativa

Alternativa simple, evalúa solamente la parte verdadera.

```
if(condicion) {
     instrucción 1;
}
```

Alternativa simple, evalúa el caso verdadero y el falso.

```
if(condicion){
     instrucción 1;
}
else {
     instrucción 2;
}
```

Sentencia if-else anidadas.

```
if(condicion 1){
     instrucción 1;
}
else if(condicion 2) {
     instrucción 2;
}
else {
     instrucción 3;
}
```

Alternativa múltiple.

```
switch(variable){
     case caso1:
          sentencia 1;
          break;
case caso2:
          sentencia 2;
          break:
……
default:
          sentencia n;
}
```

Ejemplo

```
import java.util.Scanner;
public class AlternativaMultiple {
    public static void main(String[] args) {
        char ec=' ';
        String estadoCivil="";
        Scanner entrada=new Scanner(System.in);
        System.out.println("Estado civil (s/c/v/d): ");
        ec=entrada.next().toLowerCase().charAt(0);
        switch(ec){
```

```
            case 's':
            estadoCivil="soltero";
            break;
            case 'c':
            estadoCivil="casado";
            break;
            case 'v':
            estadoCivil="viudo";
            break;
            case 'd':
            estadoCivil="divorciado";
            break;
            default:
            estadoCivil="desconocido";
        }
        System.out.println("Su estado civil es "+estadoCivil);
    }
}
```

Explicación

`ec=entrada.next().toLowerCase().charAt(0);`

Lee un ***String*** del teclado, lo convierte a mayúscula y asigna el primer carácter a la variable ***ec***.

Estructura iterativa

```
while(condicion){
     sentencias;
}
```

Se ejecutan las sentencias mientras la condición sea verdadera. Si la condición es falsa, las sentencias no se ejecutan nunca.

```
do{
     sentencias:
} while(condicion);
```

Se ejecutan las sentencias mientras la condición sea verdadera. Si la condición es falsa, las sentencias se ejecutan una sola vez.

```
for(inicio;condicion;incremento){
     sentencias;
}
```

Las sentencias se ejecutan mientras la condición sea verdadera. Las variables involucradas en la condición pueden inicializarse e incrementarse dentro de los paréntesis.

Ejemplo

```
import java.util.Scanner;
```

```
public class Repeticion {

    public static void main(String[] args) {

        float num=0,sum=0;

        final int MAX=5;

        Scanner entrada=new Scanner(System.in);

        for(int i=0;i<MAX;i++){

            System.out.println("Ingrese un N°: ");

            num=entrada.nextFloat();

            sum+=num;

        }

        System.out.println("Promedio: "+sum/MAX);

    }

}
```

La variable **i** se puede declarar dentro de la repetición, pero solamente se conoce dentro de las llaves del **for**.

Actividades

Indicadores

- Creación correcta de un proyecto 1 p.
- Declaración de la clase que define el método main 1 p.
- Declaración de las variables 1 p.
- Implementación del programa 3 p.

1. Un tirador efectúa 5 disparos sobre un blanco. Según la distancia al centro, recibe el puntaje que se indica:

   DISTANCIA PUNTAJE

   0 <=D <=1 10

   1 < D <=5 5

   5 < D <=10 1

   D > 10 0

   Determinar el puntaje final del jugador.

2. Ingresar el estado civil de varias personas y contar cuántas son de cada clase. El ingreso finaliza con una respuesta negativa del usuario.
3. Ingresar varios números decimales, finalizar con un valor cero. Informar el porcentaje de números negativos.
4. Ingresar N que es una cantidad de números enteros. Luego ingresar esa cantidad de números. Encontrar el menor y contar las veces que se repite.
5. Ingresar un número entero positivo y validarlo. Informar todos sus divisores.