


TIEMPO PARA DESARROLLAR LA PRÁCTICA:

Prepararse para la Práctica y leer los objetivos	10 min.
Desarrollo de Ejemplos	30 min.
Desarrollar Ejercicios propuestos	50 min.
Guardar y apagar equipo.	10 min.

I. OBJETIVOS

Al finalizar la práctica, el estudiante será capaz de:

- Utilizar las herramientas para crear aplicaciones de Java
- Definir la estructura de los programas fuente de Java
- Identificar el proceso de compilación de Aplicaciones Java.

II. INTRODUCCIÓN TEORICA.
Sintaxis de Java Básico:
Variables

Una variable es un contenedor de datos identificado mediante un nombre (identificador). Dicho identificador se utilizará para referenciar el dato que contiene.

Toda variable debe llevar asociado un tipo que describe el tipo de dato que guarda. Por tanto, una variable tiene:

Un tipo.

Un identificador.

Un dato (o valor).

Declaración de variables

Es la sentencia mediante la cual se define una variable, asignándole un tipo y un identificador:

tipo identificador;

int contador;

Adicionalmente se le puede asignar un valor inicial mediante una asignación:

tipo identificador = valor;

int contador = 10;

Si no se le asigna un valor, se inicializará con el valor por defecto para ese tipo.

Variables primitivas vs. complejas

Una variable de tipo primitivo contiene el dato directamente:

byte a = 10;

Una variable de tipo complejo contiene una referencia (puntero) a la zona de memoria donde está el objeto:

String s = new String("Hola");

Ejemplos de sintaxis de JAVA:

Para el desarrollo de algunos de los siguientes ejemplos se deberá de crear un nuevo proyecto en el entorno de desarrollo de NetBeans de categoría General, Tipo de Proyecto Java Application. Con un nombre que contenga sus iniciales y ejemploP3 y Luego comenzara agregando un nuevo archivo de

categoría Java Classes y de tipo Java Class, por cada uno de los ejemplos siguientes con el nombre de la clase a crear.

Ejemplo 1

```
public class Variables1 {
    static boolean unBoolean;
    static byte unByte;
    static short unShort;
    static int unInt;
    static long unLong;
    static float unFloat;
    static double unDouble;
    static char unChar;
    static String unString;

    public static void main(String[] args) {
        System.out.println("El boolean vale: " + unBoolean);
        System.out.println("El byte vale: " + unByte);
        System.out.println("El short vale: " + unShort);
        System.out.println("El int vale: " + unInt);
        System.out.println("El long vale: " + unLong);
        System.out.println("El float vale: " + unFloat);
        System.out.println("El double vale: " + unDouble);
        System.out.println("El char vale: " + unChar);
        System.out.println("El String vale: " + unString);
    }
}
```

Ejemplo 2

```
public class Variables2 {
    public static void main(String[] args) {
        boolean unBoolean = true;
        byte unByte = 10;
        short unShort = 10;
        int unInt = 10;
        long unLong = 10;
        float unFloat = 3.14F;
        double unDouble = 3.14;
        char unChar = 'A';
        String unString = new String("Hola");

        System.out.println("El boolean vale: " + unBoolean);
        System.out.println("El byte vale: " + unByte);
        System.out.println("El short vale: " + unShort);
        System.out.println("El int vale: " + unInt);
        System.out.println("El long vale: " + unLong);
        System.out.println("El float vale: " + unFloat);
        System.out.println("El double vale: " + unDouble);
        System.out.println("El char vale: " + unChar);
        System.out.println("El String vale: " + unString);
    }
}
```

Una variable de instancia

es una variable definida para las instancias de una clase (cada objeto tiene su propia copia de la variable de instancia).

El ámbito de una variable de instancia abarca todos los métodos no estáticos de una clase:

- o Cuando es privada, todos los métodos pueden acceder al valor almacenado en la variable de instancia.
- o Cuando es pública, se puede acceder a ella desde cualquier lugar en el que se disponga de una referencia a un objeto de la clase.

Una variable estática

es una variable definida para la clase (compartida entre todas las instancias de una clase).

El ámbito de una variable estática:

- o Si es privada, cubre todos los métodos estáticos de la clase en que está definida.
- o Si es pública, abarca todos los métodos estáticos de todas las clases que formen parte de la aplicación.

Una variable local es una variable definida dentro del cuerpo de un método.

El ámbito de una variable local comienza en su declaración y termina donde termina el bloque de código ({}) que contiene la declaración.

- Un bloque es un conjunto de sentencias agrupadas entre llaves ({}):

El ámbito de una variable es la parte del programa en la que podemos hacer referencia a la variable

Uso de variables locales

En Java, las declaraciones se pueden poner en cualquier parte del código de un método, no necesariamente al principio:

```
void method ()
{
    int i=0;          // Declara e inicializa i

    while (i<10) {    // i está definido aquí
        int j=0;      // Declara j
        ...           // i y j definidos
    }                 // j ya no está definido

    System.out.print(i); // i todavía está definido
}                     // i deja de estar definido
```

Las variables locales han de declararse antes de utilizarse. Se pueden declarar variables locales con el mismo nombre en diferentes bloques de código. Incluso se podrían declarar en bloques de código no anidados dentro de un mismo método, aunque no es recomendable hacerlo.

- Las variables de instancia se inicializan automáticamente al crear un objeto (a 0 o null), mientras que las variables locales de un método tenemos que inicializarlas nosotros antes de usarlas.

Ámbito de las variables

El ámbito de una variable es la zona de código donde se puede referenciar dicha variable a través de su identificador.

El lugar de definición de una variable establece su ámbito.

Ámbitos: Atributos (o variables miembro).

Parámetros de método.

Variables locales: siempre hay que inicializarlas.

Variables de bloque: siempre hay que inicializarlas.

```
PublicclassAmbito1 {
// Atributos
// Declaración de atributos.
public static voidmain(String[] args)
{
//Parámetros
// Declaración de variable local.
//Locales
if(true)
{
// Declaración de variable de bloque.
//De bloque
}
}
}
```

Ejemplo 3

```
public classAmbito2 {
public static void main(String[] args){
if(true)
{
inti = 12;
}
System.out.println("El valor de i es: " + i);
}
}
```

Ejemplo 4

```
public classAmbito3 {
static inti = 5;
public static void main(String[] args) {
inti = 10;
System.out.println("El valor de i es: " + i);
}
}
```

Ejemplo de Operadores Unarios

```
public class OperadoresUnarios {
public static void main(String[] args) {
int x = 0;
int y = 0;
y = ++x;
System.out.println("y vale: " + y + ", x vale: " + x);
y = x++;
System.out.println("y vale: " + y + ", x vale: " + x);
}
}
```

Ejemplo de Bucles

```
public class Bucles {
public static void main(String[] args) {
int cont1 = 0;
```

```

while(cont1 < 3){
    System.out.println(cont1);
    cont1++;
}
int cont2 = 0;
do{
    System.out.println(cont2);
    cont2++;
}
while(cont2 < 3);
for(int cont3 = 0; cont3 < 3; cont3++)
{
    System.out.println(cont3);
}
}

```

Ejemplos 5

```

public class Temp {
    public static void main(String[] args){
        int x = 1;
        while(x<10)
        {
            x = x + 1;
            if(x>3)
            {
                System.out.println("Hola");
            }
        }
    }
}

```

Cómo es posible la multiplataforma en Java

Java es compatible con todos los sistemas porque basa su funcionamiento en los ByteCodes, que no es más que una precompilación del código fuente de Java.

Estos Byte Codes no son el programa en Java propiamente dicho, sino un archivo que contiene un código intermedio que puede manejar la Máquina Virtual de Java. Cada sistema operativo dispone de una Máquina Virtual de Java que puede interpretar los Byte Codes y transformarlos a sentencias ejecutables en el sistema en cuestión.

Ejemplos de Clases en JAVA.

Agregar una clase al proyecto con el nombre de HolaFecha.java

```

import java.util.*;
public class Holafecha {
    public static void main(String[] args) {
        System.out.println("Hola, tu Primera practica de JAVA : \n");
        System.out.println("La Fecha y Hora de Hoy es : " + new Date());
    }
}

```

Agregar una clase al proyecto con el nombre de calculos.java

```

class calculos
{

```

```

public static void main(String argv[])
{
    int n1 = 5;
    int n2 = 10, r1;
    float n3 = 20;
    float n4 = 33, r2;
    double n5 = 52.525456665;
    double n6 = 20.3652542555, r3;
    r1 = n1+n2;
    r2 = n3+n4;
    r3 = n5+n6;
    System.out.println("Suma de Numeros Enteros : "+r1);
    System.out.println("Suma de Numeros Flotantes : "+r2);
    System.out.println("Suma de Numeros Dobles : "+r3);
}
}

```

Agregar una clase al proyecto con el nombre de Alumnos.java

```

class Alumnos {
    public static void main (String[] arg) {
        String pedro= "Lopez, Pedro";
        String perico= "Lopez, Pedro";
        String manolo= "Pi, Manuel";
        System.out.println(pedro);
        System.out.println(perico);
        System.out.println(perico + " [perico]");
        System.out.println(manolo);
        System.out.println(manolo.substring(0, 2)); // Pi
        System.out.println(manolo.length()); // 10
        System.out.println(pedro.equals(perico)); // true
        System.out.println(pedro.equals(manolo)); // false
        System.out.println(pedro.compareTo(perico)); // 0
        System.out.println(pedro.compareTo(manolo)); // < 0
        System.out.println("una cadena normal");
        System.out.println("Jose Canas (alias \"pepon\")");
        System.out.println("Jos\u00e9 Calu00f1as (alias \"042pep\u00f3n042\")");
    }
}

```

Ejercicios Prácticos:

Para el desarrollo de los siguientes ejercicios deberá de crear un nuevo proyecto en el entorno de desarrollo de NetBeans de categoría General, Tipo de Proyecto Java Application. Luego comenzara agregando un nuevo archivo de categoría Java Class y de tipo Java Classes, porcada uno delos ejercicios siguientes con el nombre de la clase a crear.

1. Crear una clase de Java de nombre "Mensaje" que envíe como salida la siguiente frase: "BIENVENIDO A PROGRAMACIÓN IV".
2. Crear un clase que calcule la edad de una personas dada y que muestre como salida el siguiente mensaje "La edad calculada es: 00".
3. Crear una clase que calcule el área y el perímetro de una circunferencia.

4. Desarrollar un programa Java que calcule el factorial de un número entero.
5. Desarrollar un programa Java que sume los números del 1 al 100 (ambos inclusive).
6. Desarrollar un programa Java que muestre por pantalla los números primos del 1 al 1000 y todos los años bisiestos entre el año 2000 y el 3000.
7. Iniciar el análisis y diseño del sistema de planilla con diagrama de clases para esta:



Lineamientos para el desarrollo del Sistema de Planillas a desarrollar en las clases practicas de la materia de programación IV.

La cual tendrá las siguientes ponderaciones:

- ✓ 5% de la primera evaluación practica.
- ✓ 10% de la segunda evaluación practica.
- ✓ 15% de la tercera evaluación practica.

Se realizaran 3 entregas en todo el ciclo las cuales serán de acuerdo a las fechas de evaluación de la materia. Para llevar un control de los avances que se están teniendo en el aprendizaje de los alumnos y de la aplicación.

✓ Mantenimiento de Catálogos:

- Empleados
 - Datos:
 - Código
 - numero de identificación
 - primer nombre
 - segundo y tercer nombre
 - primer apellido
 - segundo apellido
 - apellido de casada
 - sexo
 - dirección
 - estado civil
 - teléfonos de contacto (3 como máximo)
 - No de Cuenta Bancaria.
 - Fecha de ingreso a la empresa
 - Fecha de nacimiento
 - No. de NIT
 - No. de AFP
 - No. ISSS
 - Nombre de AFP a la que pertenece
 - Salario Base
 - Departamento al que pertenece
 - Cargo
- AFP
 - datos
 - nombre de la AFP
 - Porcentaje de aplicación
 - mayor remuneración
- ISSS
 - datos
 - tope mensual
 - porcentaje de aplicación
- Departamentos
 - datos

- código
 - nombre del departamento
- Cargos
 - datos
 - Código.
 - Nombre del cargo
- usuarios del sistema
 - datos
 - código
 - nombres
 - apellidos
 - login
 - contraseña
 - fecha de creación
 - autorizado por
 - tipo de usuario(definir 3 niveles estáticos: bajo, medio, superior)
- Descuentos/Extras
 - datos
 - código
 - código del empleado
 - concepto del descuento/Extra
 - valor
- ✓ Cálculos del sistema
 - deducciones
 - AFP: multiplicar (salario base del empleado + extras – descuentos) por el porcentaje de aplicación al cual esta sujeto(en base a la AFP que pertenece)
 - Renta: multiplicar (salario base del empleado + extras – descuentos – lo descontado de AFP) por el 10% en concepto de renta.
 - ISSS: multiplicar (salario base del empleado + extras – descuentos) por el porcentaje de aplicación del seguro recordando que el monto tiene un tope mensual.
 - Total de Descuentos
 - sumar todas la deducciones mas el monto de descuentos del empleado
 - Sueldo Liquido
 - salario base + extras – total de descuentos
- ✓ Consultas y/o reportes
 - Empleados
 - Planilla de pagos
 - Descuentos /extras