Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Tecnicatura en Informática Aplicada al Diseño Multimedia y Sitios Web

Elementos de Programación

Unidad 3: El lenguaje de Programación Java

Docente: Ing. Risso, Oscar Luis
Año: 2024

Índice

Unidad 3: El lenguaje de Programación Java

Primera Sesión de Estudios

En este Tema

Estudiaremos aquí la estructura y sintaxis de los programas Java y conoceremos los elementos que utiliza: constantes, tipos, variables, expresiones, operadores, sentencias.

Java dispone de muchos tipos de datos para evitar errores en la lógica del programa y para optimizar el uso de los recursos de la computadora. Comenzaremos conociendo los tipos de datos simples.

Objetivos

Al finalizar este tema Ud. debe lograr:

- Conocer la estructura de los programas en lenguaje Java y los principales elementos que utiliza.
- Dominar la sintaxis del lenguaje a través de ejemplos simples.
- Conocer y emplear con criterio los tipos de datos simples de Java.
- Resolver algoritmos empleando la sintaxis de Java.

1. Algo de historia

Java fue diseñado en 1990 por James Gosling, de Sun Microsystems, como software para dispositivos electrónicos de consumo. Curiosamente, todo este lenguaje fue diseñado antes de que diese comienzo la era World Wide Web, puesto que fue diseñado para dispositivos electrónicos como calculadoras, microondas y la televisión interactiva.

En los primeros años de la década de los noventa, Sun Microsystems decidió intentar introducirse en el mercado de la electrónica de consumo y desarrollar programas para pequeños dispositivos electrónicos. Tras unos comienzos dudosos, Sun decidió crear una filial, denominada FirstPerson Inc., para dar margen de maniobra al equipo responsable del proyecto.

Inicialmente Java se llamó Oak (roble en inglés), aunque tuvo que cambiar de denominación, debido a que dicho nombre ya estaba registrado por otra empresa. Se dice este nombre se le puso debido a la existencia de tal árbol en los alrededores del lugar de trabajo de los promotores del lenguaje.

Tres de las principales razones que llevaron a crear Java son:

1. Creciente necesidad de interfaces mucho más cómodas e intuitivas que los sistemas de ventanas que proliferaban hasta el momento.
2. Fiabilidad del código y facilidad de desarrollo. Gosling observó que muchas de las características que ofrecían C o C++ aumentaban de forma alarmante el gran costo de pruebas y depuración. Por ello en sus ratos libres creó un lenguaje de programación donde intentaba solucionar los fallos que encontraba en C++.
3. Enorme diversidad de controladores electrónicos. Los dispositivos electrónicos se controlan mediante la utilización de microprocesadores de bajo precio y reducidas prestaciones, que varían cada poco tiempo y que utilizan diversos conjuntos de instrucciones. Java permite escribir un código común para todos los dispositivos.

Por todo ello, en lugar de tratar únicamente de optimizar las técnicas de desarrollo y dar por sentada la utilización de C o C++, el equipo de Gosling se planteó que tal vez los lenguajes existentes eran demasiado complicados como para conseguir reducir de forma apreciable la complejidad de desarrollo asociada a ese campo. Por este motivo, su primera propuesta fue idear un nuevo lenguaje de programación lo más sencillo posible, con el objeto de que se pudiese adaptar con facilidad a cualquier entorno de ejecución.

Habitualmente tendemos a asociar el termino "Java" al desarrollo de páginas web en Internet. Los neófitos creen que Java es un lenguaje para desarrollar páginas web y esto es totalmente erróneo. La confusión surge porque Java permite incrustar código dentro de páginas web para que sea ejecutado en el navegador del usuario. Estos son los famosos Applets que fueron promocionados en los años '90, pero hoy día son obsoletos y quedaron prácticamente en desuso.

Tampoco debemos confundir Java con JavaScript. El primero es un lenguaje de Programación y el segundo es un lenguaje de scripting que permite agregar funcionalidad dinámica en las páginas web. JavaScript no tiene nada que ver con Java. También es cierto que podemos utilizar Java para desarrollar páginas web, esta tecnología se basa en el desarrollo de aplicaciones tipo Servlets, esto es parte de lo que se conoce como JEE (Java Enterprise Edition) y excede el alcance del curso.-

2. Convenciones del lenguaje Java

2.1. Comentarios

Los comentarios son textos que conforman la documentación interna del programa y de las rutinas. Es muy útil emplear comentarios para describir lo que hace cada porción del código. Esto nos ayuda a depurar (corregir), entender y mantener nuestros programas con un gran ahorro de tiempo. Para poner comentarios en nuestro código fuente java tenemos tres opciones:

Una línea de comentario que empiece por //

```
1 // Esto es un comentario
2 System.out.println ("Hola mundo");
```

Varias líneas encerradas entre /* y */

```
1 /* Todo esto
2  también es un
3  comentario */
4 System.out.println ("Hola mundo");
```

Finalmente, podemos hacer varias líneas de comentario entre /** y */

```
1 /** Todo esto
2  también es un
3  comentario */
4 public void unMedodo (int unParametro) {
5  ...
6 }
```

La diferencia entre el comentario que empieza por /** y el que empieza por /* es que el primero sale en la documentación generada por javadoc y en la mayoría de herramientas que hacen documentación a partir del código. El comentario que empieza por /* no sale en la documentación.

Javadoc es una utilidad de Oracle para la generación de documentación de APIs en formato HTML a partir de código fuente Java. Javadoc es el estándar de la industria para documentar clases de Java. La mayoría de los IDEs los generan automáticamente.

Javadoc también proporciona una API1 para crear doclets2 y taglets3 , que le permite analizar la estructura de una aplicación Java. Así es como JDiff puede generar informes de lo que ha cambiado entre dos versiones de una API. Para ampliar este tema consulte `http://es.wikipedia.org/wiki/Javadoc`

Hay unos detalles, sin embargo, que conviene tener en cuenta para comentarios que van a salir en la documentación que se generará con Javadoc. Conviene poner uno de estos comentarios delante de cada clase, cada método y cada atributo, de esta forma, en la documentación aparecerán convenientemente documentadas las clases, métodos y atributos. Lo siguiente puede ser un ejemplo.

Ejemplo 3.1

Observe el código 1 implementado en Java. Se muestran varios comentarios para describir los pasos o acciones propuestas.

El algoritmo permite tomar 2 datos numéricos enteros de las cajas de diálogo que se muestran, efectuar su suma y mostrarla en una tercer caja de diálogo.

```
1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5  package u3_ejemplo1;
6  /**
7  *
8  * @author FICH
9  */
10 import javax.swing.JOptionPane;
11 public class U3_Ejemplo1 {
12  /** Este programa ingresa dos numeros e informa su suma
13  * @param args the command line arguments
14  */
15  public static void main(String[] args) {
16  // TODO code application logic here
17  int a,b;// Defino dos variables numericas tipo entero
18  a= Integer.parseInt(JOptionPane.showInputDialog("Ingrese un número"));//Ingreso el
        valor de a;
19  b= Integer.parseInt(JOptionPane.showInputDialog("Ingrese otro número")); //Ingreso
        el valor de b
20  JOptionPane.showMessageDialog(null,"La suma es: "+(a+b)); //Muestro la suma de a+b
21  }
22 }
```

Código 1: Código fuente en Java

Los comentarios son ignorados por el compilador, por lo tanto no afectan al algoritmo.

Sugerencia

Pierda algunos segundos escribiendo comentarios. Evitará así perder minutos y horas valiosas, tratando de entender partes del programa que necesita corregir, ampliar o reutilizar.

2.2. Mayúsculas y minúsculas en Java

Java hace distinción entre mayúsculas y minúsculas para los nombres o identificadores de los elementos del lenguaje. Es decir, que NO podemos escribir indistintamente cualquiera de estas sentencias :

```
1  a= Integer.parseInt(JOptionPane.showInputDialog("Ingrese un número"));//Correcto
2
3  a= integer.parseint(joptionpane.showinputdialog("ingrese un número"));// Incorrecto
```

El efecto NO será el mismo.

Importante

Es recomendable no modificar las líneas que aparecen automáticamente en el programa fuente del proyecto (programa principal) si no tiene experiencia en Java.

2.3. Elementos del lenguaje Java

Todo programa Java se construye a base a elementos (tokens) y separadores. Los separadores son simplemente los espacios en blanco y los comentarios. Los elementos (tokens) son las unidades con significado más pequeñas que se emplean en la construcción de sentencias simples.

2.4. Nombres o identificadores

Los identificadores son los nombres que toman los elementos de un programa, tal como una constante, una variable, una función, un tipo de dato, un componente. También el proyecto y las clases tienen nombre. Algunos identificadores corresponden a elementos predefinidos de Java y se denominan identificadores estándar. Pero la mayoría de las veces es el programador quien debe proponer el identificador de un elemento de programa; para hacerlo en Java, recuerde las siguientes reglas:

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

En Java, un identificador comienza con una letra, un subrayado (_) o un símbolo peso ($). Los siguientes caracteres pueden ser letras o dígitos. Se distinguen las mayúsculas de las minúsculas y no hay longitud máxima.

Son identificadores válidos	**No son identificadores válidos**
identificador	546identificador
nombre_Usuario	(Comienza con numeros)
nombreUsuario	Nombre Usuario
_variable_del_sistema	(Tiene un espacio intermedio)
$transaccion	Nombre.Usuario
	(Tiene un punto)
Su uso sería, por ejemplo:	_variable_añadida
int contador_principal;	(La letra ñ no esta permitida)
char _lista_de_ficheros;	$transacción
float $cantidad_en_Ptas;	(las vocales acentuadas no son válidas)

Sugerencia

La elección adecuada de los identificadores favorece la comprensión del programa. No se deberían usar identificadores muy largos ni demasiado cortos. Es recomendable que sugieran un significado pues favorece la lectura y comprensión de la lógica del programa. Por ejemplo: si se desea nombrar una variable que contenga el promedio de edades de un grupo de personas, es posible identificarla con p; pero mucho mejor es con PromedioEdades

2.5. Palabras reservadas

Las siguientes son las palabras clave que están definidas en Java y que no se pueden utilizar como identificadores:

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
cast	future	generic	inner	var
operator	outer	rest	(y mas...)	

2.6. Constantes

Un valor constante en Java se crea utilizando una representación literal de él. Java utiliza cinco tipos de elementos: enteros, reales en coma flotante, booleanos, caracteres y cadenas, que se pueden poner en cualquier lugar del código fuente de Java. Cada uno de estos literales tiene un tipo correspondiente asociado con él.

2.7. Constantes Literales

Son elementos que mantienen su valor durante toda la ejecución del programa. Ejemplo:

- 3.452
- -107
- 31.98E+3
- $04
- 'A'
- "Universidad"
- False
- True

2.8. Constantes con nombres

Se antepone la palabra final a la declaración

```
Sintaxis: final <tipo> identificador = <valor>

Ejemplo: final double Pi = 3.141592654
```

3. Variables

Son posiciones de memoria que permiten organizar la información que maneja un programa. Al declarar las variables en el programa Java en la declaración a través de un identificador y un tipo asociado a dicha variable, son asignadas en la memoria virtual del sistema operativo. Como su nombre lo indica, la información que contiene una variable puede cambiar o variar durante la ejecución o vida del programa.

3.1. Declaración de variables en Java

```
<tipo> Indentificador;
<tipo> Indentificador = <valor>;
<tipo> Lista de identificadores;
```

3.2. Convenciones

Los identificadores asociados a variables se suelen poner en minúsculas. Ejemplo:

```
Int COnTAdor //MAL
Int contador //BIEN
Int saldo, debe, haber; //BIEN
```

Cuando el identificador esta formado por varias palabras el identificador comienza en minúscula y las palabras sucesivas comienzan con su primer letra en mayúscula y el resto en minúsculas. Ejemplo:

```
Int cajachicadelbanco; //Mal
Int caja_chica_del_banco //Aceptable
Int cajaChicaDelBanco; //Bien
```

Las variables se pueden inicializar cuando se declaran. Ejemplo;

```
Int a, b; a= b= 0;
Int contador = 0;
Int saldo = 0, debe = 0, haber = 0;
```

Una vez declarada una variable, puede definirse; esto es, colocar un dato en la posición de memoria que ella representa.

La carga o definición de datos en las variables se puede realizar a través de dos acciones:

- Asignación
- Lectura

4. Tipos de Datos

Podemos distinguir 2 tipos de datos en Java

1. Datos Simples
2. Datos Compuestos (Vectores y Matrices), Cadenas (Strings)

4.1. Datos Simples

- **Enteros:** Los tipos de datos simples ordinales son aquellos sobre los que se puede establecer un orden y una secuencia.

byte	8 bits	rango: -128 a 127
short	16 bits	rango: -32,768 a 32,767
int	32 bits	rango: -2,147,483,648 a 2,147,483,647
long	64 bits	rango: -9,223,372,036,854,775,808L a 9,223,372,036,854,775,807L

Por ejemplo: 21 077 0xDC00

- **Reales en coma flotante:** Tipo simple No Ordinal

float	32 bits	rango: +/- 3.4E+38F (6-7 dígitos importantes)
double	64 bits	rango: +/- 1.8E+308 (15 dígitos importantes)

Por ejemplo: 3.14 2e12 3.1E12

- **Booleanos:** Tipo simple ordinal
  - true
  - false
- **Caracteres:** Tipo simple ordinal (Tabla de código ASCII)

Consultar: `http://es.wikipedia.org/wiki/ASCII`

char	16 bits	Conjunto de caracteres Unicode ISO

Por ejemplo: \a \t \u 'z'

4.2. Datos Compuestos

- **Vectores y Matrices (Arrays)**

Se pueden declarar en Java arrays de cualquier tipo:

```
char s[];
int iArray[];
```

Incluso se pueden construir arrays de arrays:

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

Los límites de los arrays se comprueban en tiempo de ejecución para evitar desbordamientos y la corrupción de memoria.

En Java un array es realmente un objeto, porque tiene redefinido el operador [ ]. Tiene una función miembro: length. Se puede utilizar este método para conocer la longitud de cualquier array.

```
int a[ ][ ] = new int[10][3];
a.length; /* 10 */
a[0].length; /* 3 */
```

Para crear un array en Java hay dos métodos básicos. Crear un array vacío:

```
int lista[ ] = new int[50]; (1)
```

o se puede crear el array con sus valores iniciales:

```
String nombres[ ] = { "Juan","Pepe","Pedro","Maria" };
```

Esto que es equivalente a:

```
String nombres[ ];
nombres = new String[4];
nombres[0] = new String( "Juan" ); //BIEN
nombres[1] = new String( "Pepe" ); //BIEN
```

```
nombres[2] = new String( "Pedro" ); //BIEN
nombres[3] = new String( "Maria" ); //BIEN
```

No se pueden crear arrays estáticos en tiempo de compilación:

```
int lista[50]; //MAL (generará un error en tiempo de compilación)
```

Tampoco se puede rellenar un array sin declarar el tamaño con el operador new:

```
 int lista[ ]; //MAL
 for( int i=0; i < 9; i++ )
 lista[ i ] = i;
```

Es decir, todos los arrays en Java se crean en tiempo de ejecución y son estáticos. Para convertir un array en el equivalente a un array dinámico en C/C++, se usa la clase vector, que permite operaciones de inserción, borrado, etc. en el array.

5. Operadores

Los operadores de Java son muy parecidos en estilo y funcionamiento a los de C. En la siguiente tabla aparecen los operadores que se utilizan en Java, por orden de precedencia.

5.1. Operadores aritméticos

Son operadores binarios, requieren siempre de dos operandos.

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)
- Resto de la división entera (%)

5.2. Operadores de asignación

Operador	Utilización	Expresión equivalente
+=	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
=	op1 = op2	op1 toma el valor de op2
= = = =	op1=op2=op3	asignación múltiple, todos toman el valor de op3

5.3. Operadores unarios

Los operadores (+) y (-) unarios sirven para mantener o cambiar el signo de una variable, constante o expresión numérica. Su uso en Java es el estándar de estos operadores.

5.4. Operador instanceof

El operador instanceof permite saber si un objeto pertenece o no a una clase determinada. Es un operador binario cuya forma general es:

NombreDeObjeto instanceof NombreDeClase

y devuelve true o false, según el objeto pertenezca o no a la clase.

5.5. Operador condicional ?

Este operador permite realizar bifurcaciones condicionales sencillas. Su forma general es:

ExpresionLogica ? resultado1 : resultado 2;

Donde es evaluada "ExpresionLogica", si devuelve verdadero (true) se ejecutará resultado1 y sino se ejecutara resultado2.

5.6. Operadores Incrementales

En java hay 2 operadores, (++) incremento y (- -) decremento. Estos incrementa o decrementan en una unidad a la variable que son aplicados. Se pueden utilizar de 2 formas:

1. Precediendo a la variable. (Ej. ++ i) en este caso se incrementa la variable y luego se utiliza, con el valor incrementado en 1.

2. Siguiendo a la variable. (Ej. i ++) en este caso se utiliza la variable sin incrementar y cuando finaliza la ejecución de la expresión a la que pertenece se incrementa la variable en 1.

5.7. Operadores relacionales

Operador	Utilización	Resultado es true (verdadero)
>	op1 > op2	Si op1 es mayor que op2
>=	op1 >= op2	Si op1 es mayor o igual que op2
<	op1 < op2	Si op1 es menor que op2
<=	op1 <= op2	Si op1 es menor o igual que op2
==	op1 == op2	Si op1 es igual que op2
! =	op1 ! = op2	Si op1 es diferente que op2

5.8. Operador (+) y (+=) para concatenar cadenas de caracteres

Para las cadenas, se pueden utilizar los operadores relacionales para comparaciones además de + y += para la concatenación. Ejemplo:

```
1 String apellidoNombre = apellido + ", " + nombre;
2 apellidoNombre += " - " + edad;
```

Se puede utilizar para concatenar en la salida por pantalla el valor de una variable con una expresión constante, Ejemplo:

```
1 "El precio de venta es $ " + precio + " de contado."
```

5.9. Operadores Lógicos

Una tabla de verdad, o tabla de valores de verdad, es una tabla que muestra el valor de verdad de una proposición compuesta (op1, op2).

Consultar `http://es.wikipedia.org/wiki/L%C3%B3gica_proposicional`

Operador	Utilización	Resultado
&& (AND)	op1 && op2	true si op1 y op2 son true, si op1 es flase no se evalúa op2
|| (OR)	op1 || op2	true si op1 u op2 son true, si op1 es true no se evalúa op2
! (Negación)	! op1	Devuelve el inverso del valor lógico actual
& (AND)	op1 & op2	true si op1 y op2 son true, siempre se evalúan op1 y op2
| (OR)	op1 | op2	true si op1 u op2 son true, siempre se evalúan op1 y op2

5.10. Operadores que actúan a nivel de bits

Operador	Utilización	Resultado
$\gg$	op1 $\gg$ op2	Desplaza los bits de op1 a la derecha una distancia op2
$\ll$	op1 $\ll$ op2	Desplaza los bits de op1 a la izquierda una distancia op2
$\gg>$	op1 $\gg>$ op2	Desplaza los bits de op1 a la derecha una distancia op2 (+)
&	op1 & op2	Operador AND a nivel de bits
|	op1 | op2	Operador OR a nivel de bits
$\wedge$	op1 $\wedge$ op2	Operador XOR a nivel de bits
$\sim$	op1 $\sim$ op2	Operador complemento, invierte el valor de cada bit

Los operadores numéricos se comportan como esperamos:

```
1 int + int = int
```

Los operadores relacionales devuelven un valor booleano. Para las cadenas, se pueden utilizar los operadores relacionales para comparaciones además de + y += para la concatenación:

```
1 String nombre = "nombre" + "Apellido";
```

El operador = siempre hace copias de objetos, marcando los antiguos para borrarlos, y ya se encargará el garbage collector de devolver al sistema la memoria ocupada por el objeto eliminado.

5.11. Separadores

Sólo hay un par de secuencias con otros caracteres que pueden aparecer en el código Java; son los separadores simples, que van a definir la forma y función del código. Los separadores admitidos en Java son:

- **( ) - paréntesis**. Para contener listas de parámetros en la definición y llamada a métodos. También se utiliza para definir precedencia en expresiones, contener expresiones para control de flujo y rodear las conversiones de tipo.
- **{ } - llaves**. Para contener los valores de matrices inicializadas automáticamente. También se utiliza para definir un bloque de código, para clases, métodos y ámbitos locales.
- [ ] **- corchetes**. Para declarar tipos matriz. También se utiliza cuando se referencian valores de matriz.
- **; - punto y coma**. Separa sentencias.
- **, - coma**. Separa identificadores consecutivos en una declaración de variables. También se utiliza para encadenar sentencias dentro de una sentencia for.
- **. - punto**. Para separar nombres de paquete de subpaquetes y clases. También se utiliza para separar una variable o método de una variable de referencia.

6. Sentencias

Una sentencia es una orden que se le da al programa para realizar una tarea específica, esta puede ser: mostrar un mensaje en la pantalla, declarar una variable (para reservar espacio en memoria), inicializarla, llamar a una función, etc. Las sentencias acaban con ; este carácter separa una sentencia de la siguiente. Normalmente, las sentencias se ponen unas debajo de otras, aunque sentencias cortas pueden colocarse en una misma línea. He aquí algunos ejemplos de sentencias

```
1 int i = 1;
2 import java.awt.*;
3 System.out.println("El primer programa");
4 rect.mover(10, 20);
```

En el lenguaje Java, los caracteres espacio en blanco se pueden emplear libremente como separadores, esto lo podremos ver en los sucesivos ejemplos. Es muy importante para la legibilidad de un programa la colocación de unas líneas debajo de otras empleando tabuladores. El editor del IDE nos ayudará automáticamente en esta tarea sin apenas percibirlo.

6.1. Bloques de código

Un bloque de código es un grupo de sentencias que se comportan como una unidad. Un bloque de código está limitado por las llaves de apertura y cierre . Como ejemplos de bloques de código veremos mas adelante la definición de una clase, la definición de una función miembro, una sentencia iterativa for, los bloques try ... catch, para el tratamiento de las excepciones, etc. (lo analizaremos en detalle mas adelante)

6.2. Expresiones

Una expresión es todo aquello que se puede poner a la derecha del operador asignación "=". Por ejemplo:

```
1 int x = 123;
2 double = (x+100)/4;
3 float area = circulo.calcularArea(2.5);
4 rectangulo r = new Rectangulo(10, 10, 200, 300);
```

- La primera expresión asigna un valor a la variable x.
- La segunda, realiza una operación
- La tercera, es una llamada a una función miembro calcularArea desde un objeto circulo de una clase determinada
- La cuarta, reserva espacio en memoria para un objeto de la clase Rectangulo mediante la llamada a una función especial denominada constructor.

6.3. Ejemplo 3.2

Dada una variable entera t que almacena el tiempo transcurrido del día en segundos, escriba las expresiones que permitan calcular las horas, minutos y segundos transcurridos del día (h, m, s). Ejemplo:
t= 3723; h= 1; m= 2; s= 3;

Ayuda

h= t / 3600;
m= (t % 3600) / 60; //El operador % devuelve el resto de la división entera.
s= t % 60;

Código: Creamos un nuevo proyecto en NetBeans IDE con el nombre U3_ejemplo02 y copiamos el siguiente código

```
package u3_ejemplo02;
/**
*
* @author FICH
*/
import javax.swing.JOptionPane;
public class U3_Ejemplo02 {
 /**
 * @param args the command line arguments
 */
 public static void main(String[] args) {
 // TODO code application logic here
 int t; //Almacenara el tiempo en segundos
 t= Integer.parseInt(JOptionPane.showInputDialog("Ingrese el tiempo en segundos"));
     //Lectura del tiempo
 int h= t / 3600; //horas
 int m= (t % 3600) / 60; //minutos
 int s= t % 60; //segundos
 JOptionPane.showMessageDialog(null,"Horas: "+h+" \nMinutos: "+m+" \nSegundos: "+ s
    ); //Ventana de salida
 }
}
```

Código 2: Ejemplo 3.2

Compilamos (F9) y Ejecutamos (F6).
Veremos la siguiente interfaz:

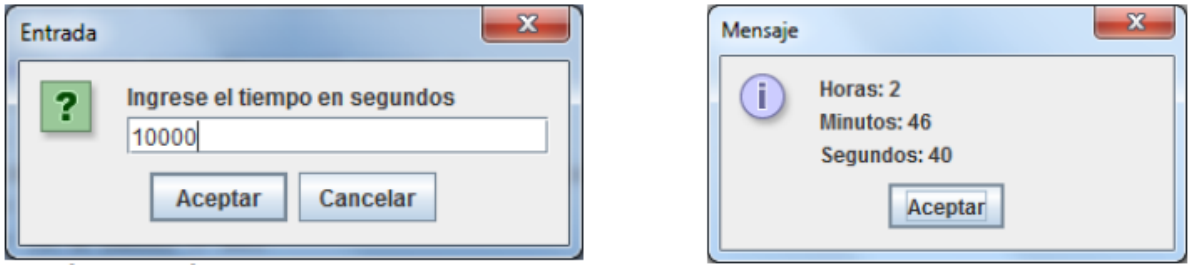

Figura 1: Salida Ejemplo 3.2

7. La clase Math

La clase Math representa la librería matemática de Java. Las funciones que contiene son las de todos los lenguajes, parece que se han metido en una clase solamente a propósito de agrupación, por eso se encapsulan en Math.

7.1. Funciones matemáticas

Si se importa la clase, se tiene acceso al conjunto de funciones matemáticas estándar.

Math.abs(x) para int, long, float, double	Math.ceil(double)
Math.sin(double)	Math.floor(double)
Math.cos(double)	Math.rint(double)
Math.tan(double)	Math.pow(a,b)
Math.asin(double)	Math.round(x) para double y float
Math.acos(double)	Math.random() devuelve un double
Math.atan(double)	Math.max(a,b) para int, long, float y double
Math.atan2(double,double)	Math.min(a,b) para int, long, float y double
Math.exp(double)	Math.E para la base exponencial
Math.log(double)	Math.PI para PI
Math.sqrt(double)	

7.2. Ejemplo 3.3

Escriba un proyecto que genere dos números reales (con decimales) entre 0 y 100, aleatoriamente y devuelva el de mayor valor absoluto.

Ayuda

1. El método max devuelve el mayor entre 2 valores. "Math.max"
2. El método abs devuelve el valor absoluto de un número. "Math.abs"
3. La función random() devuelve un numero al azar entre 0 y 1.

Código: Creamos un nuevo proyecto en NetBeans IDE con el nombre U3_ejemplo3 y copiamos el siguiente código:

```
1  package u3_ejemplo03;
2  /** @author FICH
3  */
4  import static java.lang.Math.*;
5  import javax.swing.JOptionPane;
6  public class U3_Ejemplo03 {
7   /**
8   * @param args the command line arguments */
9   public static void main(String[] args) {
10  // TODO code application logic here
11  double x, y; //declaro x, y
12  x= Math.random()*100; //asigno valores aleatorios
13  y= Math.random()*100;
14  double maxAbs = Math.max (Math.abs (x), Math.abs (y));// utilizando max y abs
        asigno el mayor
15 //muestro por pantalla el resultado.-
16  JOptionPane.showMessageDialog(null,"1er. Numero: "+x+"\n2do. Numero: "+y+"\nMayAbs
        : "+maxAbs);
17  }
18 }
```

Código 3: Ejemplo 3.3

Compilamos (F9) y Ejecutamos (F6).

7.3. Ejemplo 3.4

Calcular el área de un círculo ingresando el radio por teclado.

Ayuda

Area = PI * $(radio)^2$

Código: Creamos un nuevo proyecto en NetBeans IDE con el nombre U3_ejemplo4 y copiamos el siguiente código.

```
package u3_ejemplo04;
/**
*
* @author FICH
*/
import static java.lang.Math.*;
import javax.swing.JOptionPane;
public class U3_Ejemplo04 {
 /**
 * @param args the command line arguments
 */
 public static void main(String[] args) {
 // TODO code application logic here
 double r, a;
 r = Double.parseDouble(JOptionPane.showInputDialog("Ingrese el radio")); // radio
     de
un circulo
 a = Math.PI * Math.pow(r,2); // calcula area
 JOptionPane.showMessageDialog(null,"Area del circulo es " + a);
 }
}
```

Código 4: Ejemplo 3.4

Compilamos (F9) y Ejecutamos (F6).

8. Resumen

1. Hemos visto algunas de las palabras reservadas lenguaje Java
2. Los identificadores en Java deben comenzar con una letra, '$' o '_' y no pueden llevar espacios en su nombre y no deben tener caracteres especiales como +, -, ? y otros.
3. Los identificadores que declaramos en nuestro programa no deben coincidir con ningunos de los identificadores reservados para el lenguaje Java (como abstract, continue, for, new, switch, etc.).
4. Las constantes poseen un único valor en todo el programa.
5. Las variables son posiciones de memoria donde almacenamos datos. Pueden cambiar su valor en diferentes lugares del programa mediante la sentencia de asignación o lectura. Siempre deben declarase antes de ser empleadas
6. Mediante las expresiones podemos combinar variables y constantes mediante operadores para obtener resultados. Los operadores tienen un orden de aplicación (precedencia) que puede modificarse con el uso de paréntesis (de forma similar a como lo hacemos en matemáticas).
7. Los tipos de datos determinan la forma en que se almacena la información en memoria y la forma en que se puede operar sobre esa información. Cuando declaramos variables indicamos tipo, luego su nombre y a continuación podemos o no asociarle un valor a esa variable. Los tipos de datos determinan los valores posibles que puede contener una variable (por ejemplo una variable de tipo byte no puede contener el número 3,45 ni el número 345).

8. Los tipos de datos simples ordinales son aquellos sobre los que se puede establecer un orden y una secuencia. Después del caracter 'A' sigue el caracter 'B' pero no podemos saber cual número sigue al 3,45 (por eso los reales o flotantes no son ordinales).

9. Los tipos de datos simples ordinales más importantes son: byte, int, bool y char. Dos tipos de datos simples NO ordinales usuales son: float y double

10. El cociente entre dos enteros da como resultado un entero. Por ejemplo, al dividir 20 entre 7 nos da como resultado 2.

11. El operador (%) módulo, da como resultado el resto de la división entera. Por ejemplo 20 %7 da como resultado 6 que es el resto de la división entre 20 y 7.

12. La tabla de códigos ASCII define el orden de los caracteres.

13. El tipo lógico se llama bool en Java y permite almacenar valores de verdad: verdadero (true) y falso (false).

14. Existen varios operadores que permiten construir expresiones de tipo lógico. Por ejemplo: $<, >, =, <=, >=, <>$