CENTRO DE TECNOLOGIAS DE INFORMACION Y COMUNICACION

JAVA BASICO

Instructor: LENIN LEON
Email: leninleon0720@gmail.com
Celular: 6234-5608

Identificadores

En Java, un identificador empieza con una letra, el carácter de subrayar o el signo $. Los demás caracteres pueden contener dígitos. Todos los identificadores son sensibles a mayúsculas / minúsculas.
Ejemplos de identificadores validos:

Identificadores

variable
nombreUsuario
Nombre_Usuario
_numero
$cadena

Los últimos tres ejemplos son muy poco usados en la generalidad de los programas. Los identificadores pueden contener palabras reservadas, pero no pueden ser palabras reservadas; por ejemplo, es valido integer, pero no int.

Palabras Reservadas

abstract	do	implements	private	throw
boolean	double	import	protected	throws
break	else	instanceof	public	transient
byte	extends	int	return	true
case	false	interface	short	try
catch	final	long	static	void
char	finally	native	super	volatile
class	float	new	switch	while
continue	for	null	synchronized	
Default	if	pack	this	

Palabras Reservadas

Nota.- En Java, true, false y null se escriben en minúsculas, al contrario que en C++.
No existe un operador sizeof; el tamaño y representación de todos los tipos es fija y no es dependiente de la implantación.
Las palabras goto y const no se usan en Java.

Tipos de datos

Java define ocho tipos de datos primitivos y uno especial. Se pueden agrupar en: lógicos, textuales, integrales y de punto flotante.
El tipo de dato boolean sólo tiene dos valores: false y true. En C/C++ se permite que valores numéricos sean interpretados como valores lógicos, pero no es el caso de Java; sólo se permiten valores lógicos.
Los tipos de datos textuales son: char y String. Los caracteres se representan por el tipo de dato char. Los caracteres se encierran entre apóstrofes.

Tipos de datos

El tipo String, que no es primitivo, es usado para representar secuencias de caracteres. Una cadena de caracteres se encierra entre comillas.
"Esto es un mensaje"

Tipos de datos

Los tipos integrales son: byte, short, int y long. Todos son números con signo. Los números se pueden representar en forma decimal, octal o hexadecimal.

2	Valor decimal es dos
077	El cero que inicia indica un valor octal
0xBC27	0x indica un valor hexadecimal

Modificadores

Dentro de las palabras reservadas, Java utiliza las siguientes para modificar el acceso a una variable, clase o función y se colocan al inicio de la declaración: public, protected, default, private.

Modificadores

El modificador public da acceso a cualquier objeto externo.
public int numero; // cualquier objeto puede accesar a esta variable

El modificador protected da acceso a objetos que son parte del mismo paquete, y las subclases. (Más adelante se explica el concepto de paquete)

Modificadores

El modificador default da acceso a objetos que son parte del mismo paquete. Sin embargo, en los programas no se especifica el modificador porque no hay una palabra para ello.

```
int numero; // acceso default
```

Modificadores

El modificador private da acceso únicamente a la clase que lo contiene.
private int numero; // únicamente lo puede accesar la clase

Convenciones en la programación

Clases.- Los nombres de las clases deberían ser sustantivos, utilizando mayúsculas para la primera letra y minúsculas para las restantes, y se pueden mezclar varios sustantivos.

class CuentaBancaria

Convenciones en la programación

Interfaces.- Los nombres de las interfaces deberían tener la primera letra mayúscula, como en los nombres de clase.

interface Cuenta

Convenciones en la programación

Métodos.- Los nombres de los métodos deberían ser verbos, todo el verbo en minúscula. Se pueden agregar sustantivos con la primera letra en mayúscula. Evitar el uso de subrayas.

void revisarCuenta()

Convenciones en la programación

Constantes.- Las constantes de tipos de datos primitivos deberían escribirse completamente en mayúsculas y separadas las palabras por subrayas. Las constantes de objeto pueden combinar mayúsculas y minúsculas

final int MAX_CREDITO

Convenciones en la programación

Variables.- Todas las variables deberían ser en minúsculas, y si se agregan palabras se separarán con una letra mayúscula. Evitar el uso del signo $.
primerUsuario

Las variables deben tener significado e indicar su uso. Las variables de una letra deberían evitarse, excepto las que suelen usarse en ciclos (x, y, i, j) para controlarlo.

Convenciones en la programación

Otras convenciones de la programación incluyen el uso de llaves ({}) alrededor de un bloque de instrucciones, incluso cuando se trate de una sola instrucción, ya que esto ayuda en el mantenimiento del programa.

```
if(condición)
{
bloque
}
```

El espaciado ayuda en la comprensión del programa. Se sugiere escribir una instrucción por línea y usar indentación de uno o dos espacios.

Convenciones en la programación

Los comentarios también ayudan en la comprensión y mantenimiento del programa al dar una descripción clara de lo que hace cada función y el uso de las variables.
Ejemplo:

```
// primer programa en Java
public class HelloWorld
{
    public static void main(String argv[])
    {
        System.out.println("Hello world!");
    }
}
```

Convenciones en la programación

En detalle:

// primer programa en Java

La primera línea es un comentario.

public class HelloWorld

{

Las siguientes dos líneas son la declaración de la clase, que al momento de ser compilado el programa, generará un archivo .class. Es importante que el nombre de la clase sea el mismo que el nombre del archivo:

si la clase se va a llamar HelloWorld, el archivo se debe llamar HelloWorld.java.

Convenciones en la programación

public static void main(String argv[])
{
En las siguientes dos líneas se declara el inicio del programa. Para que el intérprete de Java pueda ejecutar el programa debe tener la misma sintaxis (excepto para el nombre del parámetro de main). Se declara public para que lo pueda accesar el intérprete de Java. Se declara static porque no se ha creado algún objeto y no se crea una instancia. Se declara void porque no se regresa valor alguno. En este ejemplo no se va a esperar parámetros de la línea de comandos. En argv[] se guardan los parámetros y la primera posición contiene el primer parámetro, no el nombre del programa:
argv[0] parametro1
argv[1] parametro2

Convenciones en la programación

System.out.println("Hello world!");
La siguiente línea muestra el uso de una clase y un método que imprime en la salida estándar (la pantalla) un mensaje.

```
  }
}
```

Finalmente se termina el bloque del método main y la declaración de la clase. Una vez que se tiene el código fuente en el archivo HelloWorld.java se usa el compilador de Java de la siguiente manera:
javac HelloWorld.java

Convenciones en la programación

Si el compilador no regresa mensajes de error, se habrá creado un nuevo archivo HelloWorld.class en el mismo directorio que el código fuente.

Después de la compilación, se puede ejecutar el programa y ver el resultado usando el intérprete de Java:
java HelloWorld

Laboratorio 1

Modificar el programa de “HelloWorld.java” para que reciba un nombre por medio de un parámetro en la línea de comandos y que lo imprima en forma de saludo. Por ejemplo:

C:\>java Hello JoseLuis

Esto deberá desplegar una salida:

Gusto en conocerte JoseLuis

Inicialización de variables

Java no permite que una variable tenga un valor indefinido. Cuando un objeto es creado, sus variables son inicializadas con los siguientes valores:

- byte 0
- short 0
- int 0
- long 0L
- float 0.0F
- double 0.0D
- char '\u0000' (NULO)
- boolean false
- todas las referencias null

Inicialización de variables

Si algún objeto hace referencia a algo con valor de null, creará una excepción (un error que es manejable).
Para evitar que las variables tengan valores indeseables, se debe asignárseles algún valor útil. El compilador estudia el código para determinar que cada variable ha sido inicializada antes de su primer uso. Si el compilador no puede determinar esto, entonces ocurre un error en tiempo de compilación.

Inicialización de variables

```
public void calcula()
{
  int x = (int)(Math.random() * 100);
  int y;
  int z;
  if(x > 50)
  {
    y = 9;
  }
  z = y + x;
  // el posible uso antes de la inicialización de y creara un
  error de compilación
}
```

Expresiones lógicas

Los operadores relacionales y lógicos regresan un valor boolean. En Java no existe conversión automática de

int a boolean, como en C++.
int i = 1;
if(i) // error en tiempo de compilación
if(i != 0) // correcto

Operadores y su Precedencia

Los operadores en Java son muy similares en estilo y función a aquellos en C y C++.

El operador + se puede utilizar para concatenar cadenas de caracteres, produciendo una nueva:

```
String saludo = "Sr. ";
String nombre = "Luis " + "Torres";
String persona = saludo + nombre;
```

Operadores y su Precedencia

Los operadores && (and) y || (or) realizan una evaluación corta en expresiones lógicas. Por ejemplo:

```
String unset = null;
if((unset != null) && (unset.length() > 5))
{
  // hacer algo con unset
}
```

Operadores y su Precedencia

La expresión que forma a if() es legal y completamente segura. Esto es porque la primera subexpresión es falsa, y es suficiente para probar que toda la expresión es falsa. El operador && omite la evaluación de la segunda subexpresión y una excepción de null pointer es evitada. De forma similar, si se usa el operador || y la primera subexpresión es verdadera, la segunda subexpresión no es evaluada porque toda la expresión es verdadera.

Cast

Cuando la asignación de valores no es compatible por los tipos de datos, se usa un cast para persuadir al compilador de reconocer tal asignación. Esto se puede hacer para asignar un long a un int, por ejemplo.

```
long bigValue = 99L;
int smallValue = (int)(bigValue);
```

- No es necesario el segundo grupo de paréntesis, los que encierran a bigValue, pero es muy recomendable dejarlos.

Flujo de programa

Sentencia if/else

Permite elegir una de dos opciones. La sintaxis básica de la sentencia if/else es:

```
if(condición)
{
instrucción_o_bloque
}
else
{
instrucción_o_bloque
}
```

Flujo de programa

Sentencia if/else

Ejemplo:

```
int aleatorio = (int)(Math.random() * 100);
if(aleatorio < 50)
{
System.out.println("menor a 50");
}
else
{
System.out.println("mayor o igual a 50");
}
```

Flujo de programa

Sentencia switch.

Permite seleccionar una de varias opciones. La sintaxis para switch es la siguiente:

```
switch(expresión_a_evaluar)
{
  case valor1:
    instrucciones;
    break;
  case valor2:
    instrucciones;
    break;
  default:
    instrucciones;
    break;
}
```

Flujo de programa

Sentencia switch.

El valor de expresion_a_evaluar debe ser compatible con el tipo int, como short, byte y char. No se permite evaluar long o valores de punto flotante.

Flujo de programa

Sentencia switch.
Ejemplo:

```
switch(colorNum)
{
  case 0:
    setBackground(Color.red);
    break;
  case 1:
    setBackground(Color.green);
    break;
  case 2:
    setBackground(Color.blue);
    break;
  default:
    setBackground(Color.black);
    break;
}
```

Flujo de programa

- La sentencia for.
  Permite realizar una serie de instrucciones mientras se cumple una condición. La sintaxis básica para for es:

```
for(inicialización;condición;alteración)
{
  instrucciones;
}
```

Flujo de programa

La sentencia for.
Ejemplo:

```
int x;
for(x = 0;x < 10;x++)
{
  System.out.println("dentro de for");
}
System.out.println("fin de for");
```

Flujo de programa

La sentencia for.
El tercer parámetro puede ser tanto de incremento como de decremento, y no únicamente de uno en uno. Java permite el uso de comas dentro de la declaración de for, como en C, por lo que lo siguiente es legal:

```
for(i = 0, j = 0;j < 10;i++,j++)
```

Flujo de programa

La sentencia for.
En el ejemplo anterior, la variable x es "visible" en el método en el que es declarada. Se puede usar una variable que sea visible únicamente para el ciclo for:

```
for(int x=0;x<10;x++)
{
...
}
// una vez terminado el ciclo, x ya no puede ser accesada
```

Flujo de programa

La sentencia while.
Permite realizar una serie de instrucciones mientras se cumple una condición. La sintaxis básica de while es:

```
while(condición)
{
  instrucciones;
}
```

Flujo de programa

Ejemplo:

```
int i = 0;
while(i<15)
{
  system.out.println("dentro de while");
  i+=2;
}
```

Flujo de programa

La sentencia do/while.
Permite realizar una serie de instrucciones hasta que deje de cumplirse una condición. La sintaxis básica de la sentencia es:

```
do
{
  instrucciones;
}while(condición);
```

Paquetes

Java provee el mecanismo de paquetes (package) como una forma de organizar las clases. Se puede indicar que las clases en el código fuente van a pertenecer a un paquete empleando la palabra package.

```
package empresa.sistemas;
public class Empleado
{
...
}
```

- $ javac -d <ruta> Archivo.java

Paquetes

La declaración de paquete, si la hay, debe estar al inicio del código fuente, puede estar precedida únicamente de comentarios. Solo se permite una declaración package por archivo fuente. Los nombres de los paquetes son jerárquicos, separados por puntos. Por lo general, los elementos de los paquetes son escritos enteramente en minúsculas.

Paquetes

Una vez compilado el archivo, puede ser usado por otro mediante la sentencia import, que indica donde se encuentran los paquetes. Import debe preceder a todas las declaraciones de clases.

```
import empresa.sistemas.*;
public class JefeArea extends Empleado
{
  String departamento;
  Empleado subordinados[];
...
}
```

Laboratorio 2

1. Realizar un programa que determine si un número ingresado por el usuario es primo o no.

Instructor: Lenin León email: leninleon0720@gmail.com celular: 6234-5608