# JAVA Y LA PROGRAMACION ORIENTADA A OBJETOS

PROFESOR ERIC GAMESS

D.E.A. en Informática industrial de la Universidad de Toulouse.
Actualmente se desempeña como profesor de la Universidad del Valle,
Departamento de Ciencias de la Computación.
E-mail: eric@borabora.univalle.edu.co

## INTRODUCCIÓN

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de la década de los años 90. Java es inspirado en gran parte de C++ y a un nivel menor de Objective C y Smalltalk.

Según la Sun Microsystems, Java fue creado inicialmente porque C++ no era adecuado para ciertas tareas. El éxito del lenguaje empieza en 1995, cuando Netscape anunció que su visualizador, Navigator, trabajaría con programas Java insertados en las páginas Web. La idea era ofrecer a los desarrolladores de páginas HTML, una forma de crear documentos interactivos y animados.

## SIMILITUDES Y DIFERENCIAS ENTRE C++ Y JAVA

Una forma para hablar de Java es compararlo con C++. A continuación se resaltan las similitudes y diferencias entre los dos lenguajes.

- Java cuenta con ocho tipos primitivos. El tipo **boolean** hace parte de Java, y sus posibles valores son **false** y **true**. Los tipos **byte, short, int** y **long** permiten declarar variables enteras con signo sobre 1, 2, 4 y 8 bytes respectivamente. Los tipos **float** y **double** permiten declarar variables reales sobre 4 y 8 bytes, respectivamente. El tipo **char** se utiliza para declarar variables de caracteres. Java utiliza Unicode para representar los caracteres. Un carácter en Unicode es representado sobre 16 bits. Unicode es la unificación de varios conjuntos de caracteres e incluye entre otros, los caracteres latinos, griegos, cirílicos, hebraicos, etc.
- Java cuenta con las mismas estructuras de control de C++ **(if, if/else, for, while, do/while** y **switch).** La única diferencia es que una condición en Java debe ser de tipo **boolean,** lo que hace el código más claro.
- Los apuntadores son seguramente la más poderosa herramienta de C++ pero también la más peligrosa. Java no cuenta con apuntadores, lo que asegura que el programador no puede tener acceso a una posición de memoria que no le pertenece. Al no tener apuntadores, los programas de

Java son más fáciles de leer. La carencia de dichos apuntadores obliga al programador a pasar los argumentos de una función por valor, lo que tiene como consecuencia que no es posible realizar en Java una función que intercambie el contenido de dos variables enteras.

- Java no cuenta con la sobrecarga de operadores como C++. Esta falta hace que el lenguaje sea más fácil de aprender, pero también complica la escritura y la legibilidad de ciertas aplicaciones. Por ejemplo, es mucho más claro utilizar el operador + o el operador* cuando se trata de hacer la suma o el producto de dos números complejos, en lugar de utilizar nombres de métodos.
- Java cuenta con un recolector de basura, lo que simplifica mucho el desarrollo de programas dinámicos. De hecho, no existe en Java, el operador **delete** de C++. A pesar de la evolución del hardware y de las mejoras hechas sobre los algoritmos de recolección de basura, este proceso es muy costoso en tiempo y hace que los programas de Java sean más lentos que sus equivalentes de C++.
- El código fuente de C++ no es portable y muchas de sus primitivas dependen del compilador utilizado y de la arquitectura. Se ha hecho un gran esfuerzo para que el código fuente de Java sea portable. No existe ninguna primitiva en Java que depende de la arquitectura. Por ejemplo, las variables de C++ que son guardadas en registros **(register)** no existen en Java. Los tipos primitivos de Java ocupan siempre la misma cantidad de bytes en la memoria y eso es totalmente independiente del compilador utilizado, mientras que algunos compiladores de C++ representan las variables de tipo **int** sobre dos bytes y otros sobre cuatro bytes.
- En Java todos los métodos son virtuales, mientras que en C++ un método puede o no ser virtual.
- Java representa un compromiso entre un lenguaje totalmente compilado (como C++) y totalmente interpretado (como Smalltalk o Perl). El compilador de Java (javac) produce un código intermedio, independiente de la arquitectura, que se llama el código de bytes. El interpretador de Java (java) ejecuta dicho código de bytes. Desde hace poco, algunas empresas proponen productos para generar ejecutables a partir de Java. Para más información sobre este tema, ver «Creación de ejecutables».
- En Java todas las funciones y variables deben ser parte de una clase, a diferencia de C++ que permite que las funciones y variables se declaren en forma global.
- Java, en su versión actual, no soporta las plantillas de C++. Es una de las grandes críticas que se hacen al lenguaje, pero Java todavía está en evolución y es muy probable que las futuras versiones tengan plantillas.
- En su versión actual, Java no cuenta con funciones de entrada para leer un tipo primitivo desde una terminal a diferencia de C y C++ donde el conjunto de funciones para leer datos es muy potente **(scanf, gets, getchar, cin,...)**
- Java no soporta la herencia múltiple. Sin embargo, Java permite que una clase herede múltiples interfaces que según un artículo publicado por la Sun Microsystem, ofrece la mayoría de los beneficios de la herencia múltiple de C++ y evita sus dificultades.
- La mayoría de los operadores de C++ son implementados en Java. En Java, los operadores lógicos (NOT), && (AND con corto recorrido) y || (OR con corto recorrido) solamente se aplican a operandos booleanos, lo que hace que sus programas sean más claros. Java cuenta con nuevos operadores lógicos como & (AND sin corto recorrido), |(OR sin corto recorrido) y ^ (XOR). Note que como los enteros de Java tienen signo, se ha tenido que introducir el operador >>> para hacer un desplazamiento hacia la derecha entrando ceros (desplazamiento lógico hacia la derecha).
- Java tiene una clase raíz llamada **Object.** En Java, todas las clases son derivadas de forma directa o indirecta de la clase **Object.** En C++, no hay clase raíz.
- Java cuenta con un poderoso operador llamado **instanceof.** Con este operador se puede saber si una referencia puede remitir a una cierta clase. Por ejemplo, si **a** es una referencia, **a instanceof String** devuelve **true** si **a** puede referir un objeto de tipo **String.** Este operador no existe en C++ y permite implementar en forma muy elegante la noción de polimorfismo. Ver «Segundo ejemplo de programación en Java.»
- Las bibliotecas de C++ son similares a los paquetes de Java.
- Java no cuenta con soporte para argumentos de función por omisión como en C++.
- En C++, el alcance de las variables definidas en la parte **inicialización** de un **for** va desde su declaración hasta el final del bloque que contiene el **for**. Eso era un bug de concepción de C++. Java lo tiene corregido, así que el alcance de las variables de un **for** se limita a dicha estructura.
- Todos los objetos de Java son creados en la memoria **heap** por medio del operador **new.** En C++, los objetos pueden ser creados en la memoria **heap** y en la pila.
- En Java no existe la noción de función amiga o clase amiga como se conoce en C++.

## LA CLASE CONSOLE

La realización de programas sencillos, siempre pasa por la lectura y escritura de datos. Como Java no cuenta con funciones para hacer entradas desde una terminal, por lo menos es cierto en las versiones actuales, en los ejemplos de programas siguientes se utiliza una clase llamada **Console** que fue implementada por el profesor David Eck de la Universidad de Hobart and William Smith. La creación de un objeto de tipo **Console**, abre una ventana donde el programador puede leer y visualizar datos. Para cerrar la ventana, hay que mandar el mensaje **close()** a la instancia. Los métodos de esta clase son **getByte(), getShort(), GetInt(), getLong(), getFloat(), getDouble(), getBoolean(), getChar()** y **getIn()** para leer un dato de tipo **byte, short, int, long, float, double, boolean, char** y **String** respectivamente. Con el método **put** se puede visualizar un dato de tipo **byte, short, int, long, float, double, boolean, char** o **String** sin salto de línea. El método **putIn** es similar al método **put** con la diferencia que visualiza el dato y pasa a la línea siguiente. Usted puede recuperar la clase **Console** por ftp (ftp://borabora.univalle.edu.co/pub/eric/java).

## INSTRUCCIONES PARA EJECUTAR UN PROGRAMA

Se debe digitar el código fuente de Java dentro de un archivo con extensión **.java.** Es preferible que el nombre del archivo sea el de una clase que contiene, ya que el compilador genera un archivo de código de bytes para cada clase implementada dentro de dicho archi-

vo. Si el archivo fuente se llama **Ejemplo.java,** se compila con:

**javac Ejemplo.java**

Observe que el compilador genera un archivo de código de bytes para cada clase declarada en el archivo fuente. Se interpreta el código de bytes con:

java Ejemplo

## PRIMER EJEMPLO DE PROGRAMA EN JAVA

El programa siguiente permite que el usuario entra un número, y visualiza el factorial de dicho número calculándolo de forma iterativa y recursiva. Este programa no tiene un interés particular en la programación orientada a objetos y tiene como único objetivo familiarizar al programador con el uso del compilador y del interpretador.

```
//Ejemplo de cálculo del factorial
class Factorial
{
public static void main(String[] args)
{
Console io=new Console();
io.put(«\n\nDigite un entero...»);
int n=io. getInt();
io.putln(«El factorial de «+n+» es «+facto1(n));
io.putln(«El factorial de «+n+» es «+facto2(n));
io.close();
}
private static int facto1(int n)
{
if(n==0)
return 1;
else
return n*facto1(n-1);
}
private static int facto2(int n)
{
int result=1;
for(int i=2; i<=n; i++)
result *=i;
return result;
}
}
```

El método llamado **main** en la clase **Factorial** es el primer método a ser ejecutado. En este método se lee un entero llamado **n,** y se visualiza dos veces el factorial de **n,** calculado por los métodos **facto1** y **facto2.** El método **facto1** calcula el factorial de un entero de forma recursiva, mientras que **facto2** lo hace de forma iterativa. Para ejecutar el programa:

1. Recupere por ftp el archivo **Console.class** (ftp://borabora.univalle.edu.co/pub/eric/java)
2. Digite el programa anterior en un archivo llamado **Factorial.java**
3. Genere el código de bytes con **javac Factorial.java**
4. Ejecute el código de bytes con **java Factorial**

## SEGUNDO EJEMPLO DE PROGRAMA EN JAVA

El siguiente programa permite ilustrar el uso del operador **instanceof** y muestra una forma muy elegante que tiene Java para implementar el polimorfismo. Al ejecutar el programa aparece un mensaje para seleccionar si se va a trabajar con números complejos o con números enteros. Si el usuario escoge la opción C (Complejos), el programa lo guía para leer dos números complejos y visualiza la suma de dichos complejos. Si el usuario escoge la opción E (Enteros), el programa hace lo mismo, pero esta vez con dos números enteros, si no el programa visualiza un mensaje de error.



```
class Complex
{
double r, i;
public Complex(double r, double i)
{
this.r=r;
this.i=i;
}

public String toString()
{
if(i==0.0)
return «»+r;
else if(i<0.0)
return «»+r+i+'i';
else
return «»+r+'+'+i+'i';
 }
}
class Polimorfismo
{
public static void main(String args[])
{
Console io=new Console();
Object a,b,s;
io.put(«\n\n Trabajar con [C]omplejos o [E]enteros...»);
char resp=io.getChar();
if(Character.to UpperCase(resp)=='C')
{
double r,i;
io.put(«Parte real del complejo 1 ...»);
r=io.getDouble();
io.put(«Parte imag del complejo 1 ...»);
i=io.getDouble();
a=new Complex(r,i);
io.put(«Parte real del complejo 2 ...»);
r=io.getDouble();
io.put(«Parte imag del complejo 2 ...»);
i=io.getDouble();
b=new Complex(r,i);
}
else if(Character.toUpperCase(resp)=='E')
{
io.put(«Entre el valor del entero 1 ...»);
a=new Integer(io.getInt());
io.put(«Entre el valor del entero 2 ...»);
b=new Integer(io.getInt());
}
else
{
io.putln(«Error»);
io.close();
return;
}
s=suma(a,b);
io.putln(«La suma es:»+s);
io.close();
}

private static Object suma(Object a, Object b)
{
if(a instanceof Complex)
{
Complex valorA=new Complex(((Complex)a).r,((Complex)a).i);
Complex valorB=new Complex(((Complex)b).r,((Complex)b).i);
return new Complex(valorA.r+valorB.r, valorA.i+valorB.i);
}
else
{
int valorA=((Integer)a).intValue();
int valorB=((Integer)b).intValue();
return new Integer(valorA+valorB);
}
}
   }
```

En la primera parte del programa, se implementa una clase llamada **Complex** que permite definir las operaciones sobre los números complejos que son necesarias con el fin de realizar el programa. El primer método es un creador y el segundo permite convertir un número complejo a una cadena de caracteres

de forma implícita. En la clase **Polimorfismo,** el método **main** se encarga de que el usuario seleccione si va a trabajar con números complejos o enteros, hace la lectura de los dos números y visualiza la suma. El método **suma** es polimorfo y hace la suma de sus dos *argumentos teniendo en cuenta si son* números complejos o enteros. Esta forma de polimorfismo no existe en C++ pero se puede implementar una función polimorfa en C++ equivalente a **suma** por medio de plantillas.

## CREACION DE EJECUTABLES

Al ser interpretado, Java es un lenguaje lento. Ciertas aplicaciones implementadas en Java son del orden de veinte a treinta veces más lentas que sus equivalentes de C++. Hoy en día, existen algunos productos que permiten *obtener un ejecutable a partir de un archivo* fuente de Java o de un archivo de código de bytes. A continuación, se citan dos productos:

1. Supersede de la sociedad Asymetrix. Ver http://www.asymetric.com/
2. Microsoft SDK (utilitario jexegen). Ver http://www.microsoft.com/java/sdk/default.htm

Al generar un ejecutable, se pierde la independencia de la arquitectura. No he tenido la oportunidad de trabajar con estos productos, pero es muy probable que *ellos sean limitados a ciertas aplicaciones.*

## PLATAFORMAS QUE SOPORTAN JAVA

Al día de hoy, Java ha sido adaptado a muchas plataformas diferentes. Sun Microsytems distribuye en forma gratuita su JDK (Java Development Kit) para Windows 95, Windows NT, MacOS 7.5, Sparc Solaris y X86 Solaris. Se puede recuperar el kit de desarrollo (compilador, interpretador, depurador, visualizador de Applets, ...) de Sun Microsystems por ftp (ftp://ftp.javasoft.com/pub).

IBM ha portado el JDK para Windows 3.1, OS2 y AIX. Ver http://ncc.hursley.ibm.com/javainfo/.

Existe una versión del JDK para Linux. Ver http://java.blackdown.org/java-linux.html.

El producto Guavac de la Free Software Foundation (GNU) contiene dos utilitarios. El primero llamado **guavac** es un compilador que genera el código de bytes a partir de un archivo fuente de Java (el equivalente de **javac** en el kit de desarrollo de SUN). El segundo llamado **guavad** es un desensamblador. *Al día de hoy, en la distribución* de Guavac, no hay un interpretador. Los dos utilitarios están escritos en C++ y se pueden compilar en cualquier plataforma equipada del compilador C++ de la GNU. Ver

http://HTTP.CS.Berkeley.EDU/~engberg/guavac/

Además, algunas empresas como Symantec (Visual Café) y Microsoft (J++) han creado un ambiente de desarrollo de programas para Java que funciona bajo Windows 95 y Windows NT donde se encuentran herramientas de muy buena calidad (compilador, interpretador, depurador, visualizador de Applets, editor de Applets, editor de menús, etc...)

## MAS INFORMACION SOBRE JAVA

Si usted está interesado en Java, puede buscar información en la red Internet con un software de búsqueda. He *seleccionado algunos sitios como http://* /www.javasoft.com que es el sitio oficial de Sun Microsystems donde se puede encontrar información y documentación sobre Java,http://www.gamelan.com donde se puede encontrar ejemplos de programas y tutoriales sobre el lenguaje.

Se han creado varios grupos de noticias sobre Java. Si usted tiene acceso a grupos de noticias, le recomiendo los grupos comp.lang.java.programmer,comp.lang.java.misc y comp.lang.java.announce.

La empresa Black Dirt ha desarrollado un utilitario llamado **V Bto.Java** que permite convertir un archivo fuente de *Microsoft* Visual Basic en su equivalente Java. Ver http://www.blackdirt.com.

He desarrollado algunos ejemplos de programas en Java, donde la clave es la programación orientada a objetos. Si usted está interesado en ellos, los puede recuperar por ftp(ftp://boraboora.univalle.edu.co/pub/eric/java).

## CONCLUSIONES

Se ha criticado mucho C++ como lenguaje de programación orientado a objetos. Java, en esta parte, no me parece superior a C++. Por mucho, el éxito del lenguaje Java se debe a su particularidad de animar las páginas Web. Java es de todos modos un lenguaje que va a evolucionar en los futuros años. Se puede observar al nivel mundial que muchas empresas de desarrollo de software ya han invertido en Java y entre ellas están Sun Microsystems, Microsoft, Borland, Oracle, Symantec, Asymetrix, Intermetrics, Metrowerks. En los años que vienen, seremos testigos de una gran batalla entre C++ y Java.

## BIBLIOGRAFIA

*The Java Handbook,* Patrick Naughton, Osborne McGraw-Hill.

*Java in a Nutshell,* David Flanagan, O'Reilly & Associates.

*The Java Language Specification,* Gosling, Joy and Steele, Addison Wesley.