Java en el Mundo Académico

W. Iván Coro Cruz

Departamento de Ingeniería de Sistemas
Universidad Católica Boliviana San Pablo
Cochabamba, Bolivia
e-mail: coro@ucbcba.edu.bo

Java es un lenguaje de programación relativamente nuevo que ha sido fuertemente difundido desde sus inicios a mediados de la década de los noventa. Su gran influencia en la industria, aunque a veces demasiado saturada de propaganda, ha causado que el mundo académico adopte Java como lenguaje introductorio en algunos casos y en otros como lenguaje requerido en programas de pregrado en Ciencias de la Computación y ramas afines. En este artículo exploraremos el pasado y la situación actual de Java dentro del contexto académico global, su evolución con respecto a la enseñanza del mismo y proyecciones futuras.

Java en el mundo real

La tarea de reemplazar otros lenguajes de programación por Java en cursos introductorios universitarios es más complicada de lo que uno puede pensar, ya que existen muchos puntos de vista válidos a considerar, como ser la facilidad de aprendizaje, sintaxis del lenguaje, demanda de la industria y entornos de desarrollo disponibles, entre otros. Desde fines de la década de los noventa, la demanda de profesionales competentes en Java tuvo un incremento espectacular, primordialmente en países desarrollados donde empresas gigantescas poseen aplicaciones empresariales complejas. Cabe resaltar que la demanda de profesionales con conocimientos de Java se presenta en campos diferentes: demanda de conocimientos de Java versión Empresarial (J2EE[1]) impulsada por la industria de países desarrollados, demanda para crear aplicaciones pequeñas independientes que pueden ser incrustadas en páginas Web, de cualquier manera, esta demanda de la industria ha causado el replanteamiento de muchos programas universitarios para adoptar Java como lenguaje requerido.

Desventajas al enseñar Java

Uno de los problemas que frenaron la adopción de Java como primer lenguaje en algunas universidades está relacionado con el grado de complejidad inicial que Java presenta para estudiantes principiantes de programación. Otros lenguajes como Pascal permiten enseñar un concepto a la vez, lo cual no sucede con Java, ya que el estudiante debe aprender varios aspectos en paralelo. Por ejemplo consideremos el programa más simple que se puede implementar en Java:

```
class HolaMundo
{
   public static void main(String args[])
   {
       System.out.println("Hola Mundo!");
   }
}
```

Aun cuando sólo muestra en pantalla la frase `Hola Mundo!`, podemos ver que el

[1] Java 2 Entrerprise Edition.

código introduce conceptos de programación orientada a objetos (OOP[2]). Para entender completamente este programa, el estudiante está enfrentado con clases, métodos estáticos y arreglos. Tratar de explicar a un estudiante principiante conceptos de OOP, prematuramente, es un reto fuera de lugar que puede ocasionarle confusión. Los cursos introductorios universitarios por lo general tienen el objetivo de la resolución algorítmica y la creación de programas procedimentales. En otros lenguajes de programación, como Perl, el anterior programa puede ser escrito en una sola línea de código como este:

```
print "Hola Mundo!";
```

Otro problema en Java lo constituyen los entornos de desarrollo de programas. La controversia se presenta cuando el docente debe escoger la herramienta. Existen dos opciones: usar un entorno popular en la industria o simplemente usar un editor y realizar la compilación y depuración manualmente. Lamentablemente los entornos del mundo real no fueron diseñados para principiantes, porque poseen una serie de opciones que son incomprensibles para estudiantes de primer año. Por este motivo se deben dedicar varias sesiones de clase solamente para enseñar el manejo adecuado del entorno.

Java es un lenguaje inherentemente orientado a objetos que hace difícil este proceso de enseñanza. Afortunadamente, la comunidad académica nos provee de estrategias para salir a flote de este problema a través de herramientas especiales de enseñanza, las cuales examinaremos en las próximas secciones de este artículo.

Tendencias actuales

La tendencia general favorece a Java desde sus inicios. Aunque es difícil encontrar estadísticas sobre Java en el mundo académico, existen sondeos que nos muestran que Java sigue como lenguaje de programación favorito entre los estudiantes de primer año. Los números son diferentes entre regiones y a veces incluso presentan datos sorprendentes como es el caso de Australia. Entre los años 1971 y 1997 el 92 % de las universidades australianas enseñaba Pascal como primer lenguaje [1]. Desde 1997 hubo un cambio radical. Actualmente el lenguaje más popular entre las universidades australianas es Java [1].

En Estados Unidos un sondeo revela que Java es también el lenguaje favorito con una proyección del 56 % de aceptación para el anō académico 2002-2003.

El cuadro 1 claramente muestra que Java continúa en ascenso en el mundo académico y que C++ es el lenguaje más constante en los programas académicos estadounidenses. Por otro lado, el reporte [2] también indica que se considera el paradigma orientado a objetos como primordial en 82 % de los programas académicos y el 31 % considera que el paradigma procedimental es el primordial (algunas universidades consideran ambos como primordiales).

Otro sondeo realizado por *The Gartner Group* [4] en enero del año 2000 en Estados Unidos muestra que Java era ofrecido en 87 % de las instituciones encuestadas y que además era obligatorio en 56 % de los programas en Ciencias de la Computación. Pero el avance de Java no se manifiesta solamente a nivel universitario sino también a nivel de enseñanza secundaria. En Estados Unidos existe una institución denominada *The College Board*[3] que se encarga de administrar exámenes tanto de admisión a universidades como de conocimientos en diferentes ramas. La prueba AP (*Advanced Placement*) en Ciencias de la Computación mide los conocimientos de pre-bachilleres para obtener crédito de conocimiento avanzado. Un avance reciente y favorable para Java es su incorporación de Java como lenguaje

[2]Object Oriented Programming.

[3]`www.collegeboard.com`

Primer Lenguaje	1995-96	1996-97	1997-98	1998-99	1999-00	2001-02	Proyección 2002-03
Ada	12	18	19	7	6	4	2
BASIC	-	-	-	1	-	-	
C	17	14	11	20	19	11	9
C++	32	39	47	50	54	40	40
Eiffel	2	-	2	1	-	-	
FORTRAN	2	-	-	-	-	-	
Java	-	-	9	22	22	49	56
Pascal	36	23	6	2	5	2	0
Scheme	2	4	4	1	-	-	

Cuadro 1: Lenguajes enseñados en cursos introductorios. Obtenido del Reporte Investigativo de los departamentos que ofrecen grados académicos en universidades de USA [2].

base para esa prueba, efectiva desde el año académico 2003-04 [4]. Esto significa que la demanda de aprendizaje de Java por jóvenes preuniversitarios, al menos en USA, se incrementará aún más en el futuro.

Herramientas para enseñanza de Java

Podríamos predicar sobre las buenas características de Java durante muchas horas. Sin embargo, estas cualidades hacen al lenguaje más complejo para ser explicado en una pizarra. Existen herramientas y librerías diseñadas específicamente para la instrucción de lenguajes de programación, no sólo de Java sino también de otros lenguajes como C/C++ o Pascal. Uno de los entornos interesantes para programadores principiantes es el conocido *Karel el Robot*, que ha sido portado exitosamente para la enseñanza de varios lenguajes de programación.

Karel el Robot

La idea original de Karel el Robot fue concebida por Richard Pattis quien diseñó esta herramienta para la enseñanza de programación en Pascal a principios de los años ochenta. Karel ha sido divulgado y utilizado ampliamente en la comunidad académica, especialmente en universidades estadounidenses. Es más, Karel fue portado para la instrucción de otros lenguajes de programación como ser C, C++ y recientemente Java. Algunas universidades, como es el caso de Stanford University, hicieron su propia versión de Karel para la enseñanza de C a programadores principiantes. Otro ejemplo de Karel para la instrucción de otros lenguajes es *Karel++* [5], el cual fue creado por Joseph Bergin y sirve para la introducción de programación orientada a objetos en C++. Las diferentes versiones de Karel vienen con buena documentación y hasta con libros texto. La idea fundamental es proporcionar un entorno gráfico donde el estudiante pueda manipular un *robot virtual* a través de instrucciones en un lenguaje específico. Además, el robot puede interactuar con otros objetos y se mueve dentro de un arreglo de calles y avenidas que simulan una ciudad. Es decir, el estudiante principiante en programación puede ver gráficamente cómo un robot puede moverse, girar, recoger objetos, llevar objetos de un lugar a otro, etc. El anzuelo que usamos para obtener la atención de los estudiantes es, en gran parte, el entorno gráfico en el que se desenvuelven todas las acciones del robot.

Java no ha podido ser excluído del mundo de Karel, actualmente podemos encontrar diferentes sabores de Karel como ser *Karel J. Robot*[4] y *Karel the Robot* de Byron Weber [5]. Personalmente encuentro que la versión de Weber es muy estable y el autor provee un libro texto simple de entender,

[4] `http://csis.pace.edu/~bergin/KarelJava2ed/Karel++JavaEdition.html`

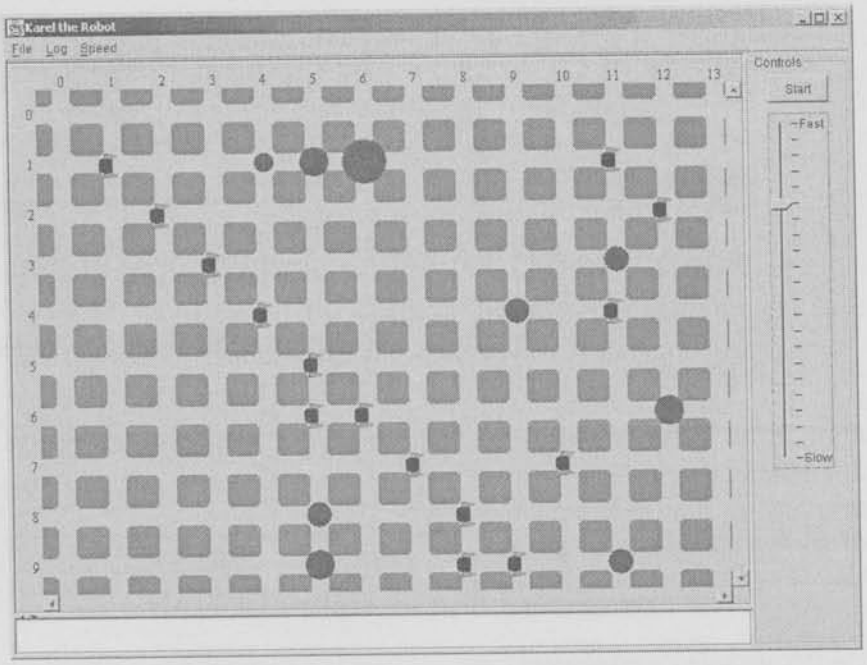

Figura 1: Interfaz de Karel el Robot (versión Java) donde se muestra varios robots que recogen rocas representadas por los círculos. La imagen pertenece a la ejecución de un proyecto realizado por los estudiantes Daniel Álvarez, Reynaldo Arcani, Marco Arce, Ariel Argandoña y Diego Collarana.

aunque sólo está disponible en Inglés y falta mejorar algunos de sus capítulos. El Karel de Weber para Java ofrece un paquete de clases, que puede ser fácilmente configurado usando cualquier ambiente de desarrollo de programas como ser *JBuilder* o *Forte for Java*[5]. Su entorno gráfico simple y dinámico hace que Karel sea atractivo.

Mis experiencias personales con Karel fueron muy satisfactorias. Actualmente, uso Karel para la instrucción de Java en una de las materias del programa de Ingeniería de Sistemas de la Universidad Católica Boliviana - Regional Cochabamba. Para poder usar Karel y su documentación se ha obtenido el permiso explícito del creador de Karel (versión Java). Se ha podido percibir un entusiasmo inmediato de los estudiantes con respecto a su manejo, pero ese entusiasmo puede ser apagado si se decide usar Karel para varias de las tareas asignadas en el semestre porque lo vuelve monótono. Un inconveniente que se tuvo que superar es el hecho de que el libro texto de Karel es incompleto y sólo disponible en Inglés, lo cual no permite la lectura fluida del estudiante, preocupado por traducir algunas palabras al castellano. Aparte de lo mencionado anteriormente, la librería es estable y no se ha encontrado ningún problema de compilación o ejecución durante los dos semestres consecutivos durante los cuales se usó Karel. Todos los problemas que los estudiantes encontraron con Karel eran relacionados a su propio código y no a Karel. Con respecto a las tareas que se pueden asignar con Karel podríamos decir que existe una gran variedad dependiendo del nivel requerido. En el caso específico de la materia dictada por el

[5]JBuilder es un producto de Borland Corp y Forte for Java es un producto de Sun Microsystems.

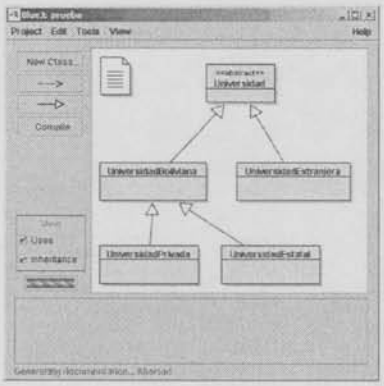

Figura 2: Interfaz gráfica de BlueJ para la estructuración de un programa en Java compuesto por clases, interfaces, y clases abstractas.

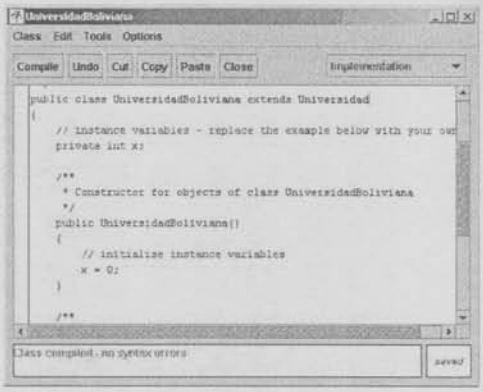

Figura 3: Imagen del Editor de código fuente Java usado en BlueJ.

autor, la respuesta de los estudiantes frente a proyectos con dificultad elevada fue, en promedio, bastante satisfactoria. Karel abrió la puerta de la imaginación a muchos estudiantes que presentaron soluciones muy elegantes. Muchos conceptos básicos de programación orientada a objetos, como ser herencia y polimorfismo son fácilmente explicados con Karel. La Figura 1 muestra una imagen congelada de la interfaz gráfica de Karel con varios robots realizando una tarea colaborativa.

BlueJ

Otra herramienta muy difundida para la enseñanza de Java como primer lenguaje es BlueJ[6], desarrollada como un proyecto de investigación en la universidad australiana Monash University. Actualmente, BlueJ es un proyecto patrocinado por la compañía creadora de Java, Sun Microsystems. BlueJ es constantemente actualizado para funcionar con las últimas versiones de Java. Comparado con Karel, BlueJ ofrece varias otras opciones interesantes a considerar, como ser ejecución interactiva de instrucciones, diseño de clases en forma gráfica, además de proporcionar editor, compilador y depurador propios. En cuanto a la documentación, el equipo encargado del desarrollo de BlueJ espera publicar un libro texto para el aprendizaje de Java OO[7] usando BlueJ.

La herramienta facilita la implementación de programas usando una interfaz gráfica como se puede apreciar en la Figura 2. BlueJ es bastante compacto y presenta al estudiante las herramientas estrictamente necesarias y a la vez fáciles de usar, para poder crear programas en Java. A menudo sucede que un estudiante principiante tiene dificultades en manejar los entornos de desarrollo de programas como JBuilder por cuanto éstos presentan una serie de opciones y menús entendibles sólo por profesionales en la industria, pero no por programadores principiantes.

En realidad Karel y BlueJ van en direcciones diferentes pero con un mismo objetivo: hacer que el aprendizaje de Java sea lo más atrayente posible y que el estudiante se enfoque a entender conceptos de programación sin distraerse con detalles específicos de un entorno de desarrollo.

[6]Sitio oficial: `www.bluej.org`

[7]Orientado a Objetos.

Las dos herramientas mencionadas en este artículo son sólo parte de una gama heterogénea y exquisita. Universidades prestigiosas en el ámbito de Ciencias de la Computación han desarrollado su propia herramienta para la enseñanza de Java. MiniJava [3], desarrollado en Stanford University, es un claro ejemplo de los esfuerzos por la investigación en el campo de la enseñanza de programación en Java.

Conclusiones

Hemos visto que el gran peso de Java en la industria tuvo una influencia inmediata en instituciones académicas donde se vieron forzados , al menos parcialmente, a adoptar este lenguaje requerido como primer lenguaje. La tendencia general de las universidades, principalmente extranjeras, con respecto a Java es demostrado en el cambio de sus programas curriculares de pregrado.

Adicionalmente, enseñar Java es un reto para los docentes, porque los estudiantes deben digerir paralelamente muchos conceptos. De cualquier manera este problema puede ser superado usando herramientas instruccionales como ser Karel, BlueJ y otros más. En el futuro se observa a Java como un lenguaje que proporcionará muchos más cambios empezando, desde la secundaria, pero no como un lenguaje que reemplace a lenguajes sólidos como C++, sino como un lenguaje para enseñar OOP antes que programación procedimental.

Agradecimientos

A los estudiantes Daniel Álvarez, Reynaldo Arcani, Marco Arce, Ariel Argandoña y Diego Collarana por realizar mejoras a su proyecto con Karel, el cual fue mencionado en este artículo. Quiero también agradecer a Javier Rojas Balderrama por su colaboración con LaTeX.

Referencias

[1] Michael de Raadt, Richard Watson, y Mark Toleman. Language trends in introductory programming courses. Reporte técnico, Informing Science - InSite, University of Suthern Queensland, Australia, June, 2002.

[2] Renée McCauley y Bill Manaris. Comprehensive Report on th 2001 Survey of Departments Offering CAC-Accredited Degree Programs. Reporte Técnico CoC/CS TR# 2002-9-1, Department of Computer Science, College of Charleston, Charleston, SC 29424, USA, May, 2002.

[3] Eric Roberts. An overview of MiniJava. En *Proceedings of the thirty second SIGCSE technical symposium on Computer Science Education*, pp 1–5, Charlotte, North Carolina, United States, 2001. ACM Press.

[4] Bruce Stewart. Java as a Teaching Language, Enero, 2001. Artículo localizado en `http://java.oreilly.com`.

[5] Byron Weber. *Karel the Robot, Learning to Program in Java.*