AEIOU: Una herramienta de apoyo en la enseñanza de la programación orientada a objetos

Guillermo Licea, Luis G. Martínez, Leocundo Aguilar, Reyes Juárez-Ramírez
Ingeniería en Computación, Universidad Autónoma de Baja California
Tijuana, Baja California, México

RESUMEN

AEIOU (Ambiente para la Enseñanza Integral de Objetos en Universidades) es una herramienta de programación Java para apoyar las prácticas y tareas en la secuencia de cursos de programación orientada a objetos en la carrera de Ingeniero en Computación de la Universidad Autónoma de Baja California (UABC). AEIOU permite a los estudiantes escribir, compilar, corregir y ejecutar de manera sencilla programas simples (compuestos por una clase) y de mediana complejidad (compuestos por varias clases), apoyándose en las características del entorno de programación y en marcos de clases especializados que se integran al entorno.

Palabras clave: Programación orientada a objetos, Ambiente de programación.

1. INTRODUCCIÓN

Uno de los elementos mas importantes en los planes de estudio de informática es la programación de computadoras, lo cual sigue siendo un arte y a la vez una ciencia [20]. El aprendizaje cualitativo de los lenguajes de programación permite mejorar la calidad de los productos de software y apoyar en el desarrollo de la ingeniera de software como profesión [19]. Sin embargo, la programación es una habilidad, y enseñar habilidades es mas difícil que enseñar materias como física, cálculo o química [9], considerando además, que la motivación de los alumnos que toman cursos de programación ha cambiado, los alumnos tienen varias motivaciones hot en día y no solo el aprendizaje [14].

En las últimas dos décadas se ha discutido bastante sobre el mejor enfoque para enseñar a programar. Algunos autores como Hu [12], Burton y Bruhn [4], Jacquot [13] y Tuttle [22] se inclinan por un enfoque "Primero imperativo, después orientado a objetos", mientras que otros autores como Blumenstain [3], Duke et al [6], Hadjerrouit [10], Clark y MacNish [5] prefieren el enfoque "Orientado a objetos desde el inicio".

Con la finalidad de facilitar las labores de los profesores de programación, varios autores se han dado a la tarea de desarrollar herramientas que apoyen a los alumnos para el entendimiento de conceptos y para el desarrollo de los programas asignados como prácticas o tareas dentro de sus cursos. García et al [7] desarrollaron una herramienta para apoyar la construcción de prototipos orientados a objetos, Allen et al [2] desarrollaron DrJava, un ambiente de programación educacional que permite a los alumnos concentrarse en el diseño del programa. Hsia et al [11] utilizaron DrJava para definir subconjuntos de Java que introducen a los alumnos de manera gradual en el lenguaje. Gray y Flatt [8] propusieron ProfessorJ para simplificar la interfaz con el compilador de Java y la máquina virtual. Kölling et al [16] desarrollaron BlueJ, una de las herramientas mas populares para la enseñanza del paradigma orientado a objetos con Java. Roberts [21] desarrolló MiniJava, un lenguaje orientado a la enseñanza para alumnos novatos en programación.

Otros autores como Aispuro et al [1], Koffman y Woiz [15], Lambert y Osborne [17], proponen herramientas fáciles de utilizar que ayudan a los alumnos novatos a escribir programas con interacción gráfica.

En este artículo se presenta AEIOU, una herramienta desarrollada para apoyar a profesores y alumnos en cursos de programación orientada a objetos, específicamente con el lenguaje de programación Java. AEIOU proporciona una plataforma de experimentación para extender y adaptar la herramienta según las necesidades de los profesores y alumnos, y acorde a las modificaciones al plan de estudios (cantidad de cursos de programación, secuencia de los cursos, etc.).

En la sección 2 se describe AEIOU y sus características, en la sección 3 se presentan los resultados obtenidas con la utilización de AEIOU en universidades, en la sección 4 se presentan las conclusiones y en la sección 5 se presenta el trabajo futuro.

2. DESCRIPCIÓN DE AEIOU

AEIOU es un ambiente de desarrollo Java diseñado para apoyar a los profesores en la enseñanza y facilitar a los alumnos el desarrollo de programas en cursos de programación orientada a objetos o similares. AEIOU contempla tres tipos de alumnos: principiantes, intermedios y avanzados, para lo cual se incluyen tres niveles del ambiente de desarrollo.

AEIOU para principiantes

AEIOU para alumnos principiantes (AEIOU-1) permite aplicar los conceptos básicos no orientados a objetos del lenguaje Java. Con AEIOU-1 se pueden escribir, compilar y ejecutar programas sencillos compuestos por una clase.

AEIOU ofrece dos vistas: una con la representación gráfica de las clases que componen el programa y otra con una vista específica de cada clase a través de la cual se puede modificar el código de la clase, compilar y ejecutar. En caso de encontrarse errores durante la compilación, estos se muestran en inglés y se agrega una explicación en español.

La Figura 1 muestra la vista del proyecto, la Figura 2 muestra la vista del código de la clase principal del programa insertando un error en el código y la Figura 3 muestra la vista del código sin errores.

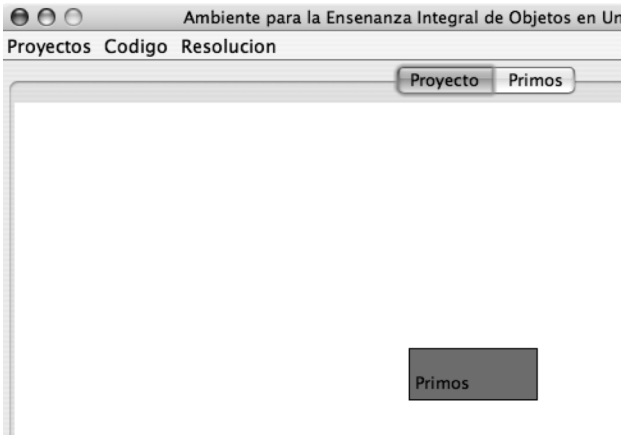

Figura 1. Vista del proyecto Primos.

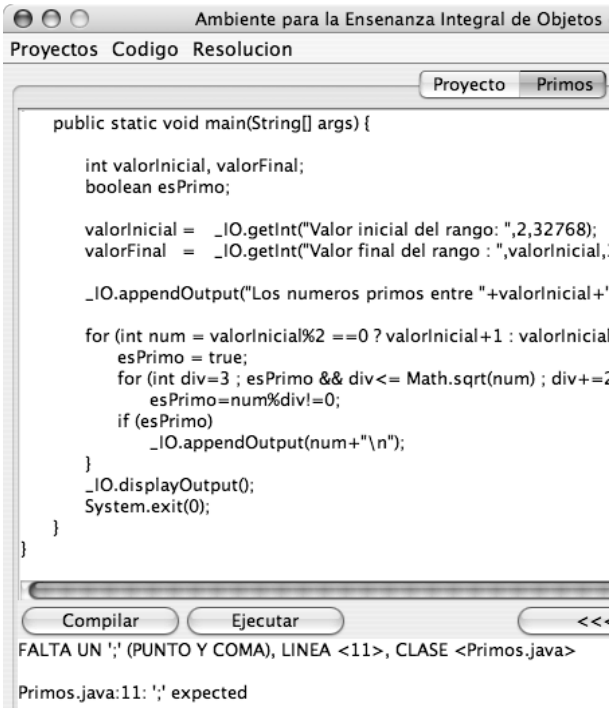

Figura 2. Vista del código con un error.

Para facilitar el desarrollo de programas interactivos, sin involucrar los paquetes gráficos de Java (AWT y Swing), AEIOU-1 proporciona una clase llamada IO (por Input/Output), la cual fue adaptada de la propuesta de Koffman y Woiz [15], que permite a los alumnos incluir en sus programas lectura y escritura de datos utilizando cajas de diálogo simples. La Figura 4 muestra los diálogos desplegados durante la ejecución del programa para encontrar números primos.

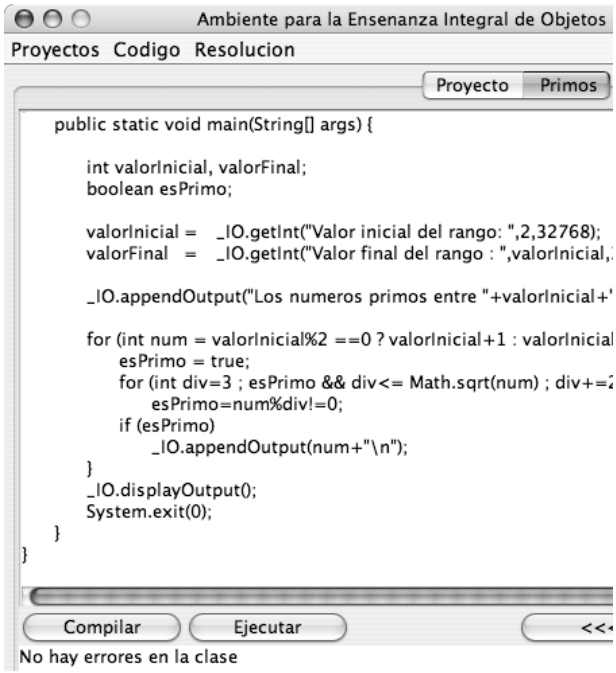

Figura 3. Vista del código sin errores.

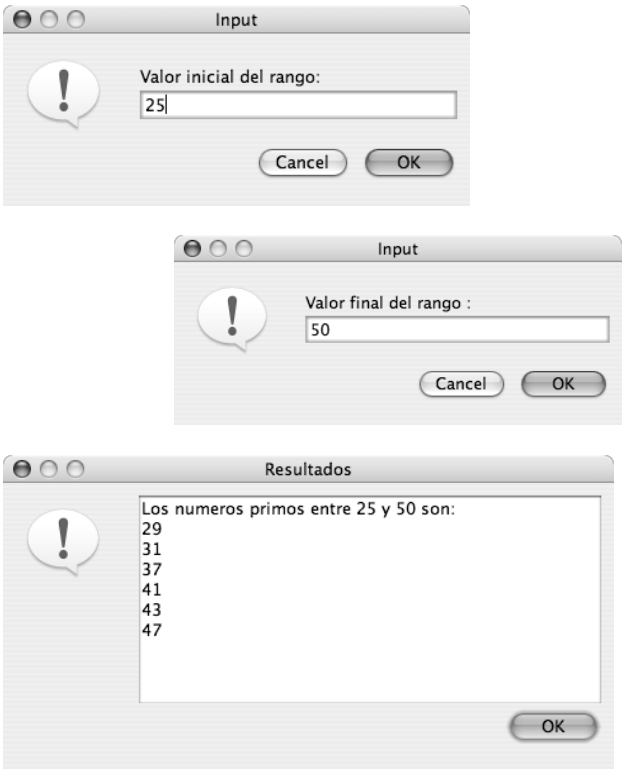

Figura 4. Ejecución del programa Primos.

AEIOU para intermedios

AEIOU para alumnos intermedios (AEIOU-2) incluye las características generales de AEIOU-1 y además permite aplicar los conceptos fundamentales de la programación orientada a objetos, a través del desarrollo de programas que incluyen la definición de jerarquías de clases. AEIOU-2 genera el código básico de una clase típica, incluyendo campos, constructores, métodos "get", "set", "equals" y "toString"; así como métodos "input" y "output" para generar diálogos para la lectura y escritura

de datos. La Figura 5 muestra la vista de proyecto y el diálogo utilizado para agregar una nueva clase al programa.

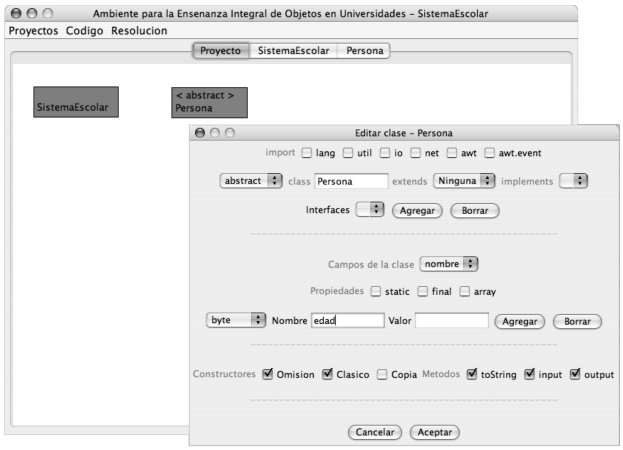

Figura 5. Diálogo para agregar una clase.

La Figura 6 muestra la vista del proyecto con una jerarquía de clases basada en la clase *Persona*, la cual es definida por el alumno.

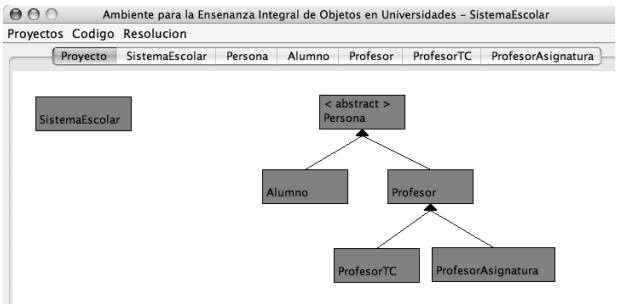

Figura 6. Vista del proyecto con una jerarquía de clases.

La Figura 7 muestra la vista del código correspondiente a la clase *Profesor* definida en la jerarquía de clases de la figura 6. Se puede observar que la sección para edición de código se encuentra dividida en dos partes. La parte superior muestra el código generado por AEIOU-2 y la parte inferior muestra el código escrito por el alumno.

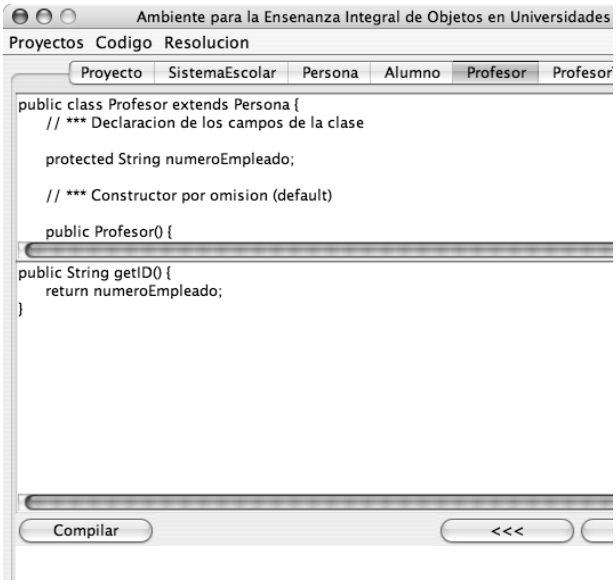

Figura 7. Vista del código de la clase Profesor.

La clase principal del programa es *SistemaEscolar,* la Figura 8 muestra la vista del código de esta clase y su ejecución.

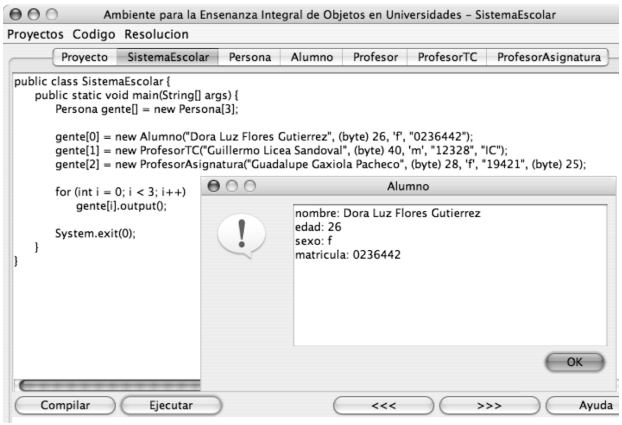

Figura 8. Vista del código de la clase SistemaEscolar y su ejecución.

AEIOU para avanzados

AEIOU para alumnos avanzados (AEIOU-3), como su versión previa, apoya el desarrollo de programas utilizando jerarquías de clases, pero agrega una nueva característica, AEIOU-3 permite a los alumnos definir, utilizar y reutilizar patrones de diseño.

La Figura 9 muestra el patrón observador (Observer) insertado en la vista de proyecto, la Figura 10 muestra la inserción de la interfaz *DisplayElement* para iniciar la adaptación del patrón observador y la Figura 11 muestra la vista del proyecto con las clases e interfaces modificadas para adaptar el patrón observador al problema particular que se está solucionando.

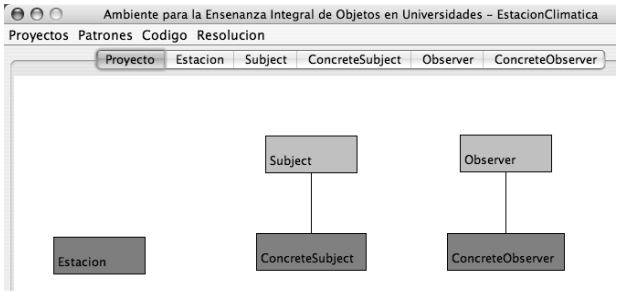

Figura 9. Inserción del patrón observador.

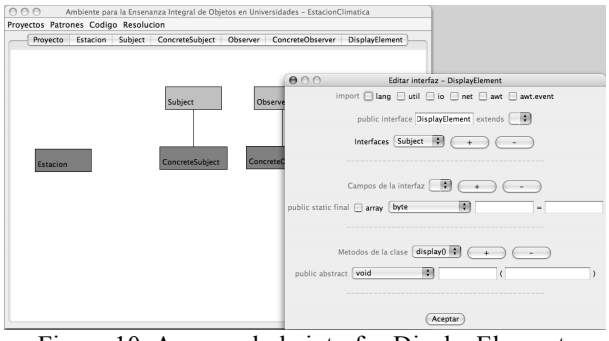

Figura 10. Agregando la interfaz DisplayElement.

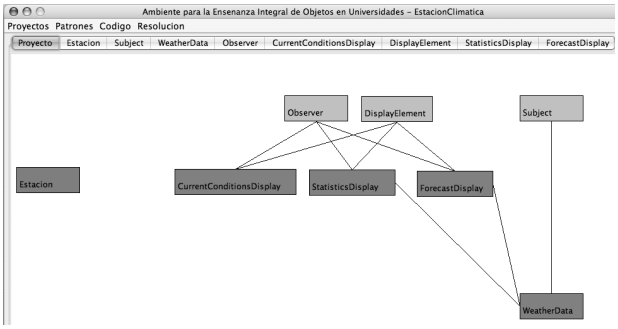

Figura 11. Adaptación del patrón observador.

La figura 12 muestra la vista del código de la clase *WeatherData*, clase que sustituye a *ConcreteSubject* en la adaptación del patrón observador y la Figura 13 muestra la vista del código de la clase Estación (la clase principal) y su ejecución.

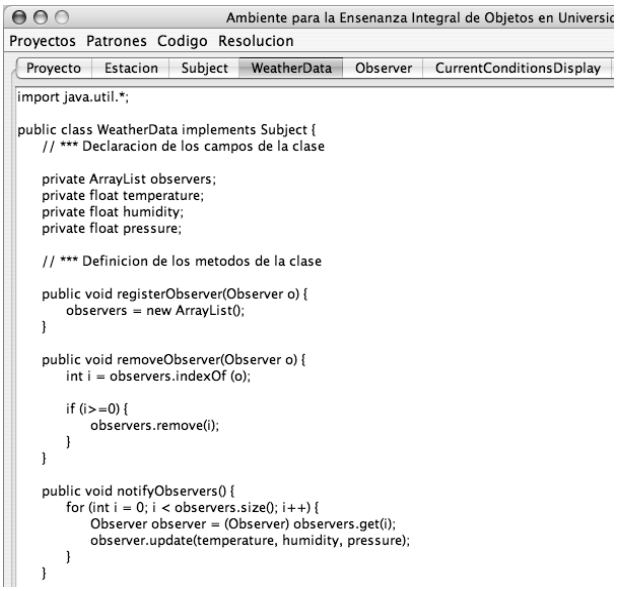

Figura 12. Vista del código de la clase WeatherData.

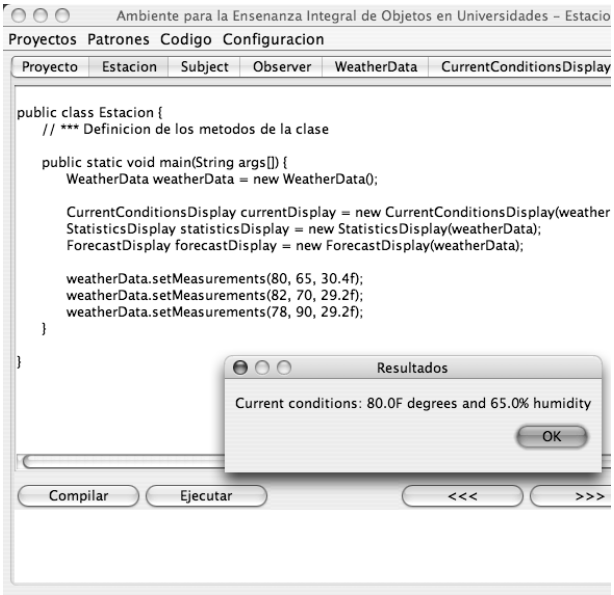

Figura 13. Vista del código de la clase Estación y su ejecución.

3. RESULTADOS

AEIOU fue diseñado para apoyar la enseñanza de los conceptos y facilitar la realización de las tareas consideradas en los 2 cursos que conforman la secuencia de cursos de programación orientada a objetos.

En la primera parte del primer curso se utiliza AEIOU-1 para apoyar la enseñanza de los conceptos básicos no orientados a objetos de Java como tipos de datos, variables, estructuras de control, etc., a través de la elaboración de programas simples.

En la segunda parte del primer curso y en todo el segundo curso se utiliza AEIOU-2 para apoyar la enseñanza de los conceptos básicos de la POO y su sintaxis en Java como clases, objetos, herencia, interfaces, jerarquías de clases, polimorfismo, etc., a través de la elaboración de programas de mediana complejidad constituidos por varias clases organizadas jerárquicamente.

En el segundo curso también se utiliza AEIOU-3 para apoyar la maduración de conceptos de la POO a través de la elaboración de programas mas complejos que involucran jerarquías de clases y patrones de diseño, así como marcos de clases especializados para aplicaciones en red y generación de la interfaz de usuario, elaborados por los mismos estudiantes.

AEIOU ha sido utilizado en la Universidad de Sonora y la Universidad Autónoma de Baja California desde 2005 para apoyar una metodología de enseñanza [18] que incentiva a los estudiantes a alcanzar un entendimiento mas profundo de los conceptos de la POO a través de la secuencia de cursos de programación.

4. CONCLUSIONES

La experiencia en el uso de AEIOU indica que el ambiente es aceptado por los estudiantes, principalmente por los principiantes, quienes se benefician al utilizar un ambiente de programación sencillo que les permite escribir programas pequeños rapidamente, compilarlos y revisar los mensajes de error en su idioma.

Los estudiantes intermedios y avanzados aprovechan la característica de AEIOU-2 y AEIOU-3 que genera el código de las clases (o la mayor parte de él) que utilizan en sus programas, esto les ahorra tiempo y esfuerzo, el cual emplean en aumentar la complejidad de la funcionalidad de sus programas.

Los estudiantes avanzados también aprovechan el catálogo de patrones que está contenido en AEIOU-3 para agilizar la elaboración de sus programas y mejorar el diseño y mantenimento de los mismos.

5. TRABAJO FUTURO

Por ser un ambiente de programación desarrollado en casa, se tiene la posibilidad de modificarlo o adaptarlo segun las necesidades o intereses de los profesores que participan en su desarrollo. Actualmente se tienen las siguientes propuestas: registrar el tiempo que cada alumno tarda en completar algún programa asignado, registrar los errores cometidos por cada alumno durante la elaboración de un programa, ofrecer tutoriales en línea a cada alumno según sus necesidades, ofrecer la posibilidad de colaborar con otros alumnos para la elaboración de los programas asignados como tarea, entre otras.

6. REFERENCIAS

[1] E. E. Aispuro, G. Licea, J. Suárez, M. A. Carreño, I. Estrada, R. Juárez-Ramírez, L. Aguilar, L. G. Martínez, "Supporting the development of interactive applications in introductory programming courses", Computer Applications in Engineering Education, Wiley interscience (Publicado en internet), 2009.

[2] E. Allen, R. Cartwright, B. Stoler, DrJava: A lightweight pedagogic environment for Java, 33rd ACM SIGCSE Technical Symposium on Computer Science Education, pp. 137-141, Covington, Kentucky, United States of America, 2002.

[3] M. Blumenstain, Strategies for improving a Java-based, first year programming course, International Conference on Computers in Education, pp. 1095-1099, Auckland, New Zealand, 2002.

[4] P. J. Burton, R. E. Bruhn, "Teaching programming in the OOP era", ACM SIGCSE Bulletin Vol. 35, pp. 111-114, 2003.

[5] D. Clark, C. MacNish, Java as a teaching language – opportunities, pitfalls and Solutions, Australasian Conference on Computer Science Education, pp. 173-179, Queensland, Australia, 1998.

[6] R. Duke, E. Salzman, J. Burmeister, J. Poon, L. Murray, Teaching programming to beginners–Choosing the language is just the first step, Australasian Conference on Computer Science Education, pp. 79-86, Melbourne, Australia, 2000.

[7] J. B. García, F. Ortín, E. García, M. Pérez, "Towards an object-oriented programming system for education", Computer Applications in Engineering Education Vol. 14, pp. 243-255, 2006.

[8] K. E. Gray, M. Flatt, ProfessorJ: A gradual introduction to Java through language levels, OOPSLA '03, pp. 170-177, Anaheim, California, United States of America, 2003.

[9] D. Gries, "Where is programming methodology these days", ACM SIGCSE Bulletin Vol. 34, pp. 5-7, 2002.

[10] S. Hadjerrouit, "Java as first programming language: A critical evaluation", ACM SIGCSE Bulletin Vol. 30, pp. 43-47, 1998.

[11] J. Hsia, E. Simpson, D. Smith, R. Cartwright, Taming Java for the classroom, 36th ACM SIGCSE Technical Symposium on Computer Science Education, pp. 327-331, St. Louis, Missouri, United States of America, 2005.

[12] C. Hu, "Rethinking of teaching objects-first", Education and Information Technology Vol. 9, pp. 209-218, 2004.

[13] J. P. Jacquot, Which use for introductory courses ?, International Conference on Principles and Practice of Programming in Java, pp. 119-124, Dublin, Ireland, 2002.

[14] T. Jenkins, The motivation of students of programming, 6th Conference on Innovation and Technology in Computer Science Education, pp. 53-56, Canterbury, United Kingdom, 2001.

[15] E. Koffmann, U. Woiz, "A simple Java package for GUI-like interactivity", ACM SIGCSE Bulletin Vol. 33, pp. 11-15, 2001.

[16] M. Kölling, B. Quig, A. Patterson, J. Rosenberg, "The BlueJ system and its pedagogy", Journal of Computer Science Education Vol. 13, pp. 1-12, 2003.

[17] K. Lambert, M. Osborne, "Easy, realistic GUIs with Java in CS1", Journal of Computing Sciences in Colleges Vol. 16, pp. 211-217, 2001.

[18] G. Licea, R. Juarez-Ramírez, L. G. Martinez, L. Aguilar, "Developing programming tools to reach a deeper understanding of advanced programming Concepts", Computer Applications in Engineering Education Vol. 18, pp. 305-314, 2008.

[19] P. Maheshwari, Teaching programming paradigms and languages for qualitative learning, 2nd Australasian Conference on Computer Science Education, pp. 32-39, Melbourne, Australia, 1997.

[20] M. A. Malik, "On the perils of programming", Communications of the ACM Vol. 43, pp. 95–97, 2000.

[21] E. Roberts, An overview of MiniJava, 32nd ACM SIGCSE Technical Symposium on Computer Science Education, pp. 1-5, Charlotte, North Carolina, United States of America, 2001.

[22] S. M. Tuttle, ¡Yo quiero Java! Teaching Java as a second programming language, Journal of Computing Sciences in Colleges Vol. 17, pp. 34-45, 2001.