ISSN 1870-4069

Hacia una herramienta web para visualizar la ejecución de programas escritos en el lenguaje Java

Miguel Castillo-Cortes, Carlos R. Jaimez-González

Universidad Autónoma Metropolitana, Unidad Cuajimalpa,
Departamento de Tecnologías de la Información, México

210368127@alumnos.cua.uam.mx, cjaimez@correo.cua.uam.mx

Resumen. Este artículo presenta una propuesta de herramienta web de apoyo para el aprendizaje de la programación a nivel universitario, la cual permitirá visualizar una animación que representa la ejecución de programas escritos en el lenguaje de programación Java. La herramienta propuesta apoyará en la comprensión de programas y entendimiento de los conceptos básicos de la programación, tales como creación de variables, asignación de valores a variables, uso de estructuras de control de flujo de programa y llamadas a funciones con paso de parámetros. En este artículo también se presenta un análisis comparativo de herramientas con funcionalidad similar a la propuesta.

Palabras clave: Tecnología educativa, computación en educación, comprensión de software, visualización de software.

Towards a Web Tool to Visualize the Execution of Programs Written in the Java Programming Language

Abstract. This paper presents a proposal of web tool to support the learning of programming at university level, which will allow to visualize an animation that represents the execution of programs written in the Java programming language. The proposed tool will help to understand programs and programming basic concepts, such as the creation of variables, setting values to variables, use of control structures and function calls with parameters. This paper also presents a comparative analysis of tools with similar functionality to the proposal.

Keywords. Educational technology, computer science in education, understanding software, software visualization.

1. Introducción

Entender un programa es tener la habilidad de comprender de qué forma el código fuente realiza la tarea para la cual fue creado. La comprensión de programas es un área de la Ingeniería de Software enfocada a elaborar técnicas y herramientas, basadas en procesos cognitivos y de ingeniería, que son utilizadas para tareas de reutilización, inspección, manutención, migración, extensión de software, entre otras aplicaciones; pero también se pueden emplear en áreas como la educación o la capacitación. Esto con el objetivo de lograr un mejor entendimiento de las aplicaciones informáticas [1].

Uno de los principales retos en el área de comprensión de programas es poder determinar la relación existente entre el dominio del problema y el dominio del programa [2]. El primero es el resultado de la ejecución del programa, por ejemplo, imprimir en pantalla el resultado de una operación; el segundo son todos los componentes del programa, que producen el comportamiento del sistema, por ejemplo, los métodos utilizados para poder resolver alguna operación [2]. Para poder representar estas relaciones, entre dominio del problema y el dominio del programa, se recurre al uso de la visualización de software (VS), la cual es una rama de la Ingeniería de Software cuyo objetivo es generar, a partir de ciertos aspectos del sistema, una o más representaciones multimedia, con la finalidad de facilitar el entendimiento del mismo [3,4].

En este artículo se presenta una propuesta de herramienta web destinada a facilitar el proceso de comprensión de programas escritos en el lenguaje de programación Java. Esta herramienta facilitará el aprendizaje debido a que fomentará la interconexión de conceptos básicos de la programación. De esta manera, se reforzarán los conocimientos del estudiante mediante representaciones visuales en el sistema.

El resto del artículo está organizado de la siguiente manera. La sección 2 presenta el marco teórico. La sección 3 describe una serie de herramientas similares a la propuesta y hace un análisis comparativo. En la sección 4 se presenta la propuesta de herramienta para visualizar la ejecución de programas escritos en el lenguaje Java. Finalmente, las conclusiones y el trabajo futuro se presentan en la sección 5.

2. Marco teórico

En el desarrollo de sistemas de comprensión de software es importante apoyarse de otros campos de estudio, tales como psicología cognitiva, la psicopedagogía, la visualización de programas, entre otros. En los siguientes apartados se abordan los temas de modelos cognitivos y visualización de software, los cuales son muy relevantes para la propuesta que se presenta en este artículo.

2.1. Modelos cognitivos

El desarrollo cognitivo es un campo de investigación de la psicología cognitiva que estudia los procesos mentales involucrados en la creación de conocimiento. Aprender es combinar el conocimiento ya adquirido con nuevos conceptos mediante procesos de aprendizaje; por ejemplo, el modelo *Bottom Up* que parte de conceptos específicos

obtenidos de la lectura del programa para posteriormente generar abstracciones generales; o el modelo *Top Down*, el cual supone que ya se tiene noción de la funcionalidad del programa para proponer una hipótesis, que posteriormente se verificará al revisar el código fuente [5,6].

Los modelos cognitivos están conformados por tres elementos base: el conocimiento, el proceso de asimilación y un modelo mental. El primero plantea dos distinciones: el conocimiento interno, que se refiere a los conocimientos que ya se poseen; y el conocimiento externo, que son los nuevos conceptos que proveerá el sistema. El segundo elemento lo conforman las estrategias de aprendizaje, tales como *Bottom Up* o *Top Down*. Finalmente, el tercero es una representación que se le da al sistema, con la cual se intenta explicar el comportamiento del mismo [5,6].

2.2. Visualización de software

La visualización de programas es una rama de la Ingeniería de Software relacionada con la representación visual de la información, cuyo objetivo es generar, a partir de ciertos aspectos del sistema, una o más vistas que facilitarán el entendimiento del mismo. Una vista es una forma de representar los elementos de un sistema y las relaciones entre estos elementos [5].

Las representaciones multimedia orientadas a la comprensión de software deben aportar tres vistas básicas: la salida del sistema, los elementos del programa que generan dicha salida, y la relación entre las dos anteriores. Éstas permiten elaborar representaciones que asocian el dominio del problema con el dominio del programa; que facilitan relacionar los conocimientos que se poseen y los conceptos usados por el sistema [4].

Dichas representaciones no son fáciles de construir, dado que están fuertemente basadas en factores cognitivos y de ingeniería, por lo que se vuelve importante apoyarse de varias áreas del conocimiento como lo son el diseño gráfico, la psicología cognitiva, la psicopedagogía, la computación y otras disciplinas relacionadas con la creación de efectos multimedia y el aprendizaje [3].

Los aspectos del software necesarios para elaborar una representación se obtienen usando técnicas de extracción de la información que se clasifican por el tipo de información que extraen, la cual puede ser estática o dinámica [2]. Las técnicas de extracción de la información estática son utilizadas para verificar que no existan errores sintácticos, lexicográficos o incluso semánticos, haciendo uso de herramientas de análisis sintáctico para examinar el código fuente [2]. Las técnicas de extracción de la información dinámica se encargan de obtener información sobre el comportamiento del programa y sus componentes (métodos, variables, invocaciones, etc.) en tiempo de ejecución. Una de las técnicas para obtener estos datos, es el uso de la instrumentación de código (IC), la cual consiste en colocar instrucciones en el código fuente con el fin de monitorear su comportamiento durante la ejecución [2].

La visualización de software incluye dos áreas fundamentales: la visualización de programas y la visualización de algoritmos, dependiendo de lo que se desea comprender. La visualización de programas permite tener vistas del código fuente y su estructura. Una representación estática es de utilidad para verificar la validez del código

fuente, usualmente se utilizan editores de código para dar una mejor presentación y legibilidad del código fuente mediante indentado, diferencias de colores entre palabras reservadas y otros identificadores. La representación dinámica muestra información del programa en tiempo de ejecución, por ejemplo, resaltando las instrucciones del código cuando éstas están siendo ejecutadas [7]. La visualización de algoritmos se enfoca en representar la semántica del programa, esto significa que genera vistas del comportamiento del programa en ejecución sin mostrar las instrucciones y operaciones que hacen posible dicho comportamiento [7]. Estos conceptos influyen en el desarrollo de herramientas de comprensión de software.

3. Estado del arte

En esta sección se hace una revisión del estado del arte de herramientas para la comprensión de programas. También se proporciona un análisis comparativo al final de la sección.

3.1. BlueJ

BlueJ [8] es un entorno de desarrollo diseñado para aprender programación orientada a objetos con el lenguaje Java. Utiliza técnicas de visualización e interacción para mejorar la experiencia del usuario. Es un entorno en constante desarrollo, el cual fue creado por la Universidad de Kent en Canterbury, Reino Unido, y la Universidad de La Trobe, en Melbourne. BlueJ colorea el fondo de cada bloque de código para identificar visualmente las secciones del programa, ayuda en la detección de llaves fuera de lugar y muestra detalles de los objetos instanciados.

3.2. Jeliot 3

Jeliot 3 [9] es una herramienta de visualización de software que muestra cómo se interpreta un programa en lenguaje Java; las llamadas a métodos, variables y operaciones se visualizan mediante una representación multimedia que permite seguir paso a paso el proceso de ejecución del programa. En la Figura 1 se observa una captura de pantalla de Jeliot3, donde se muestran 3 paneles: el panel superior izquierdo se utiliza para ingresar el código que será ejecutado; el panel inferior izquierdo muestra la salida de la ejecución del programa; mientras que el panel de la derecha visualiza la representación del programa en ejecución, donde se tienen cuatro áreas, que identifican la inicialización de variables, la representación de operaciones y la de arreglos.

3.3. jGrasp

jGrasp [10] es un entorno de desarrollo que genera de forma automática visualizaciones animadas para mejorar la comprensión del software. jGrasp es capaz de generar diagramas de estructuras de control para los lenguajes Java, C, C++, entre otros. También crea diagramas UML. El visualizador representa estructuras de datos como

pilas, colas, listas ligadas, etc. En la Figura 2 se observa una captura de pantalla de jGrasp en la cual se muestra el área donde se escribe el código fuente, al ejecutar el código fuente muestra en el panel izquierdo inferior las instancias de los elementos creados en el programa como lo son las variables, arreglos, etc. En una ventana independiente se puede visualizar de forma gráfica la representación de estas instancias.

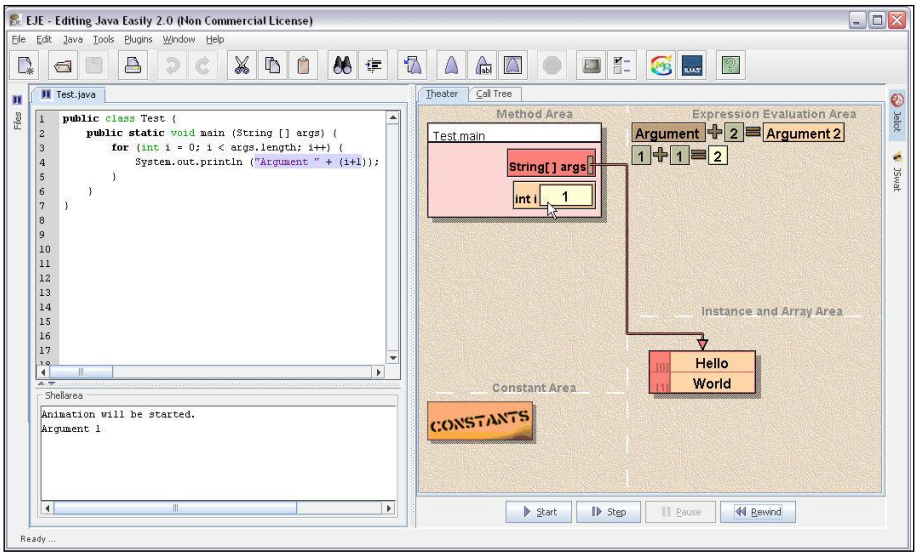

Fig. 1. Ejecución y representación de un segmento de código en Jeliot 3.

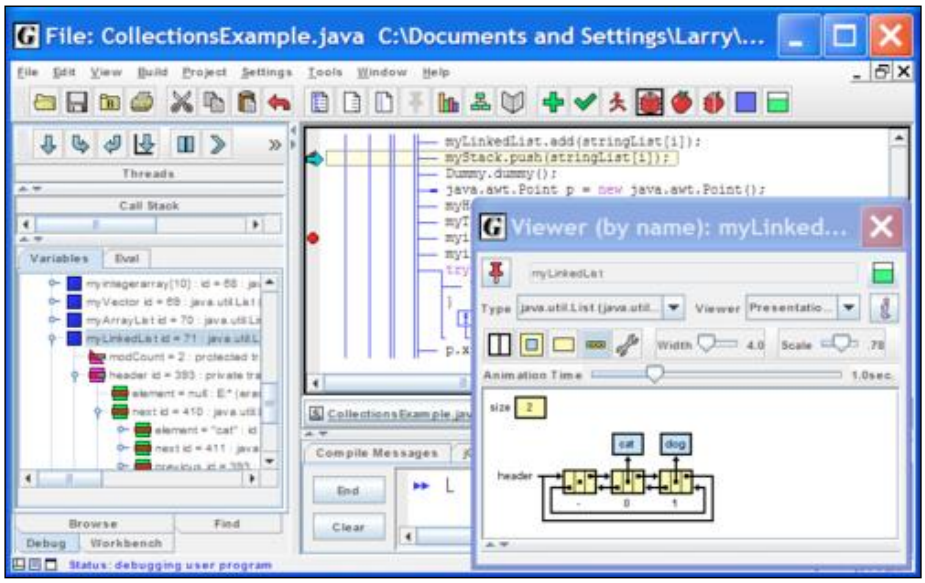

Fig. 2. Ejecución y representación de un segmento de código en jGrasp.

3.4. Scratch

Scratch [11] es una aplicación en la cual se pueden programar historias interactivas, juegos y animaciones, con tan solo definir reglas mediante bloques de instrucciones. Los proyectos desarrollados en esta aplicación pueden ser compartidos dentro de su catálogo de proyectos, de esta forma se tiene acceso a las aplicaciones creadas por terceros. En la Figura 3 se puede observar un programa realizado en la plataforma Scratch, que muestra una ventana con tres secciones, en la parte de la izquierda se tiene el área donde se visualizara la animación del programa y la parte central hay un conjunto de bloques de instrucciones que pueden ser usadas por el usuario; instrucciones tales como avanzar, girar, cambiar dirección, cambiar valor de una variable entre otras. En la tercera sección se encuentra el área donde el usuario arrastrara los bloques de instrucción que desea usar para su animación y los podrá acomodar como si de un rompecabezas de tratase. Para ejecutar la animación se tiene que dar clic sobre los bloques que se unieron.

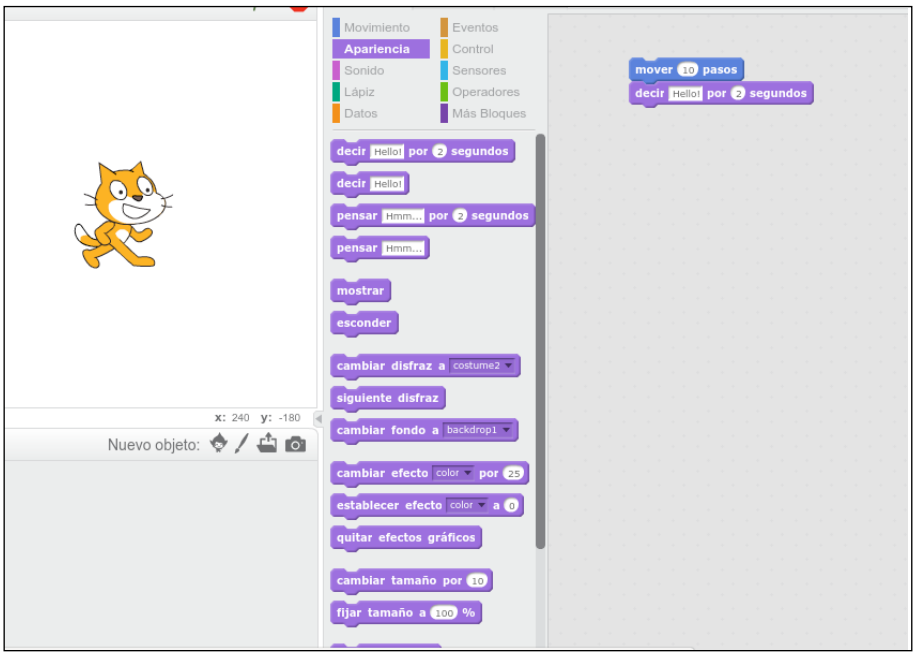

Fig. 3. Ejecución de un programa en la plataforma Scratch.

3.5. Otras herramientas

Existen otras herramientas para la comprensión de software que no se analizaron a profundidad, como las que se mostraron anteriormente. Esto fue debido a que estas herramientas no se consideraron tan relevantes como las que se revisaron detalladamente en las secciones anteriores. Algunas de éstas son las siguientes: Understand

[12], CodeSurfer [13], Imagix 4D [14], ShriMP [15], Alma [16], SeeSoft [17], Extravis [18], entre otras.

3.6. Comparación

En esta sección se presenta una tabla comparativa de características de las herramientas expuestas anteriormente y la herramienta propuesta. Las herramientas que se muestran en la Tabla 1 son las siguientes: H1) BlueJ, H2) Jeliot3, H3) jGrasp, H4) Scratch, H5) Herramienta web propuesta. El símbolo de verificación indica que la herramienta tiene la característica, mientras que la *x* indica que no la tiene.

Tabla 1. Características de las herramientas analizadas y la propuesta.

Características	*H1*	*H2*	*H3*	*H4*	*H5*
Representación del código con animaciones	×	✓	✓	✓	✓
Aplicación web	×	×	×	✓	✓
Representación de clases	×	✓	✓	×	×
Iluminado de bloques de código	✓	×	×	×	✓
Código editable	✓	✓	✓	×	✓
Representación visual de métodos	×	×	×	×	✓
Representación visual de estructuras control	×	×	×	✓	✓
Representación visual de paso parámetros	×	✓	✓	×	✓

4. Propuesta

Después de analizar las diferentes herramientas en la tabla de comparación, la herramienta web propuesta en este artículo servirá de apoyo al aprendizaje de la programación, ya que facilitará la comprensión de los programas vistos en clase o creados por los estudiantes.

El estudiante podrá entrar a la herramienta web y escribir su código en la zona correspondiente; posteriormente podrá hacer click en la opción para iniciar la animación, ya sea una animación continua o paso a paso. Para el caso de la animación continua se generará sin pausas de principio a fin; mientras que en la animación paso a paso se hará una pausa por cada instrucción de código, para que el estudiante interactué con el sistema y decida si prosigue al siguiente paso de la animación.

El entorno de la herramienta web que se desarrollará será una combinación de las diferentes herramientas ya estudiadas anteriormente, donde se tendrá una sección para escribir el código fuente elaborado en Java, una sección donde se dibujarán automáticamente todas las animaciones, y finalmente, la sección de los controles donde se podrá elegir si se inicia la animación, se reinicia o se ejecuta paso a paso. La herramienta web constará de distintos módulos independientes, tanto para el reconocimiento del código como para la animación y sus distintos escenarios.

La base de la herramienta está pensada para estudiantes principiantes en el tema, por lo cual solo se logrará hacer que la herramienta reconozca ciertos tipos de varia-

bles tanto privadas como públicas, tales como string, int, boolean, double y arreglos de los tipos mencionados; algunas estructuras de control de flujo de programa, tales como for, if, else, elseif, while, do while; y llamadas a funciones con parámetros. Para el desarrollo de la herramienta se utilizarán las siguientes tecnologías y lenguajes: HTML, JavaScript, CSS, JQuery, JQuery-ui, y three.js.

4.1. Objetivos

El objetivo general es diseñar e implementar una herramienta web enfocada a principiantes, para visualizar animaciones que mejoren la comprensión de programas escritos en el lenguaje de programación Java. Los objetivos particulares son los siguientes: 1) diseñar e implementar un mecanismo para identificar el programa del estudiante; 2) diseñar e implementar un mecanismo para desplegar automáticamente en pantalla variables en memoria, llamadas a métodos y estructuras de control de flujo de programa; 3) diseñar e implementar un mecanismo para permitir al estudiante seguir paso a paso la ejecución de un programa; 4) diseñar e implementar el sitio web; 5) determinar el conjunto de elementos que será posible reconocer por el sistema.

4.2. Metodología

El desarrollo de la herramienta web para visualizar la ejecución de programas en Java considera realizar las siguientes actividades: 1) elaboración del estado del arte, el cual ya fue concluido; 2) creación de un editor de código, el cual ya se ha iniciado; 3) diseño e implementación de un analizador léxico, básico, con el cual se pretende distinguir a partir del código fuente, los elementos que serán representados, el cual ya está en proceso de elaboración; 4) elaboración de los mecanismos para reconocer variables tanto públicas como privadas, inicialización de variables, asignación de valores, tipos de datos, métodos, parámetros enviados a través de los métodos y algunas estructuras de control; 5) diseñar e implementar los módulos de animación para los elementos que serán representados, dichos módulos serán para representar variables, funciones y estructuras de control; 6) elaborar los mecanismos que dibujen en pantalla las representaciones del código fuente; 7) desarrollar los mecanismos de animación para las representaciones; 8) implementar la interacción de la animación con el usuario; 9) desarrollo de controladores para manipular la ejecución del código fuente y de su respectiva representación.

4.3. Interfaz preliminar de la herramienta

En esta sección se muestran algunas capturas de pantalla de la interfaz preliminar de la herramienta web para visualizar la ejecución de programas escritos en el lenguaje Java. En la Figura 4 se muestra la captura de pantalla de la interfaz de inicio de la herramienta, donde se aprecian tres paneles: el panel del lado izquierdo es para escribir el programa en el lenguaje Java. El panel central es donde se visualizará la animación de la ejecución del programa, y el panel inferior contiene dos botones, uno para ejecutar el programa de manera continua, y uno para ejecutarlo o paso a paso.

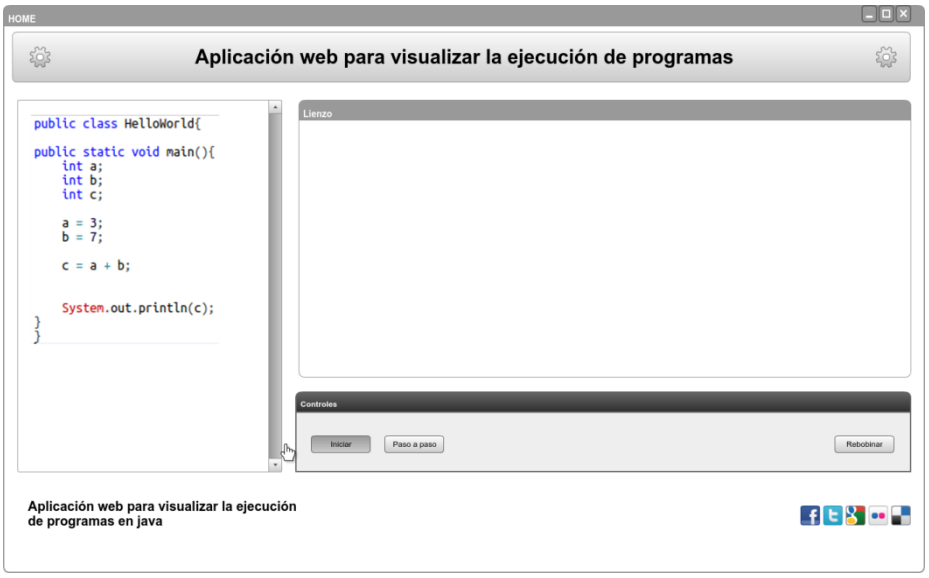

Fig. 4. Vista preliminar de la herramienta propuesta.

En la Figura 5 se muestra la captura de pantalla de la interfaz, una vez que ha empezado la ejecución del programa. Se aprecia que el programa ya ha creado tres variables (a, b y c) en el panel de visualización, las cuales aún no tienen ningún valor asignado. También puede apreciarse que el nombre del procedimiento actual es main(). La flecha muestra en el panel izquierdo la instrucción que se está ejecutando.

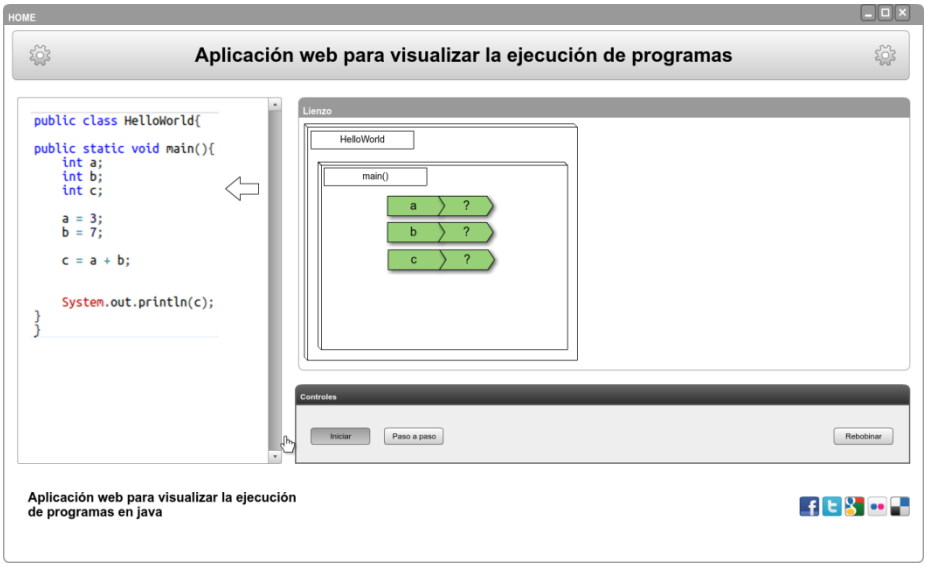

Fig. 5. Vista preliminar de la herramienta al ejecutarse el programa.

En la Figura 6 se muestra la interfaz una vez que se han asignado valores a las variables a y b. En el panel de visualización se puede apreciar la animación que dibuja 3 + 7 = 10, para asignarlo a c. Estos pasos se van mostrando en la herramienta de manera animada. La Figura 7 muestra cómo el valor de 10, resultado de la operación de a + b, viaja del área de operaciones hacia la casilla de memoria donde está la variable c.

Fig. 6. Ejecución de una operación y una asignación.

Fig. 7. Resultado de operación viaja a casilla de memoria.

En la Figura 8 se observa que el valor de 10 ha llegado a la casilla de memoria de la variable c, y el signo de interrogación ha desaparecido. Finalmente, la última instrucción del programa imprime el valor de la variable c a la consola, por lo que en el panel de visualización se presenta una consola donde el valor de c se imprime.

Fig. 8. Resultado de operación llega a casilla de memoria.

5. Conclusiones y trabajo futuro

El propósito de este artículo fue presentar una propuesta de herramienta web de apoyo para el aprendizaje de la programación a nivel universitario, la cual permitirá visualizar una animación que representará la ejecución de programas escritos en el lenguaje de programación Java. La herramienta propuesta apoyará en la comprensión de programas y entendimiento de los conceptos básicos de la programación, tales como creación de variables, asignación de valores a variables, uso de estructuras de control de flujo de programa y llamadas a funciones con paso de parámetros.

Como trabajo futuro se deben llevar a cabo las actividades propuestas en la metodología presentada, las cuales fueron concebidas después de realizar un análisis detallado de los requerimientos que la herramienta propuesta contendrá. Asimismo, se llevarán a cabo pruebas de funcionalidad y usabilidad con estudiantes y profesores.

Una vez que se tenga una versión estable de la herramienta web, se planea incorporar a una plataforma web de tutoriales interactivos que se ha diseñado y desarrollado por uno de los autores de este artículo. Esto último permitirá a los docentes crear tutoriales que utilicen esta herramienta web para sus ejemplos y ejercicios, principalmente para cursos relacionados con la programación.

Referencias

1. Berón, M., Uzal, R., Henriques, P. R., Varanda Pereira, M. J.: Comprensión de programas por inspección visual y animación. In: IX Workshop de Investigadores en Ciencias de la Computación (2007)
2. Bernardis, H., Berón, M., Riesco, D., Henriques, P., Pereira, M. J.: Extracción de información dinámica en programación orientada a objetos (Java). In: XIII Workshop de Investigadores en Ciencias de la Computación, Red de Universidades Nacionales con Carreras de Informática (RedUNCI) (2011)
3. Berón, M., Riesco, D., Montejano, G., Rangel, P., Pereira, M.: Estrategias para Facilitar la Comprensión de Programas. In: Workshop de Investigadores en Ciencias de la Computación, El Calafate, Santa Cruz, Argentina (2010)
4. Miranda, E., Berón, M., Montejano, G., Riesco, D., Henriques, P., Pereira, M. J.: Visualización de software: conceptos, métodos y técnicas para facilitar la comprensión de programas. In: XIII Workshop de Investigadores en Ciencias de la Computación, Red de Universidades Nacionales con Carreras de Informática (RedUNCI) (2011)
5. Berón, M., Uzal, R., Henriques, P. R., Varanda Pereira, M. J.: Inspección de código para relacionar los dominios del problema y programa para la comprensión de programas. In: X Workshop de Investigadores en Ciencias de la Computación (2008)
6. Berón, M.: Tesis Doctoral Inspección de Programas para Interconectar las Vistas Comportamental y Operacional para la Comprensión de Programas
7. Aquino, J. R., Hernández, J. H., Gutiérrez, D. A., Colorado, G. H.: Comparativa Entre Paquetes de Software de Apoyo a la Enseñanza de la Programación en Java
8. Berón, M., Henriques, P. R., Varanda Pereira, M. J., Uzal, R.: Herramientas para la comprensión de programas. In: VIII Workshop de Investigadores en Ciencias de la Computación (2006)
9. Moroni, N., Señas, P.: SVED: Sistema de visualización de algoritmos. In: VIII Congreso Argentino de Ciencias de la Computación (2002)
10. BlueJ. Disponible en: `http://bluej.org`
11. Jeliot. Disponible en: `http://cs.joensuu.fi/jeliot/`
12. jGRASP. Disponible en: `http://jgrasp.org`
13. Scratch. Disponible en: `http://scratch.mit.edu`
14. Scitools. Disponible en: `http://scitools.com`
15. CodeSurfer. Disponible en: `https://www.grammatech.com/products/codesurfer`
16. Imagix. Disponible en: `http://www.imagix.com/products/source-code-analysis.html`
17. Shrimp. Disponible en: `http://www.thechiselgroup.com/shrimp/`
18. Alma. Disponible en: `http://epl.di.uminho.pt/~gepl/ALMA/`