JAVATRACEIT!: software didáctico de apoyo a la docencia en Java

D. Glez-Peña, F. Fdez-Riverola, J.R. Méndez
Dpto. de Informática
Universidad de Vigo
32004 Ourense
e-mail: { riverola, moncho.mendez }@uvigo.es

F. Díaz
Dpto. de Informática
Universidad de Valladolid
40005 Ourense
e-mail: fdiaz@infor.uva.es

Resumen

El objetivo de este trabajo es dar a conocer una nueva herramienta didáctica de apoyo a la docencia en asignaturas con contenidos curriculares que incluyan Java como lenguaje de programación. En concreto, JAVATRACEIT! proporciona un entorno integrado de desarrollo adaptado a Java, que permite la compilación y el análisis de código ejecutable gracias a la implementación de un sencillo depurador y optimizador, que permiten al alumno conocer el estado de un programa en todo momento.

1. Introducción y motivación

En los últimos años, Java se ha convertido en uno de los lenguajes de programación más usados por la comunidad de desarrolladores, tanto a nivel comercial como a nivel de software libre. Este hecho, se ve reflejado en su inclusión formando parte de los numerosos temarios de estudios de informática, ya sea en el mundo universitario, ciclos formativos de grado medio y superior, academias, etc. Para la docencia de Java, se ha de tener en cuenta que ningún lenguaje de programación se escapa a la máxima de *programar se aprende programando*, por lo que son bienvenidas las herramientas que den soporte a la enseñanza práctica de los lenguajes de programación orientados a objetos.

JAVATRACEIT! [1] surgió como experiencia piloto en la Escuela Superior de Ingeniería Informática de la Universidad de Vigo [2], a partir de la necesidad de disponer de una herramienta gratuita, potente y de fácil manejo para el desarrollo de programas Java por parte de los alumnos. El proyecto, que tuvo sus inicios en el curso académico 2002/2003, se utiliza con éxito hoy en día en las diferentes asignaturas que manejan conceptos de orientación a objetos y lenguajes de programación. La idea consistió en la construcción inicial y posterior implantación de una herramienta de *depuración* y *optimización* de programas Java.

Los depuradores constituyen un componente presente en la mayoría de los entornos integrados de desarrollo (IDE), cuya misión es la de facilitar la búsqueda de errores de programación, principal causa de comportamientos no deseados en los programas desarrollados. Las funcionalidades generales que aportan los depuradores de propósito general, y más concretamente JAVATRACEIT!, son las siguientes:

- Ejecución, paso a paso, de las sentencias del programa a nivel de código fuente.
- Establecimiento de puntos de ruptura (*breakpoints*) en el código fuente.
- Visualización y modificación de los valores de las variables y tipos no primitivos.

Los optimizadores, también llamados *profilers*, proporcionan funciones destinadas a obtener información acerca de la ejecución de un programa para tratar de localizar puntos críticos donde el rendimiento no es el esperado, con la finalidad de mejorarlo y aumentar su eficiencia. Un ejemplo de información útil para lograr optimizar un programa es el uso que se hace de la memoria en un momento dado, analizando cuánta se usa y cómo se emplea.

JAVATRACEIT! proporciona un analizador de memoria que facilita las tareas de optimización. Cuenta además con un editor avanzado de código con funcionalidades como sintaxis resaltada, numeración de línea, visualización de paréntesis, etc. Con JAVATRACEIT! se consigue proporcionar a los alumnos una herramienta de apoyo a la programación que permite su iniciación y perfeccionamiento, teniendo disponible desde un principio los componentes más útiles de los grandes IDE comerciales. En este sentido, un depurador/optimizador no sólo sirve para detectar errores, es además muy útil para comprender el funcionamiento de cualquier lenguaje.

2. Arquitectura de JAVATRACEIT!

JAVATRACEIT! está implementado utilizando la plataforma JPDA (*Java Platform debugger Architecture*) [3] estructurada en tres capas:

- JVMTI (*Java Virtual Machine Tool Interface*) [4]. Especifica servicios a bajo nivel que proporciona la JVM (*Java Virtual Machine*) con el fin de ser utilizados por módulos nativos (librerías compartidas) que actúan a modo de cliente.
- JDWP (*Java Debug Wire Protocol*). Define un protocolo de comunicación vía sockets o memoria compartida entre el módulo nativo y un posible *front-end*.
- JDI (*Java Debug Interface*). Es una API 100% Java para el desarrollo de depuradores a modo de *front-end*.

Para desarrollar el *depurador* de JAVATRACEIT! se ha trabajado exclusivamente a nivel de JDI (capa superior), debido a que existe un módulo nativo JVMTI disponible en las distribuciones del JDK. Sin embargo, para el desarrollo del *optimizador* ha sido necesario trabajar con JVMTI, ya que JDI está orientado solamente a depuradores, no a tareas de optimización. Es por ello que se ha tenido que implementar un módulo nativo *ad-hoc*.

En resumen, JAVATRACEIT! puede verse como un *front-end* de depuración y optimización realizado en Java. Por lo tanto, durante la ejecución de la herramienta existen dos JVM: una para el programa a depurar/optimizar y otra para el propio JAVATRACEIT!. Los detalles de bajo nivel de la herramienta pueden consultarse en [5-6].

3. Utilización de JAVATRACEIT!

El aspecto que presenta la ventana de JAVATRACEIT! es similar a la que muestran los IDE comerciales (ver Figura 1).

JAVATRACEIT! proporciona un área de edición de código, además de varios paneles y herramientas para el alumno.

A continuación se describen brevemente las funcionalidades más comunes de JAVATRACEIT!, para su utilización por parte de un alumno de primer curso de una titulación de ciclo corto en informática.

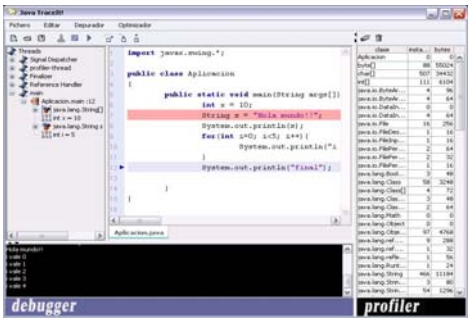

Figura 1. Ventana principal de JAVATRACEIT!.

3.1. Tratamiento de ficheros fuente

Antes de poder depurar las aplicaciones de usuario es necesario abrir sus ficheros, o al menos el fichero que contiene el método de entrada. En este sentido, se proporcionan una serie de utilidades básicas para poder gestionar estos archivos. Entre estas utilidades se encuentran: *nuevo fichero*, *abrir fichero*, *guardar*, *cerrar*, etc. El usuario tiene la posibilidad de tener abiertos tantos ficheros fuente como desee. La Figura 2 muestra el editor de código con múltiples ficheros abiertos.

Figura 2. Editor de código de JAVATRACEIT!.

3.2. Ejecución paso a paso

La depuración *paso a paso* permite controlar la ejecución de un programa. Para ello, se proporcionan tres tipos de *pasos* o formas de avanzar por el código fuente en ejecución: *step over, step into* y *step out*.

Step over es el más sencillo de todos. Cuando se está en una situación de parada ante una línea de código, si se solicita un *step over*, se tratará de ejecutar la línea de código en su totalidad. Una vez ejecutada, se estará en una nueva situación de parada en la línea siguiente. Es por tanto, un salto

de una línea de código. La Figura 3 muestra la ejecución de este tipo de salto.

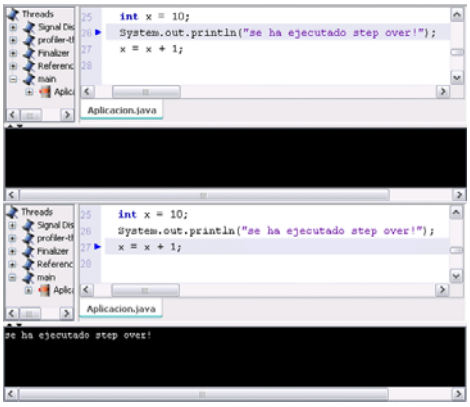

Figura 3. *Step over* en JAVATRACEIT!.

Step into avanza hacia el interior de la línea de código, es decir, busca la siguiente línea de código que se ejecuta internamente. Una sentencia puede incluir una o varias llamadas a otros métodos. Un *step into* provocará una nueva situación de parada inmediatamente antes de ejecutar la primera línea de código del primer método llamado en la sentencia exterior. Si la sentencia exterior no provoca llamadas a otros métodos, el efecto es el mismo que un *step over*. La Figura 4 muestra la ejecución de este tipo de salto.

Figura 4. *Step into* en JAVATRACEIT!.

Step out provoca que se continúe ejecutando el programa hasta que se salga del método actual, es decir, se busca el retorno al método que ha llamado al actual. Es posible que no exista ninguno más externo, por lo que *step out* provocará la finalización del hilo que ejecuta dicho método, que si es el principal, finalizará la ejecución del programa. La Figura 5 muestra la ejecución de este tipo de salto.

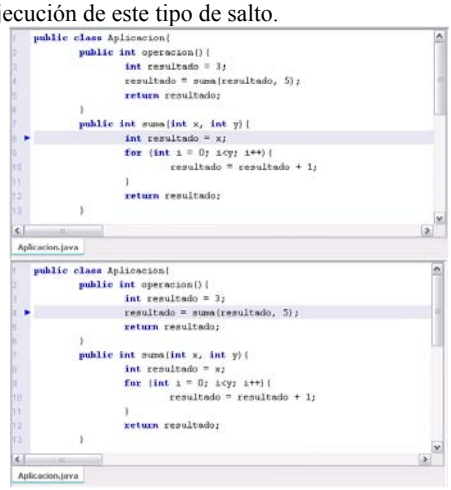

Figura 5. *Step out* en JAVATRACEIT!.

3.3. Puntos de ruptura

Los puntos de ruptura o *breakpoints* permiten detener la ejecución del programa en una línea de código. Cuando se coloca un punto de ruptura, la ejecución se detendrá cuando cualquiera de los hilos de ejecución trate de ejecutar la sentencia donde se ha colocado el *breakpoint*, provocándose una nueva situación de parada.

JAVATRACEIT! permite colocar o descartar puntos de ruptura tanto antes de la ejecución, como durante ella, lo cual facilita mucho la tarea de depuración. Los puntos de ruptura en una línea de código se muestran en un tono rojizo, tal y como muestra la Figura 6.

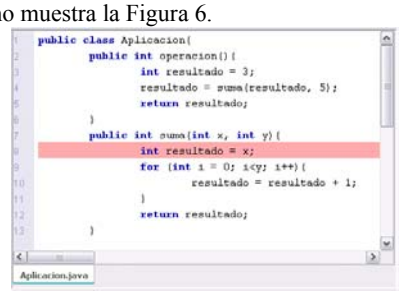

Figura 6. Punto de ruptura en JAVATRACEIT!.

3.4. Gestión de variables en memoria

La gestión de las variables es una tarea muy importante en la depuración, ya que la mayoría de los errores se descubren observando los valores erróneos que toman las variables. Para navegar por estructura de la memoria, JAVATRACEIT! proporciona el árbol mostrado en la Figura 7.

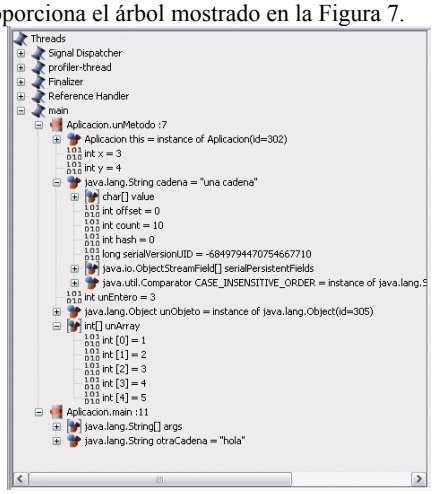

Figura 7. Árbol de memoria de JAVATRACEIT!.

Cuando el programa a depurar se encuentra detenido, se muestra su árbol de memoria. El alumno puede navegar jerárquicamente por los objetos que se encuentren visibles desde la posición actual del programa. En el nivel superior se encuentran los hilos de ejecución, dentro de ellos, los registros de activación o pila de llamadas a métodos y, finalmente, los objetos o variables, que a su vez contienen más objetos hasta llegar a los tipos de datos primitivos. Es preciso destacar que JAVATRACEIT! también permite modificar el valor de las variables de tipo primitivo.

3.5. El optimizador de aplicaciones

JAVATRACEIT! dispone de un analizador de memoria para ayudar al alumno a realizar tareas de optimización o *profiling*. Con el analizador de memoria se puede ver el uso de memoria que hace el programa a optimizar en el sentido de que se muestra cada clase Java, el número de instancias actual de dicha clase y los bytes que ocupa. Con ello, se pueden detectar posibles usos innecesarios de memoria que provocan un mal funcionamiento de la aplicación. La Figura 8 muestra su uso.

Figura 8. Analizador de memoria de JAVATRACEIT!.

4. Conclusiones

En la presente comunicación se ha presentado una herramienta de apoyo a la docencia en Java, uno de los lenguajes con mayor calado en la actualidad. Existen soluciones que en la mayoría de los casos suponen un coste muy elevado frente al carácter gratuito de JAVATRACEIT!, además de ser muy pesadas y poco manejables en ambientes docentes. Esta herramienta se encuentra disponible para su descarga libre en la dirección http://javatraceit.siteinteresa.com.

Referencias

[1] JAVATRACEIT! *http://javatraceit.siteinteresa.com*

[2] ESEI: Escuela Superior de Ingeniería Informática de la Universidad de Vigo. *http://www.ei.uvigo.es*. 2005.

[3] Sun Microsystems. Java Platform Debugger Architecture (JPDA). *http://java.sun.com/products/jpda/index.jsp.*

[4] Sun Microsystems. JVM Tool Interface v.1.0. *http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/jvmti.html.*

[5] Glez-Peña, D., Fdez-Riverola, F. *JavaTraceIt!, depurador y optimizador de aplicaciones Java*. I Congreso JavaHispano. 2003.

[6] Glez-Peña, D., Fdez-Riverola, F. *JVMTI, creación avanzada de profilers de aplicaciones con la nueva API de Tiger*. II Congreso JavaHispano. 2004.