Instituto Tecnológico de Costa Rica,

ITCR, Cartago, Costa Rica

Ingeniería en Computadores

Algoritmos y Estructuras de datos 1

Proyecto Programado 2

Profesor:

Antonio Gonzales Torres

Estudiantes:

Gunther Karolyi Gutierrez

2017238873

Eduardo Quiroga Alfaro

2017079534

Grupo 1

19/10/17

Índice

Introducción

El siguiente proyecto fue realizado por los estudiantes: Gunther Karolyi Gutiérrez y Eduardo Quiroga Alfaro, bajo la supervisión del profesor Dr.Antonio González Torres, en el Instituto Tecnológico de Costa Rica.El día 19 de octubre del año 2017.
El proyecto consiste en desarrollar un depurador para Eclipse en el que se pueda visualizar mediante un diagrama de flujo y a su vez mediante una línea que muestre la parte del código actual que se está ejecutando. Para esto vamos a utilizar una vista de Eclipse, por lo que la aplicación se va a visualizar en el mismo espacio de trabajo. El plugin tiene un funcionamiento bastante versátil al aceptar una gran variedad de código de Java, volviéndolo una herramienta con miles de propósitos, desde educación hasta verificación del funcionamiento de diferentes aplicaciones y programas. Para el desarrollo de la aplicación se debe analizar el código para dividirlo según sean las líneas (Ejemplo: If, Else, While…) por lo que se debe desarrollar un método que lea el código línea por línea y detecte los respectivos controladores según aparezcan. El plugin será desarrollado en Eclipse Neon y Oxygen, probado en Ubuntu 17.2 y Windows 10.

Descripción del problema

Para el plugin se deben desarrollar dos principales partes. Una siendo el constructor del diagrama de flujo que se va a visualizar durante el recorrido del código. Para el desarrollo de esta parte del código se debe buscar una manera de analizar el código línea por línea para detectar los distintos “Control Statements” o “Controladores” (If, Else, While, For…), y junto con esto, encontrar la manera de almacenar la condición de cada uno de estos para más adelante construir el diagrama de flujo con base en los datos analizados para que se pueda mostrar en la interfaz del plugin.

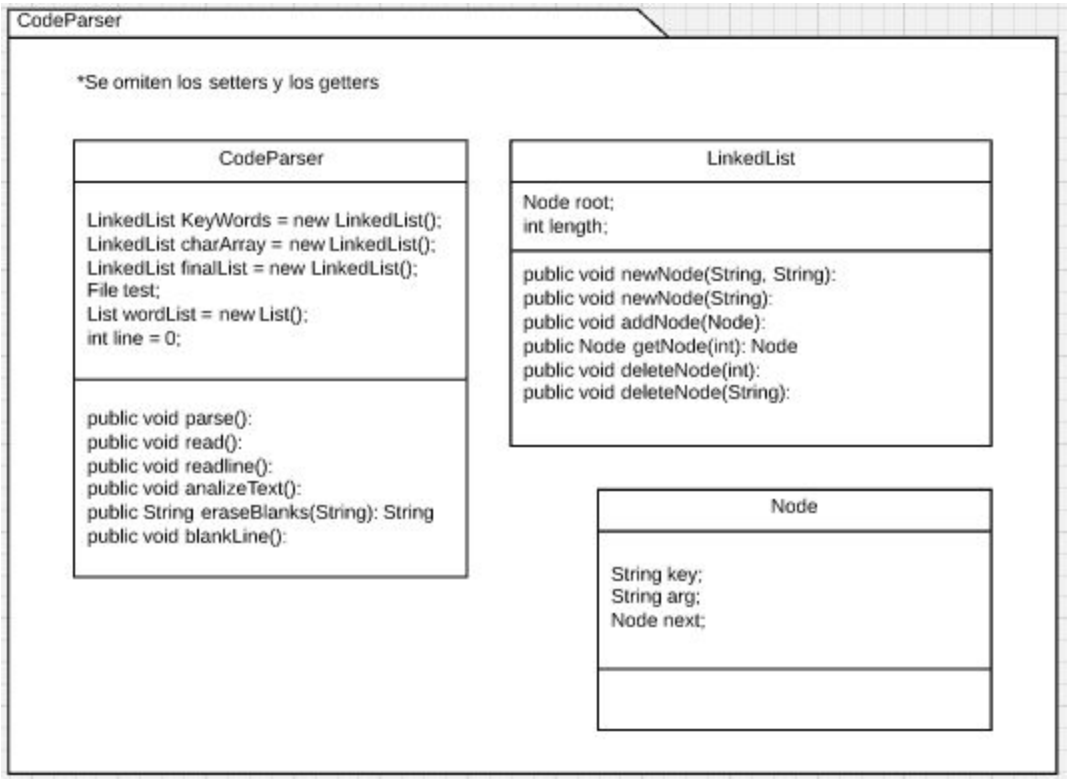
Por otro lado, también se debe considerar la parte de la depuración, donde debemos mostrar línea por línea lo que se vaya ejecutando en la máquina virtual de

Java, y esto lo podemos conseguir mediante la biblioteca de JPDA para recopilar los datos que devuelve la máquina virtual ante la ejecución del código.

Con estas dos partes del código desarrolladas se deben implementar en un plugin de Eclipse, que puede ser una ventana extra o una visualización dentro del mismo espacio de trabajo en la que podemos ver el código y la depuración simultáneamente. Para el plugin debemos considerar que cualquier tipo de código Java puede ser cargado, por lo que debemos implementar un explorador de archivos para seleccionar el código. El código también debe transformarse a un archivo de texto para que pueda ser analizado por el diagramador de flujo, por lo que debemos diseñar un método que transforme el archivo seleccionado de código Java, a texto común que sea posible recorrer e interpretar.

Para finalizar, hay que implementar todas las partes anteriores en un solo plugin de Eclipse para que sea versátil, eficiente y simple de utilizar dentro de la misma interfaz de Eclipse, para que de esta manera no se creen conflictos en el uso y este pueda ser útil y se le pueda sacar bastante provecho.

Diagramas de Clases del Diseño



Descripción de las Estructuras de Datos

Para el funcionamiento del plugin se implementó una estructura de datos solamente, que fue una lista enlazada simple. La lista enlazada simple consiste en una serie de nodos que se unen mediante un puntero que señala al nodo siguiente. Por ejemplo, en la lista tenemos una raíz, que viene siendo el primer nodo existente. Este nodo tiene un puntero dirigido al nodo siguiente, pero el nodo siguiente no tiene ningún puntero que señale el nodo anterior; por lo que es una lista simple y no doble.

Los nodos fueron desarrollados de forma que pudieran almacenar los datos necesarios, por lo que simplemente almacenan dos Strings, utilizados para almacenar en uno los “Control Statements” y en el segundo la condición del controlador almacenado en el primer String. Aparte de esto, la lista tiene todos los métodos y atributos comunes en listas, como el length y demás.

Descripción de los Algoritmos

Para el analisis del codigo utilizamos una clase llamada Code Parser, que tiene diferentes métodos para recorrer el texto línea por línea. Entre estos métodos tenemos:

- Parse: Parse consiste en un loop que almacena cada línea del código en la lista enlazada que construimos para este proyecto. Al mismo tiempo almacena los caracteres de la línea en otra lista por aparte uno por uno. Este método no tiene ninguna funcionalidad inmediata, pero es necesaria para el funcionamiento de todos los otros métodos.
- Read: Read consiste en un método que imprime el texto completo que se está analizando.
- Readline: Este método imprime una sola línea del texto que se está analizando, y guarda el índice de la última línea que se imprimió, por lo que si se vuelve a ejecutar, la línea que se va a imprimir será la siguiente a la última impresa.
- AnalizeText: Este método es el principal de la clase. Este es el que se encarga de analizar el texto línea por línea, y a la vez caracter por caracter para definir cual línea contiene un “Control-Statement” y cual es su respectivo condicional, o por otro lado si es una línea en la que se declara una variable, entre otras. Los resultados de este análisis los almacena en una nueva lista que será la lista final del análisis del texto.

- EraseBlanks: Se encarga de eliminar todas las tabulaciones y espacios en blanco que se encuentren en el texto para facilitar el análisis.
- Blankline: Elimina todas las líneas en blanco (/n) que se encuentren en el texto para que no sean detectadas como líneas extra del código.

En relación al algoritmo para conectar la virtual machine de java, se utiliza diversas iteraciones para buscar el puerto disponible y posteriormente mostrar en consola lo que está procesando en el jmv, para esto se creó la clase jvm.

Problemas Encontrados

El proyecto tiene un concepto simple a la hora de ser planteado, pero fue complicado en el proceso del desarrollo.

Uno de los principales problemas con los que nos encontramos fue la falta de documentación para el desarrollo del plugin de Eclipse. Por esta razón se complicó el desarrollo de la interfaz ya que no habían referencias de otras personas que hayan implementado el uso de los métodos necesarios para desarrollar algo similar, por otro lado, tampoco había documentos ni tutoriales que explicaran con algún tipo de detalle el proceso que se debía seguir en el desarrollo de la aplicación o de alguna vista dentro del espacio de trabajo. Debido al mismo problema, no se pudo encontrar una solución práctica y eficiente.

Entre las recomendaciones que podemos señalar para el proyecto, la principal que se destaca es averiguar y buscar toda la información necesaria para llevar a cabo las diferentes partes en las que se divida el proyecto.

También, para una mejor organización, dividir las partes del proyecto desde el principio para aprovechar el tiempo y modularizar las soluciones del mismo. De esta manera se pueden trabajar varias partes a la vez mediante diferentes compañeros de trabajo, para más adelante integrar las soluciones todas juntas.

Bibliotecas utilizadas

Para conectar el Java Virtual Machine mediante código Java fueron necesarias las siguientes bibliotecas:

- Java.io.*(Provides for system input and output through data streams, serialization and the file system.)
- com.sun.jdi.* (This package defines connections between the virtual machine using the JDI and the target virtual machine.)
- com.sun.jdi.event.* (This package defines JDI events and event processing.)

Conclusión

El programa como herramienta de análisis de código logra satisfacer los requerimientos del proyecto, de modo que conecta el virtual machine de java para el análisis en tiempo real de ejecución. Por otra parte se complicó el uso de jpda como debugger por falta de documentación y la interfaz gráfica presenta problemas de igual forma. También, se utilizó listas enlazadas como estructura de datos como solución en el algoritmo de análisis. De este modo se logra obtener mayor aprendizaje acerca de patrones de diseño, conceptos basicos de programacion y de la complejidad algorítmica. En consecuencia, se cumple con los objetivos de proyecto y se espera integrar mejoras al programa.

Bitácora

Actividad	**Tiempo (h)**
Análisis de requerimientos	3
Diseño de la aplicación	5
Programación	10

Documentación interna	1
Documentación de usuario	2
Documentación técnica	4
Pruebas	3
Total	28

Bibliografia

Para la realización de este proyecto se utilizaron las siguientes fuentes de información:

1. https://stackoverflow.com/questions/5711084/java-runtime-getruntime-getting-output-from-executing-a-command-line-program
2. https://stackoverflow.com/questions/18717831/tomcat-startup-ignoring-jpda-option-for-debug
3. http://illegalargumentexception.blogspot.com
4. http://www.vogella.com/tutorials/Eclipse3RCP/article.html

Anexos

Para la realización del proyecto se utilizó un repositorio de GitHub, el cual se puede encontrar en el siguiente enlace:

https://github.com/ed-quiroga-2103/Datos1_Proyecto2.git