**Instituto Tecnológico de Costa Rica**

**Ingeniería en Computadores**

**Algoritmos y Estructuras de datos I**

**Profesor: Antonio González Torres**

**Estudiantes: Esteban Campos Granados – 2017097066**

**Oscar González Alfaro – 2017121525**

**Año 2017**

**Indice**

|  |  |
| --- | --- |
| Introducción | 3 |
| Descripción del Problema | **3** |
| Diagrama de clases del diseño | **4** |
| URL del repositorio de versiones | **4** |
| Descripción de las bibliotecas utilizadas | **4** |
| Descripción de las estructuras de datos desarrolladas | **5** |
| Descripción detallada de los algoritmos desarrollados | **5** |
| Problemas encontrados | **6** |
| Bitacora | **6** |
| Conclusiones | **7** |
| Bibliografia | **7** |

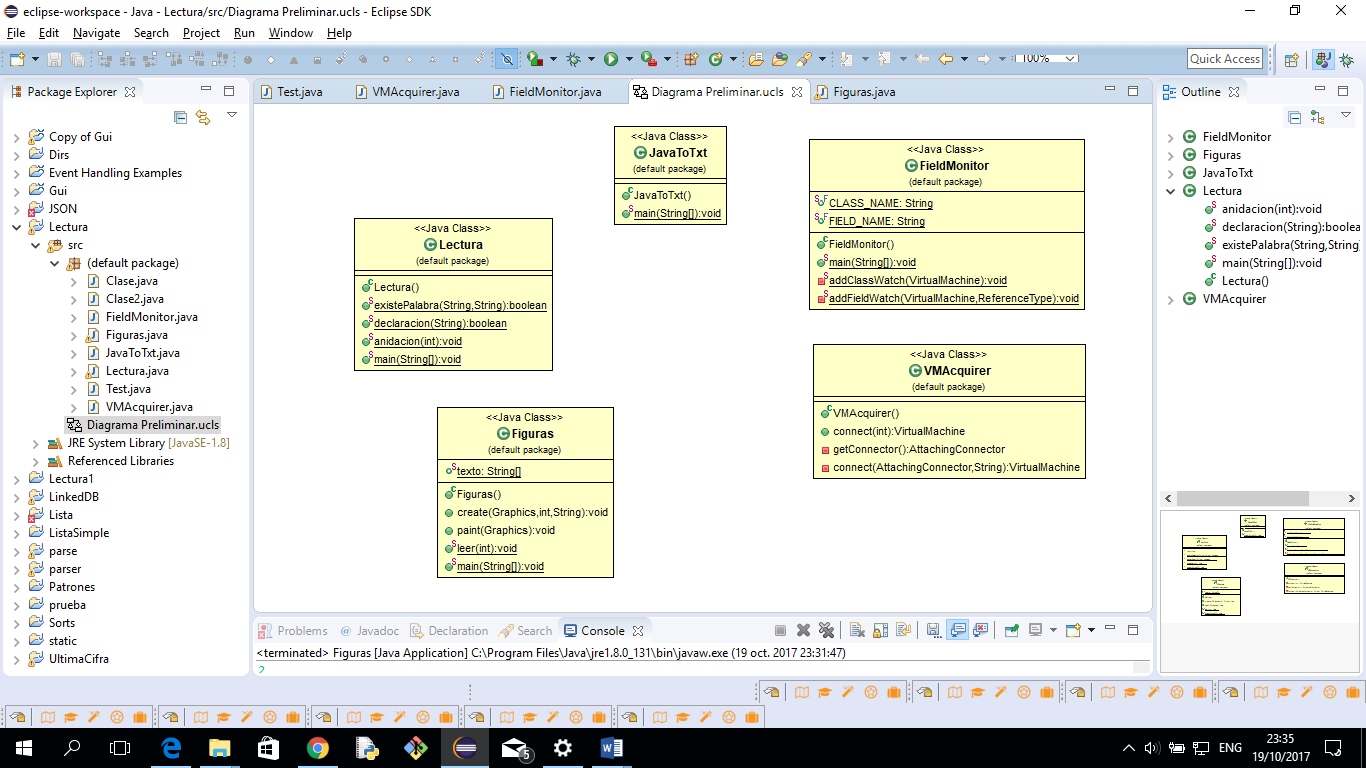
**Introducción**

Para este proyecto se nos plantea como objetivo principal el diseñar e implementar un plugin para la plataforma de software conocida como Eclipse, el cual va a tener como función principal la depuración y análisis visual de algoritmos facilitando el aprendizaje de conceptos básicos de programación, tanto programación orientada a objetos como los diversos patrones de diseño, los cuales nos van a ayudar que el código del programa sea más eficiente, versátil y de mejor comprensión , así como conceptos de bases de datos y agilizar el manejo del lenguaje de programación Java.

**Descripción del problema**

Como previamente se mencionó el proyecto se basa en el desarrollo de un plugin para la el software Eclipse, que permita la depuración de un programa de baja complejidad y a su vez nos permita conocer el tiempo de ejecución que tarda el sistema en resolverlo. En lo que respecta a la forma de actuar, primeramente, se debe poder escoger un programa y una vez seleccionado en una parte de la pantalla se debe desplegar el diagrama de flujo de este, el cual estará condicionado por el código fuente del programa previamente selecciono. La interfaz del sistema se compondrá de diversos componentes, en el lado izquierdo se podrá observar el código fuente del programa seleccionado y en el centro de la ventana se debe presentar el diagrama de flujo. Con respecto a este punto, se debe observar una animación en la cual se relacione con el código, en pocas palabras conforme se está haciendo la depuración se debe determinar la instrucción y asociarla con el diagrama. Y por último en el lado derecho de la ventana se debe observar los valores que toman las variables conforme a la depuración del programa.

**Diagrama de clases del diseño**



**URL del repositorio de versiones.**

<https://github.com/tsg3/CE1103---Proyecto-2/tree/master/Lectura>

**Descripción de las bibliotecas utilizadas.**

No se uso ninguna biblioteca externa o que no estuviese en el JDK de java

**Descripción de las estructuras de datos desarrolladas.**

**Arboles:**

Conjunto de nodos con datos enlazados, en la cualun nodo puede tener un padre y varios o ningun hijo. Usados para la animacion del diagrama

**Descripción detallada de los algoritmos desarrollados.**

**Lectura de clase java:**

Para ello, se toma la clase java a testear, creamos un archivo .txt a partir de esta, y obtenemos cada linea de codigo parseando el archivo de texto. Cada linea es evaluada por distintos criterios para determinar que se realiza en tal linea. Esto se utiliza a la hora de realizar el diagrama.

**Creacion del diagrama:**

Al darle Strings al generador del diagrama, este lo grafica en una ventana, encerrandolo en una figura y lo une con lineas a las demas figuras. Cada figura agregada, disminuye la altura en la que van a agregarse las proximas figuras

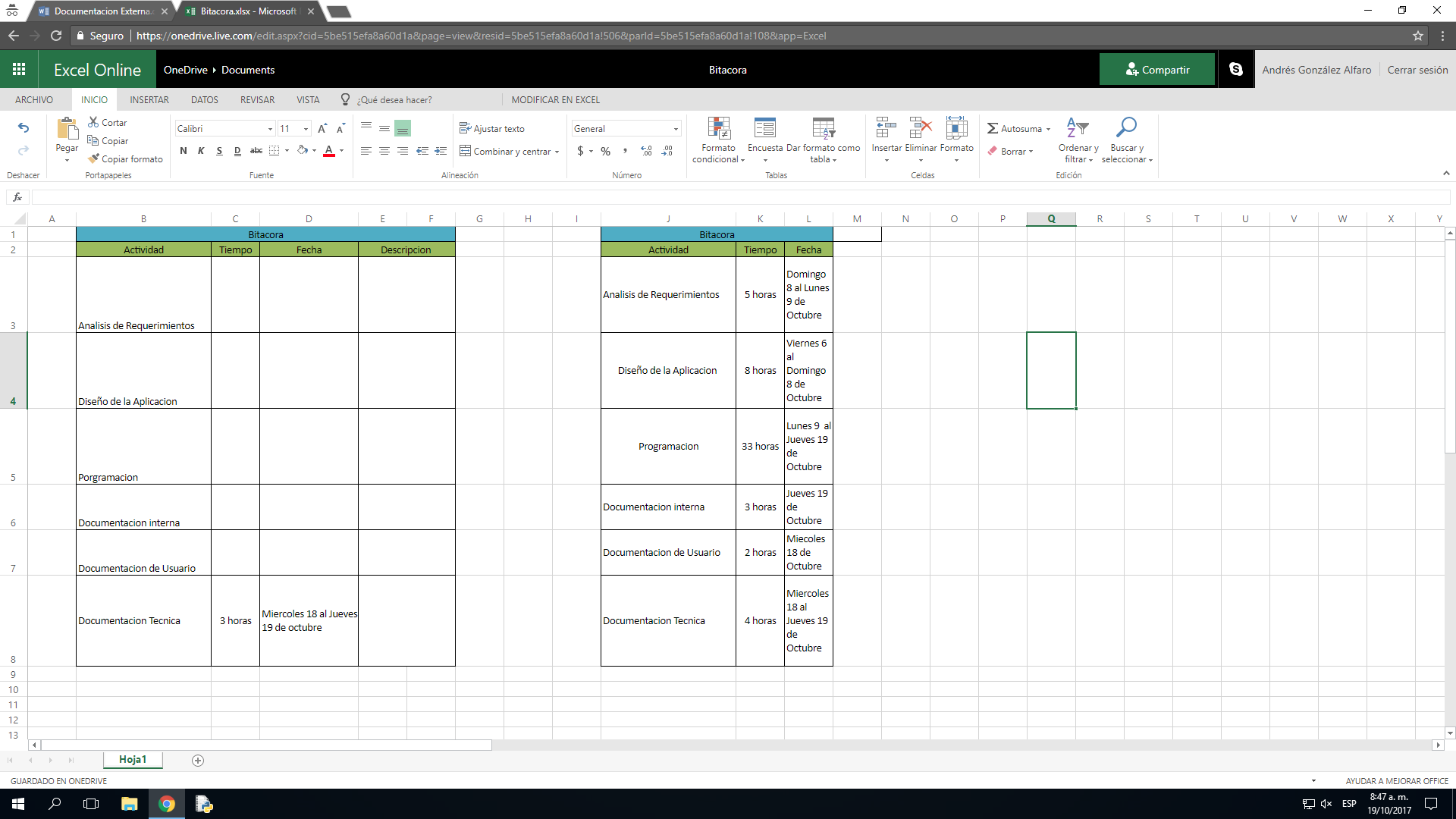
**Debugging:**

Se crea una instancia de la maquina virtual de Java, la cual se usa para realizar el testeo de la clase en cuestion

**Problemas encontrados**

No se logra la conexión a la maquina virtual de Java: Para realizar el debugging, se requeria conectarse a la maquina virtual de Java, lo cual no se logro hacer la conexión. Se intento con distintos puertos y direcciones distintas y desactivando el firewall de windows, pero sin éxito. Para solucionarlo se tuvo que usar un modelo distinto para acceder a la maquina virtual.

**Bitácora**



**Conclusiones**

Una vez finalizado el proyecto, se logró cumplir con el objetivo de diseñar e implementar un plugin de Eclipse, el cual tiene como función principal la de depurar código de baja complejidad y a su vez relacionar la instrucción que se está analizando con el diagrama de flujo mediante una animación sencilla, todo lo anterior presentado mediante el lenguaje de programación Java. Se abarcaron conceptos básicos de programación, así como los diversos patrones de diseño como, por ejemplo, Abstract Facthory, Facthory Method, Singleton, Composite e Iterator, enfocándonos en el patrón modelo vista controlador para lo que sería el diseño arquitectónico del plug-in. La interfaz del programa permite observar el código fuente del archivo seleccionado, el diagrama de flujo y una vez que se comience a depurar cada instrucción, se observa una animación en el diagrama que indica de forma visual, la parte del código que se está analizando, a su vez nos indica los valores de las variables locales y se actualizan conforme a la depuración, esto siendo posible a gracias a la conexión con la máquina virtual de Java implementada en el proyecto, al final dando como resultado un depurador de código con análisis visual, brindándonos la posibilidad de comprender la complejidad algorítmica del programa dado.

**Bibliografía**

Andrés Treviño (Publicado el 18 dic. 2012) Ejercicios Java 2 - Buscar palabra en enunciado (split). Recuperado de <https://www.youtube.com/watch?v=40ztJ1J8MiA>

Codigofacilito (Publicado el 6 mar. 2012) Leer Textos en Java. Recuperado de <https://www.youtube.com/watch?v=V2B8XW3-dck>

Michael Rennie, Andrew Niefer y Chris Aniszczyk (29 agosto del 2013) PDE. Recuperado de <http://wiki.eclipse.org/PDE>

Prof. Dr. Oscar Nierstrasz y Dr. David Rothlisberger(Diembre del 2010) A Quick-Start Tutorial to Eclipse Plug-in Development. Recuperado de <http://scg.unibe.ch/archive/projects/Bals10b-EclipsePlugins.pdf>

The SWT newsgroup (30 agosto del 2013) SWT: The Standard Widget Toolkit. Recuperado de <https://eclipse.org/swt/>

Fabian Lange (10 de octubre del 2011) Measure Java Performance – Sampling or Instrumentation?. Recuperado de <https://blog.codecentric.de/en/2011/10/measure-java-performance-sampling-or-instrumentation/>