Documentación Tarea Programada 2.

TI3404 Lenguajes de Programación

II SEMESTRE 2013

Estudiantes:
Gerardo Calderón
José David Hidalgo
Juliana Mora

Profesor:
Andréi Fuentes

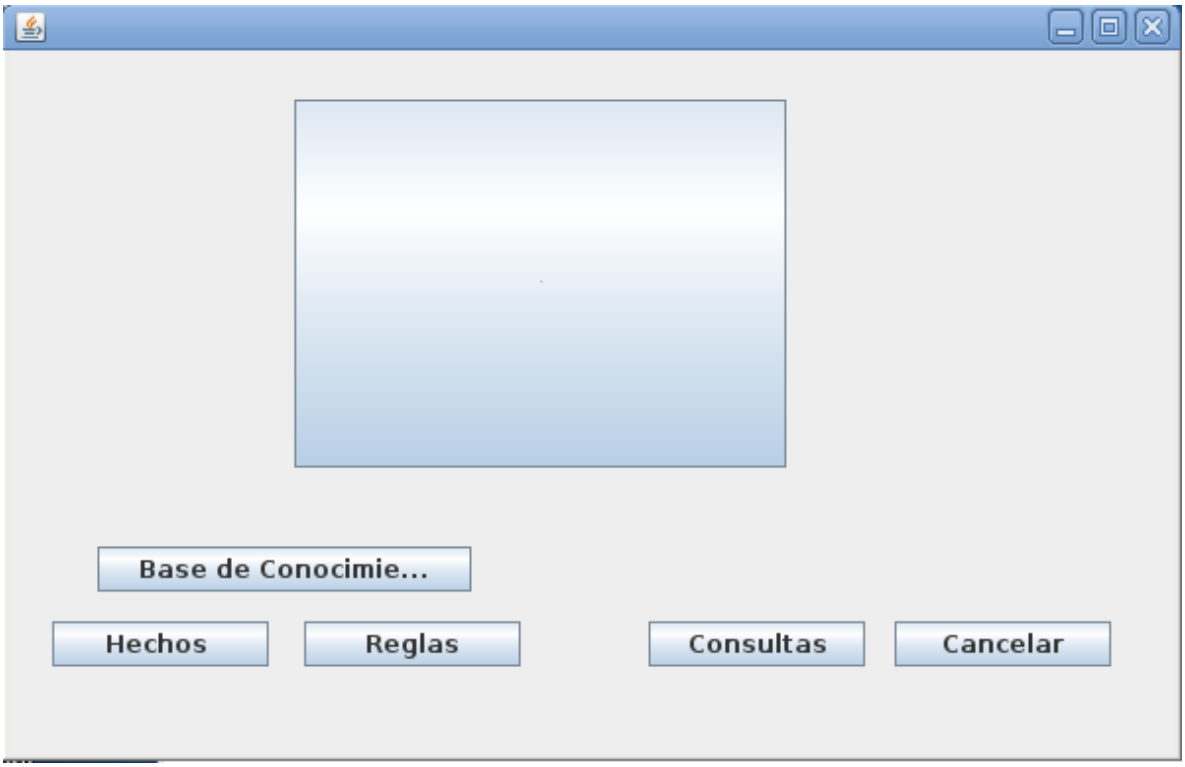
Índice

Descripción del problema

El objetivo de este programa es simular el Lenguaje de Programación, Prolog, basado en el Paradigma Lógico, en un Lenguaje de Programación del Paradigma Imperativo (ya sea Java®, C®, C++® o Python®).

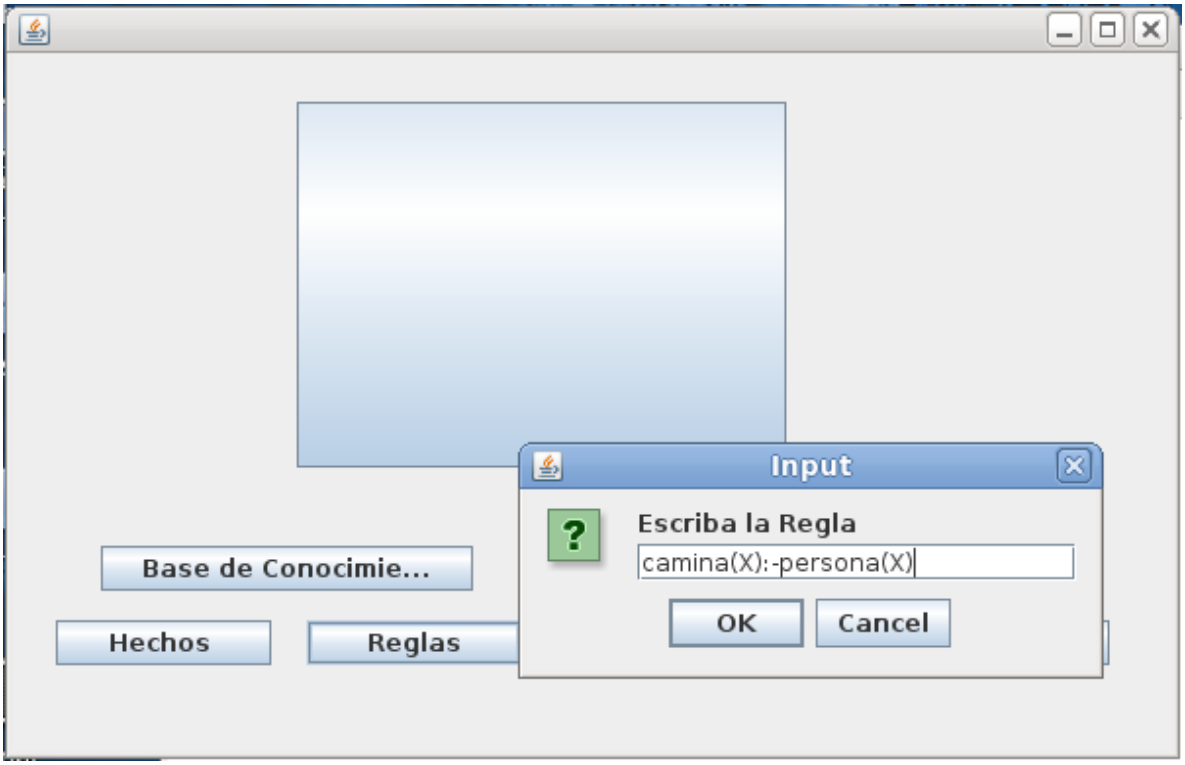
Para esto se utilizan reglas, hechos y demás, que describen los hechos tomados como verdaderos, los cuales serán guardados en una base de conocimientos, y éste se guarda en un archivo temporal (nosotros decidimos utilizar listas como nuestra base de conocimientos). Además, el programa deberá tener las “Built-In Predicates”, descritas como las funciones que tal lenguaje ya tiene “pre compilados” o ya implementadas. El programa deberá funcionar en una computadora que incluya el Sistema Operativo Linux®, ya que es necesario para la revisión y su correcto funcionamiento.

Funcionalidades del programa

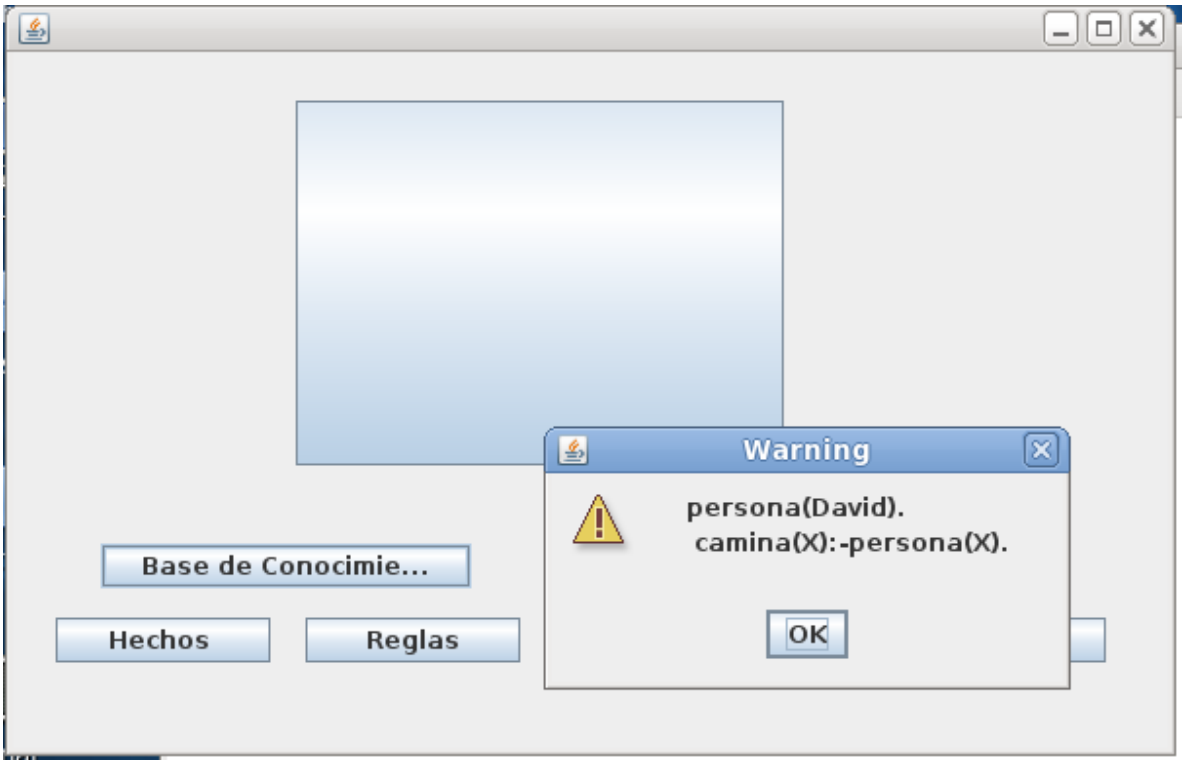


Agregar hechos a la base de conocimientos: En la interfaz gráfica, está el botón “hechos”, que permite el ingreso de hechos a la base de conocimientos, las hechos se validan al mismo tiempo que se ingresan, al presionar el botón “aceptar”, éstas se validan, y si no tienen errores, se ingresan a la base de conocimientos, éstas sirven para las consultas más adelante.

Agregar reglas a la base de conocimientos: Al igual que en el botón “hechos” está el botón “reglas”, que permite el ingreso de reglas a la base conocimientos, también, se verifican su validez léxica y sintáctica, y se agregan a dicha base de conocimientos.



Ingresar a base de conocimientos: Al presionar éste botón, se abre una pequeña ventana que despliega las reglas y los hechos agregado a la base de conocimientos, en ese orden respectivo.



Realizar consultas: En este modo, se ingresa la consulta que se quiere realizar, si el predicado ingresado corresponde con una regla o hecho en la base de conocimientos, se ejecuta la acción, para verificar si tal consulta es verdadera, si es una serie de reglas, se consulta a la base de conocimientos para verificar que las reglas sean verdaderas.

Diseño del programa

Este programa fue diseñado, en base a una interfaz gráfica, en otras palabras, se decidió la utilización de la interfaz, porque podría resultar confuso, para el usuario, operarlo por medio de consola:

- *Posiblemente digitaría alguna letra y/o número incorrecto, lo que resultaría en un error común.*

Se decidió, también, que el ingreso de hechos y reglas, fuera mediante ventanas de diálogo:

- *Se abre una nueva ventana, en la que se ingresan los hechos y las reglas.*
- *Resulta más práctico, en el sentido en que no tenga que escribir todos los hechos y reglas en una sola vez, si se equivoca en uno, puede reingresarlo, ya que al presionar el botón "aceptar" se verifica si lo que ingresó fue correcto, si no lo fue, el programa le permite su reingreso.*

Se utilizaron algoritmos específicos para:

- *Validación de las reglas y los hechos, desde verificar si la aridad es de 1 o de "n" elementos, hasta que tengan el punto al final de la oración.*
- *La creación de la ventana y botones de la interfaz gráfica del programa.*
- *Ingreso de reglas y hechos a la base de conocimientos, por medio de listas, el cual permite una fácil comprensión de su contenido, por medio de lo que el usuario ingresa.*
- *Ejecución de las reglas y los hechos, por medio del modo de consultas, diversos algoritmos ayudan a éste modo de ejecución, que ya están incorporados en el lenguaje.*

Se escogió el lenguaje de programación Java, por las siguientes razones: tenemos más experiencia en trabajar con ésta, además de su facilidad, por medio de objetos, para "encapsular" ciertas utilidades. Y es un lenguaje comprensible para entenderlo en su mayor parte. Además de que lo hemos utilizado por más de un semestre, en esta carrera, por tanto, nos resulta más fácil y de manera más eficiente.

Se utilizaron listas como archivos temporales para la base de conocimientos, debido a su sencilla interacción, y no pilas o colas, ya que un error de índices siempre es muy probable, entonces, un tipo de dato más estable fue lo que utilizamos. Y la ventaja de la temporalidad de las listas en la sola ejecución y lectura de éstas en este programa, es que son: fáciles de acceder, de interactuar y no se guardan en memoria luego determinado el programa, en vez de manejar un tipo de archivo distinto.

Librerías usadas:

Interfaz gráfica:

Biblioteca para el diseño de interfaz gráfica:

- import java.swing.*
- Se utilizan, para este programa:
    - JFrame (Recuadro de la ventana)
    - JPanel (Panel de la ventana)
    - Border (Bordes de la ventana)
    - ImageIcon (Inserción de imagen dentro del recuadro)
    - JLabel (Etiqueta, usada pricipalmente para texto)
    - JOptionPane (Esencial para el despliegue de cuadros de texto)
    - JTextField (Para el ingreso de texto por parte del usuario)
    - JButton (Despliegue de botones que realizan diversas funciones)

Biblioteca para la interacción con el usuario en interfaz gráfica

- import java.awt.* (Contenedor de varios componentes para la interacción con el usuario y también el diseño de interfaz gráfica)
- Se relaciona con el java.swing.*
- Se utilizan, para este programa:
    - Event.ActionEvent (Llamada a eventos de ciertos componentes)
    - Event.ActionListener (Lo que realiza cierto componente al ser utilizado (ya sea pasando sobre él, haciendo click, etc))
    - Color (Para agregar color a los componentes indicados)
    - Component (Representación gráfica de componentes)
    - EventQueue (Implementación de eventos o llamadas a funciones, que se almacenan temporalmente durante la ejecución del programa)

Manejo de “Strings”:

- Import java.util.StringTokenizer (Se utiliza para la división de Strings y analizarlos de manera independiente, generalmente tiene un delimitador, por las cuales separa el String en varias partes, y son enviadas como parámetros o utilizadas en variables)

Análisis de Resultados:

Se cumplieron ciertos de los objetivos principales, tales como: La funcionalidad de la administración de la base de conocimientos, por medio de la interfaz gráfica y el acceso a ésta por medio de botones en la misma, el motor de inferencia está implícito en el momento que se hacen las consultas, en el modo de consulta se trabaja con la consulta en la base de conocimientos, revisando hechos y reglas que coincidan, con la consulta realizada, dentro de nuestros métodos realizados.

Lo que no se implementó fue el backtracking y su funcionamiento, se propone terminarlo mediante, la implementación que teníamos pensado, pero ni pudimos por falta de comunicación.

Se ven errores en la implementación de la consulta con una aridad mayor a 1, fue un problema de último momento, en el cual nos estamos dando cuenta ahora.

Manual de usuario:

Bienvenido a la simulación del lenguaje Prolog, completamente trabajado en el lenguaje de programación: Java. Se está utilizando una interfaz gráfica para su mayor comodidad, cosa que no le resulte difícil de utilizar.

Ahora, comencemos con lo básico, lo primero que tiene que verificar es que usted tenga, este manual de usuario, el lenguaje de programación java (la última versión) instalada en su máquina, alguna distribución del sistema operativo de Linux, también ocupa los códigos en formato java llamados: "Principal.java" y "ListaSimple.java".

Existen varios manuales en internet, sobre cómo instalar Java en su máquina, dependiendo de su distribución, la versión actual de Java es la 7.4, luego de tener su distribución de Linux instalada y el lenguaje instalado y funcionando correctamente, siguen los demás pasos.

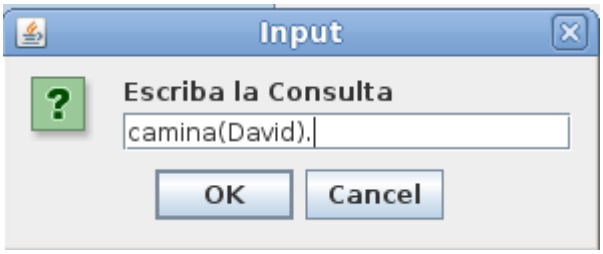
Antes de correr el programa, usted como usuario, debe saber cómo compilar en éste lenguaje, pero en vez de hacerlo buscar en internet por un manual (seguramente) incomprensible, aquí mismo le vamos a decir cómo compilar el código de vuestro programa. En primer lugar, debe tener los archivos (Principal.java y LstaSimple.java) en un mismo directorio, ya sea el escritorio o una carpeta específica dentro de su usuario, luego, en su línea de comandos (terminal u otro lugar), ingresar a la carpeta por medio del comando "cd (directorio o carpeta)" y luego accesa al compilador de java con "javac" y a la par escribe "Principal.java" ya que en éste código se comilarán los 2 códigos a la vez.

Si todo sale bien y no presenta ningún error, ya puede ejecutar el programa, con el comando "java Principal", eso permite la ejecución satisfactoria del programa.

Aquí veremos cómo utilizar el programa en unos sencillos pasos. Para agregar reglas o hechos, se presiona en alguno de los 2 botones, y escribe la regla o hecho bien, sino podría ocurrir esto:

Para consultas se presiona el botón de consultas, y de acuerdo a la base de conocimientos:

Se puede consultar cualquier cosa que se ocupe, según la base de conocimientos:



¡Y eso es todo! Ya luego de estas simples y claras instrucciones, no debería tener problemas al utilizar este programa, Esperamos que lo disfrute tanto como nosotros. Y recuerde que si tiene consultas sobre el funcionamiento del programa, o algpun error que se presente, ¡no dude en llamarnos! ¡Muchas Gracias!

----------Atentamente: El Equipo de Trabajo de la Aplicación----------

Conclusión:

Es un programa muy complejo, que tiene muchas diversas implementaciones para este programa, y el resultado es interesante, ver cómo trabaja un lenguaje de paradigma lógico, en uno imperativo; la materia vista en clase fue de ayuda importante para la realización del trabajo.

Se trabajó bastante con el lenguaje Java, su implementación en este programa fue confuso al inicio, pero al avanzar se entendió más, el objetivo del programa y las cosas que había que hacer/implementar, y además en la búsqueda de soluciones, por algún inconveniente que hubo en cuanto a funcionalidad de alguna parte específica.

Constantes pruebas fueron clave para determinar su funcionamiento interno, incluyendo la solución de problemas comunes que aparecían a lo largo del trabajo.

Para finalizar, fue una muy buena fuente de aprendizaje y la cantidad de esfuerzo no fue tomada en vano.

Referencias:

- Taringa.net. (2013). Métodos de la clase String. Recuperado de: (http://www.taringa.net/posts/apuntes-y-monografias/7545333/Metodos-de-la-clase-String.html) el 14 de Octubre del 2013.
- CodeRanch.com. (2013). Uninstalling java on Linux. Recuperado de: (http://www.coderanch.com/t/110702/Linux-UNIX/Uninstalling-java-linux) el 2 de Octubre del 2013.
- LinuxQuestions.org. (2013). Fedora 14 remove all java jdk jre and reinstall. Recuperado de: (http://www.linuxquestions.org/questions/linux-newbie-8/fedora-14-remove-all-java-jdk-jre-and-reinstall-866598/) el 2 de Octubre del 2013.
- Blogspot.com. (2013). El método SPLIT en Java. Recuperado de: (http://codigomaldito.blogspot.com/2011/06/el-metodo-split-en-java.html) el 10 de Octubre del 2013.
- Bucio.com.mx. (2013). Compilar Java desde la Terminal. Recuperado de: (http://bucio.com.mx/2010/compilar-java-desde-la-terminal.html) el 2 de Octubre del 2013.
- FedoraForum.org. (2013). ¿How do i completely uninstall all Java versions? Recuperado de: (http://forums.fedoraforum.org/showthread.php?t=276578) el 2 de Octubre del 2013.
- FedoraUnity. (2013). Using Java instead of JDK. Recuperado de: (http://fedorasolved.org/Members/zcat/using-sun-java-instead-of-openjdk) el 2 de Octubre del 2013.