Trabajo Práctico N°: 3 (Individual)

Implementación de un intérprete lógico por substitución

En el ámbito de la programación lógica los sistemas tienen una estructura que está dada por una base de datos -conformada por definiciones (facts) y reglas- más un intérprete que acepta consultas y las responde extrayendo información y operando sobre la base de datos.

Se pide desarrollar: un intérprete que, frente a consultas con todos los parámetros instanciados, responda afirmativamente o negativamente a dichas preguntas.

Las reglas deben leerse como una relación de implicación lógica, es decir que si se cumplen los objetivos de la derecha del operador :- , se cumple el objetivo de la izquierda y en definitiva la consulta tiene como respuesta SI, de lo contrario la respuesta es NO. Las letras mayúsculas indican variables sin instanciar. Las variables son sólo válidas dentro de una regla, por lo cual si aparecen en más de una regla representan ítems diferentes aunque tengan igual nombre.

Para acotar el alcance del problema las reglas serán compuestas solamente por conjunción de facts indicados por comas y debe tomarse a los objetivos (facts que componen a las reglas) como definiciones con sus parámetros sin instanciar. Atención, las decisiones de diseño que haga serán tenidas en cuenta.

Se pide una aplicación de consola implementada utilizando Java 8. La aplicación debe tener tests con una cobertura de código del 75% total.

Ejemplo de base de datos con definiciones y dos reglas.

```
varon(juan).
varon(pepe).
varon(hector).
varon(roberto).
varon(alejandro).
mujer(maria).
mujer(cecilia).
padre(juan, pepe).
padre(juan, pepa).
padre(hector, maria).
padre(roberto, alejandro).
padre(roberto, cecilia).
hermano(alberto, rodrigo).
hijo(X, Y) :- varon(X), padre(Y, X).
hija(X, Y) :- mujer(X), padre(Y, X).
```

Ejemplo de uso

```
padre(juan, pepe) → SI
padre(mario, pepe) → NO
hijo(pepe, juan) → SI
hija(maria, roberto) → NO
```

El sistema debe tomar la base de datos desde un archivo de texto y poder responder las consultas ingresadas por consola.

Restricciones

- Tp individual
- Implementado en Java
- Implementado con Test Unitarios
- Realizar fork de repositorio Git
- Los test de aceptación deben correr exitosamente, completar con más y diferentes modelos o más test de aceptación.
- La DB de entrada debe ser válida de forma completa al parsearse, si la misma no es correcta debe fallar la inicialización indicando un error en tal elemento de la db.
- La consulta de entrada debe estar bien formada, de no estarlo debe informarse con una excepción

Fecha de Entrega

- Fecha de Entrega: Antes del Lunes 6 de Noviembre 20hs via Github
- Crear y trabajar sobre un fork del repositorio: 7510-TP1-Java
- Realizar commits pequeños y atómicos por cada funcionalidad agregada o cada refactor realizado.
- Etiquetar en el github personal la entrega como: **entrega-tp-java**
- El lunes 6 de Noviembre se correrán tests de aceptación desde lo que haya en GitHub

Links Utiles

https://maven.apache.org/
https://www.digitalocean.com/community/tutorials/como-instalar-java-con-apt-get-en-ubuntu-16-04-es
https://www.eclipse.org/
http://www.mojohaus.org/cobertura-maven-plugin/

Git

https://githowto.com/
http://git-scm.com/
http://try.github.io/levels/1/challenges/1
http://www.codeschool.com/courses/try-git
http://pcottle.github.io/learnGitBranching/