Proyecto de Representación del Conocimiento

SWI - Prolog

Sistema que se fundamenta en una Base de Conocimiento de forma estructurada para determinar las Propiedades y Relaciones en Clases y Objetos.

2018

Equipo:

Jessica Sarahi Méndez Rincón

Juan Daniel Lawrence Pedroza

Marco Tulio Sánchez Rodríguez

Nahet Cortez Fuerte

Rodrigo Terpán Arenas

MAESTRÍA EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

INTELIGENCIA ARTIFICIAL

Semestre 1 2019-1

Profesores:

• Dr. Luis A. Pineda Cortes, IIMAS, UNAM

• Dr. Arturo Rodríguez García, Facultad de Ingeniería, UNAM

• Mtro. Iván Torres Rodríguez, PCIC, UNAM

Fecha de entrega Jueves 15 de Noviembre de 2018

Contenido

* [Objetivo 3](#_Toc529995206)
* [Base de Conocimientos 3](#_Toc529995207)
* [Ilustración 1 Diagrama de Taxonomía 5](file:///D:\Maestria%202018\Semestre1\03_Inteligencia%20Artificial\IA%20Proyecto%20de%20Representación%20del%20Conocimiento%2020181111.docx#_Toc529995208)
* [Archivo de la Base de Conocimiento 6](#_Toc529995209)
* [Diseño y Desarrollo de Módulos 7](#_Toc529995210)
* [Ejecución 8](#_Toc529995211)
* [Módulos del Sistema 8](#_Toc529995212)
* [Módulo de Consulta 8](#_Toc529995213)
* [Consulta de Clases 8](#_Toc529995214)
* [Consulta de Propiedades de Clases 9](#_Toc529995215)
* [Consulta de Preferencias de Propiedades de Clases 9](#_Toc529995216)
* [Consulta de Relaciones de Clases 9](#_Toc529995217)
* [Consulta de Preferencias de Relaciones de Clases 9](#_Toc529995218)
* [Consulta de Objetos 9](#_Toc529995219)
* [Consulta de Propiedades de Objetos 9](#_Toc529995220)
* [Consulta de Preferencias de Propiedades de Objetos 9](#_Toc529995221)
* [Consulta de Relaciones de Objetos 9](#_Toc529995222)
* [Consulta de Preferencias de Relaciones de Objetos 9](#_Toc529995223)
* [Módulo de Añadir 9](#_Toc529995224)
* [Añadir de Clases 9](#_Toc529995225)
* [Añadir de Propiedades de Clases 9](#_Toc529995226)
* [Añadir de Preferencias de Propiedades de Clases 9](#_Toc529995227)
* [Añadir de Relaciones de Clases 9](#_Toc529995228)
* [Añadir de Preferencias de Relaciones de Clases 9](#_Toc529995229)
* [Añadir de Objetos 10](#_Toc529995230)
* [Añadir de Propiedades de Objetos 10](#_Toc529995231)
* [Añadir de Preferencias de Propiedades de Objetos 10](#_Toc529995232)
* [Añadir de Relaciones de Objetos 10](#_Toc529995233)
* [Añadir de Preferencias de Relaciones de Objetos 10](#_Toc529995234)
* [Módulo de Eliminar 10](#_Toc529995235)
* [Eliminar de Clases 10](#_Toc529995236)
* [Eliminar de Propiedades de Clases 10](#_Toc529995237)
* [Eliminar de Preferencias de Propiedades de Clases 10](#_Toc529995238)
* [Eliminar de Relaciones de Clases 10](#_Toc529995239)
* [Eliminar de Preferencias de Relaciones de Clases 10](#_Toc529995240)
* [Eliminar de Objetos 10](#_Toc529995241)
* [Eliminar de Propiedades de Objetos 10](#_Toc529995242)
* [Eliminar de Preferencias de Propiedades de Objetos 11](#_Toc529995243)
* [Eliminar de Relaciones de Objetos 11](#_Toc529995244)
* [Eliminar de Preferencias de Relaciones de Objetos 11](#_Toc529995245)
* [Módulo de Modificar 11](#_Toc529995246)
* [Modificar de Clases 11](#_Toc529995247)
* [Modificar de Propiedades de Clases 11](#_Toc529995248)
* [Modificar de Preferencias de Propiedades de Clases 11](#_Toc529995249)
* [Modificar de Relaciones de Clases 11](#_Toc529995250)
* [Modificar de Preferencias de Relaciones de Clases 11](#_Toc529995251)
* [Modificar de Objetos 11](#_Toc529995252)
* [Modificar de Propiedades de Objetos 11](#_Toc529995253)
* [Modificar de Preferencias de Propiedades de Objetos 11](#_Toc529995254)
* [Modificar de Relaciones de Objetos 11](#_Toc529995255)
* [Modificar de Preferencias de Relaciones de Objetos 12](#_Toc529995256)

# Objetivo

Llevar a cabo la compresión de los temas de Inteligencia Artificial de Representación del Conocimiento con el Lenguaje de Programación Prolog mediante un Proyecto que ayude a construir una Base de Conocimiento para consultar la taxonomía de individuos en un dominio en particular con relaciones y propiedades.

De igual forma:

* Establecer el conocimiento de forma económica
* Usar la inferencia de manera eficiente

# Base de Conocimientos

Debido a que se requiere una taxonomía como ejemplo para el arranque del Sistema con pruebas y validación de los diversos módulos, el equipo analizó, diseño y estructuro una base con taxonomía del reino animal y con individuos que servirán de ejemplos para establecer propiedades y relaciones.

Definiéndose la siguiente estructura:

En base a Clases e Individuos u objetos se estructuraron sus propiedades y relaciones de la siguiente forma:

Para las Clases:

Class(nombre\_clase, nombre\_clase\_padre, nombre\_propiedad,nombre\_Relacion,objetos)

**Class**(nombre\_clase,padre\_clase,[],[],[]) =>Clase sin propiedades y sin relaciones

Para los objetos:

Id=>nombre\_objeto, nombre\_propiedad, nombre\_Relacion

**Id** =>nombre\_objeto,[],[] =>Objeto sin propiedades y sin relaciones

# Ilustración 1 Diagrama de Taxonomía

# Archivo de la Base de Conocimiento

[

class(top,none,[],[],[] ),

class(animales,top,[[vivos,0],[ponen\_huevos,0],[color=>desconocido,0],[[tamano=>grande,carnivoro]=>>[peligroso,2]]],[],[] ),

class(peces,animales,[[nadan,0],[acuatico,0]],[],[

[ id=>dory, [[memoria\_corto-plazo,0] ], [[not(amigo=>tux),0],[amigo=>nemo,0] ]],

[ id=>nemo, [[vive=>anemona,0] ], [[amigo=>dory,0] ]]] ),

class(aves,animales,[[vuela,0],[nace=>huevo,0],[carnivoro,0]],[],[

[ id=>piolin, [[color=>amarillo,0],[carisma=>risueno,0],[tamano=>pequeno,0],[tierno,0] ], [[mascota=>hominidos,0] ]]] ),

class(pinguinos,aves,[[not(vuelan),0]],[[come=>peces,0]],[

[ id=>pete, [[tamano=>pequeno,0] ], [[not(amigo=>arthur),0] ]],

[ id=>tux, [[trabaja=>computadoras,0] ], [[amigo=>arthur,0] ]]] ),

class(aguilas,aves,[[vive=>montana,0]],[],[

[ id=>arthur, [[tamano=>grande,0] ], [[not(amigo=>pete),0],[not(amigo=>tux),0] ]]] ),

class(mamiferos,animales,[[nace=>mama,0],[comen=>leche,0],[not(ponen\_huevos),0]],[],[

[ id=>rafiqui, [[color=>negro,0] ], []]] ),

class(ornitorrincos,mamiferos,[[nace=>huevo,0],[ponen\_huevos,0]],[],[

[ id=>perry, [[trabaja=>agente,0] ], [[enemigo=>doofenshmirtz,0],[dueno=>phineas,0] ]]] ),

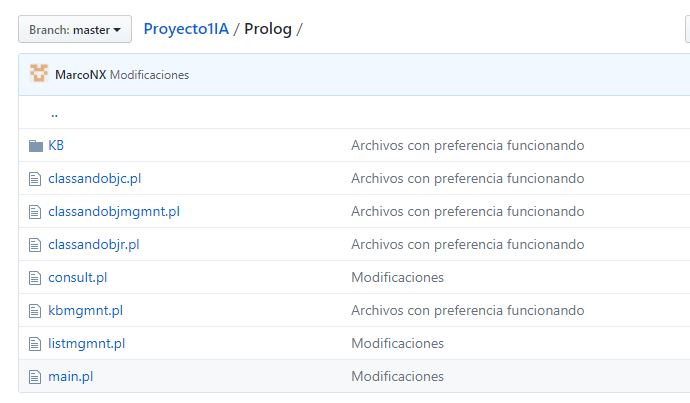
class(hominidos,mamiferos,[ [piensa=>yes,0], [seenamora=>yes,0]],[ [depredadorDe=>animales,0]],[

[ id=>phineas,[[vive=>arealimitrofe,0]],[[mascota=>perry,0]] ],

[ id=>doofenshmirtz,[[vive=>arealimitrofe,0],[trabaja=>maldad,0]],[[enemigo=>perry,0]] ]] )

]

# Diseño y Desarrollo de Módulos

Se diseñaron los archivos que contenían las diversas funcionalidades del Sistema en Prolog, de acuerdo a la estructura que se muestra en la imagen:

Teniendo en cuenta los módulos de

**main.pl**

Contiene el código para invocar las funcionalidades del Archivo así como para invocar el conjunto de funcionalidades que lleven a cabo una tarea para la transacción de información y guardado en el archivo de entrega final.

**listmgmnt.pl**

Contiene el código Core del resto de las funciones, son operaciones directas dentro de las listas que lee y codifica directamente del archivo en la base de conocimientos.

**kbmgmnt.pl**

Contiene los servicios de consulta de las Clases y de las propiedades, así como de relaciones de objetos.

**consult.pl**

Contiene los servicios de consulta de las Clases, las propiedades, las preferencias y relaciones .

**classandobjr.pl**

Contiene los servicios para el módulo de Eliminar, todas las funciones que se usan tanto para clases como para objetos, y la validación de las listas con las características necesarias.

**classandobjmgmnt.pl**

Contiene los servicios para el módulo de Modificar, todas las funciones para cambiar por ejemplo el nombre de una clase.

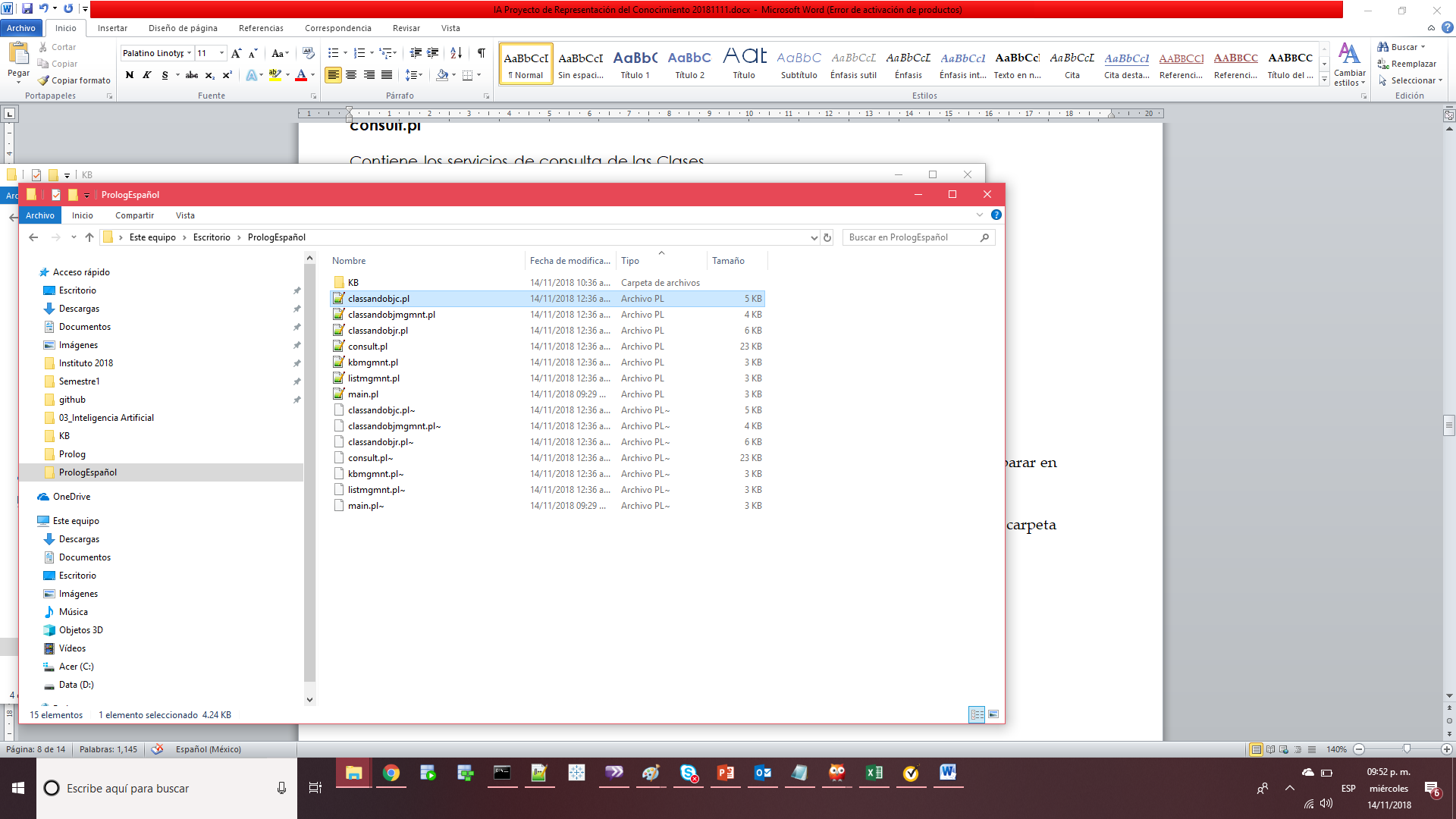
**Classandobjc.pl**

Contiene los servicios para el módulo de Agregar, todas las funciones para agregar clases y objetos, así como para agregar propiedades y relaciones tanto de clases como de objetos.

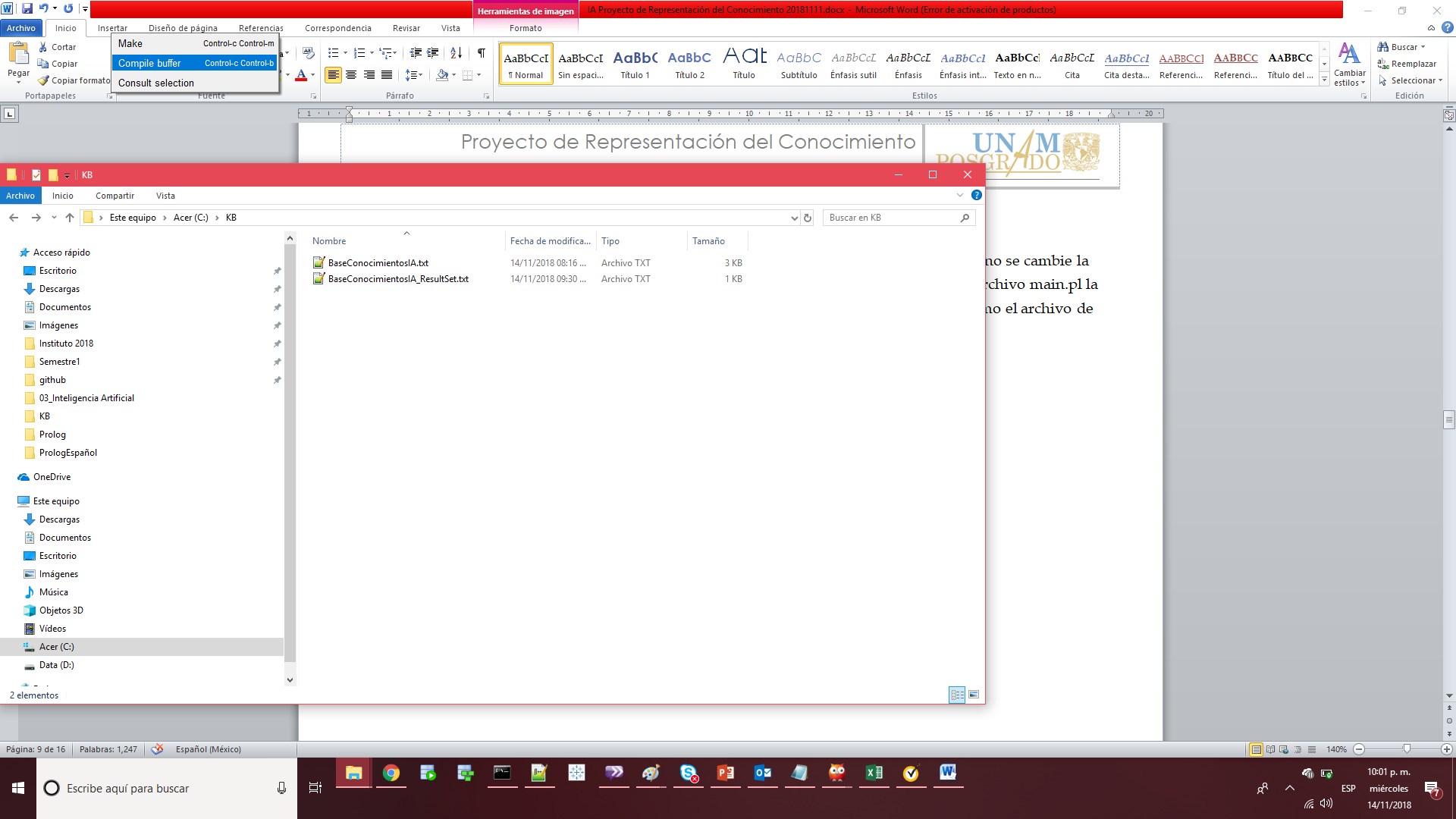
# Ejecución del Sistema

La forma más correcta en el diseño del sistema para el desarrollo de los diversos módulos, era separar en varios archivos el código y así de forma más ordenada llevar a cabo los trabajos de mantenimiento en la programación, ya que si se efectuaban cambios, estos sólo afectarían a los del módulo.

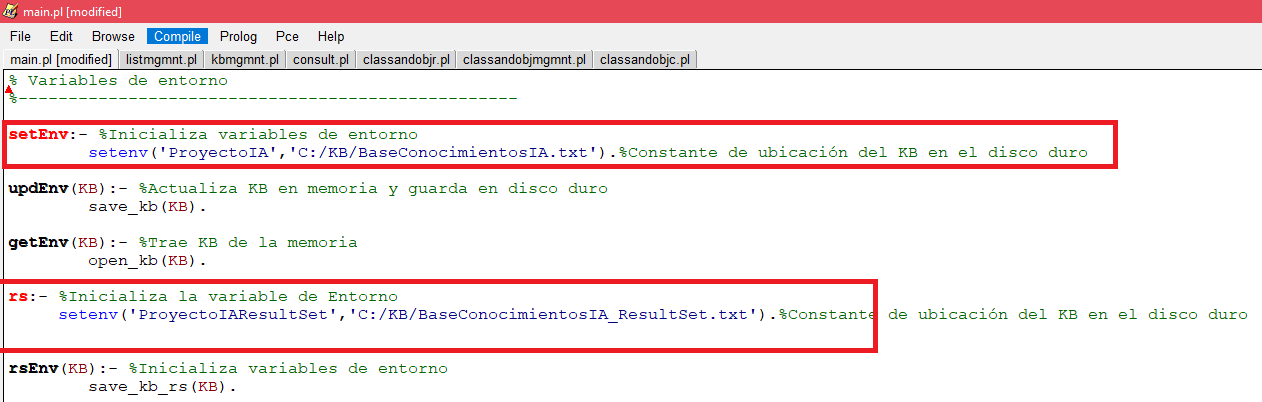
Después para que se ejecuten las consultas y las transacciones a la base de conocimiento, es necesario y de forma obligatoria estar dentro de una misma carpeta cuidando que el nombre de las funciones fuera único en toda la carpeta del proyecto, se logró crear los 7 archivos que se deben de compilar y abrir en Prolog.



La carpeta de la base de conocimiento deberá estar en la raíz de del Disco C, preferentemente para que no se cambie la variable global. En caso de que no se pueda realizar esto se deberá de modificar dentro del archivo main.pl la ruta de ubicación de la base de conocimiento y de igual forma el nombre si se cambia, así como el archivo de Result Set o de resultados de consulta.

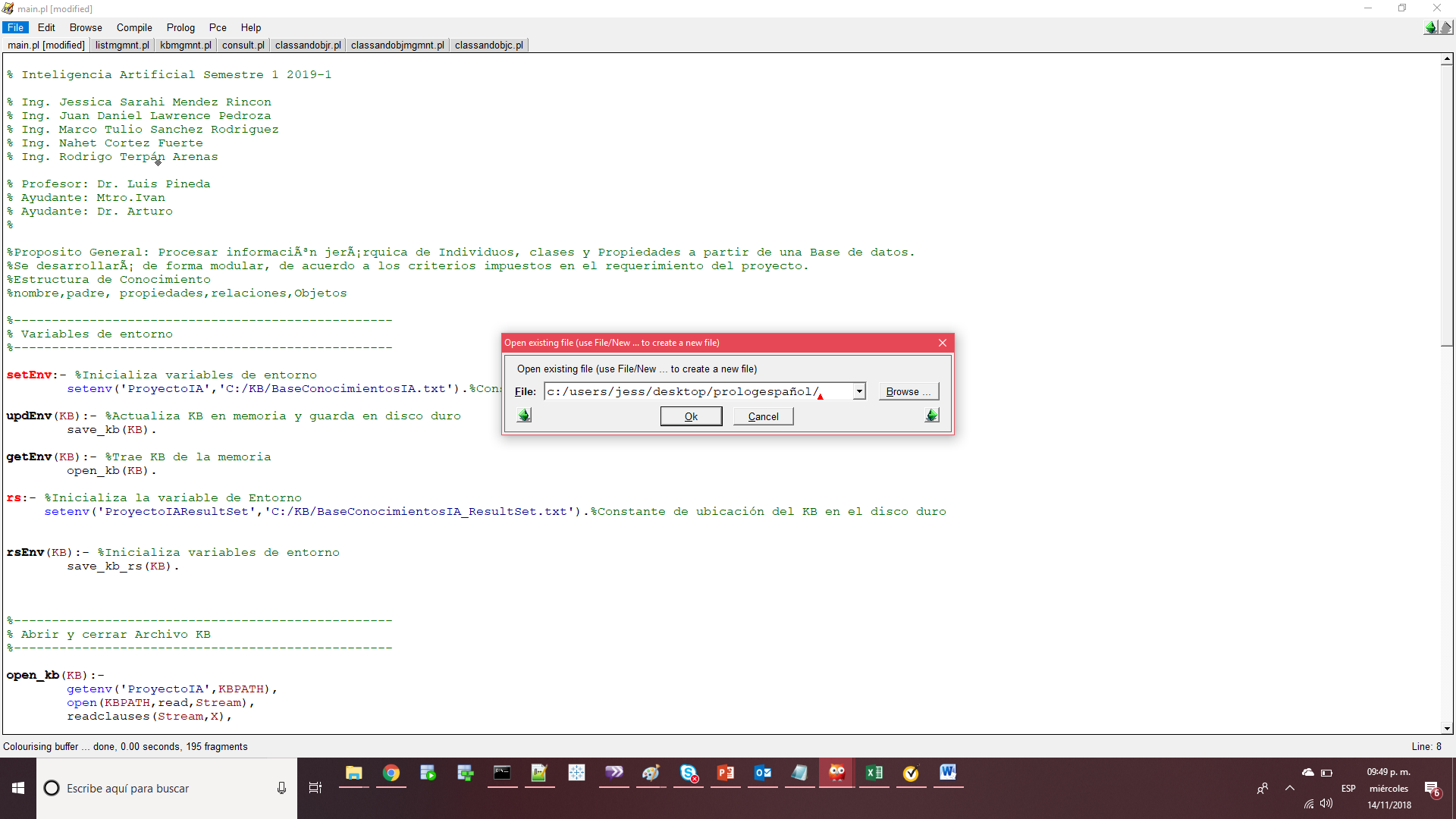


En caso de que se desee cambiar la ruta de la base de conocimientos, se necesita ir al archivo de main.pl, y modificar la ruta del archivo, en las líneas que se indican en la siguiente imagen.



Se deberá de abrir cada uno de los archivos de la carpeta en una misma sesión de Prolog.

De tal forma que se puedan contemplar todos los archivos.



Una vez que se abren se compilan, para ello puede ser desde el Editor seleccionando la opción de:

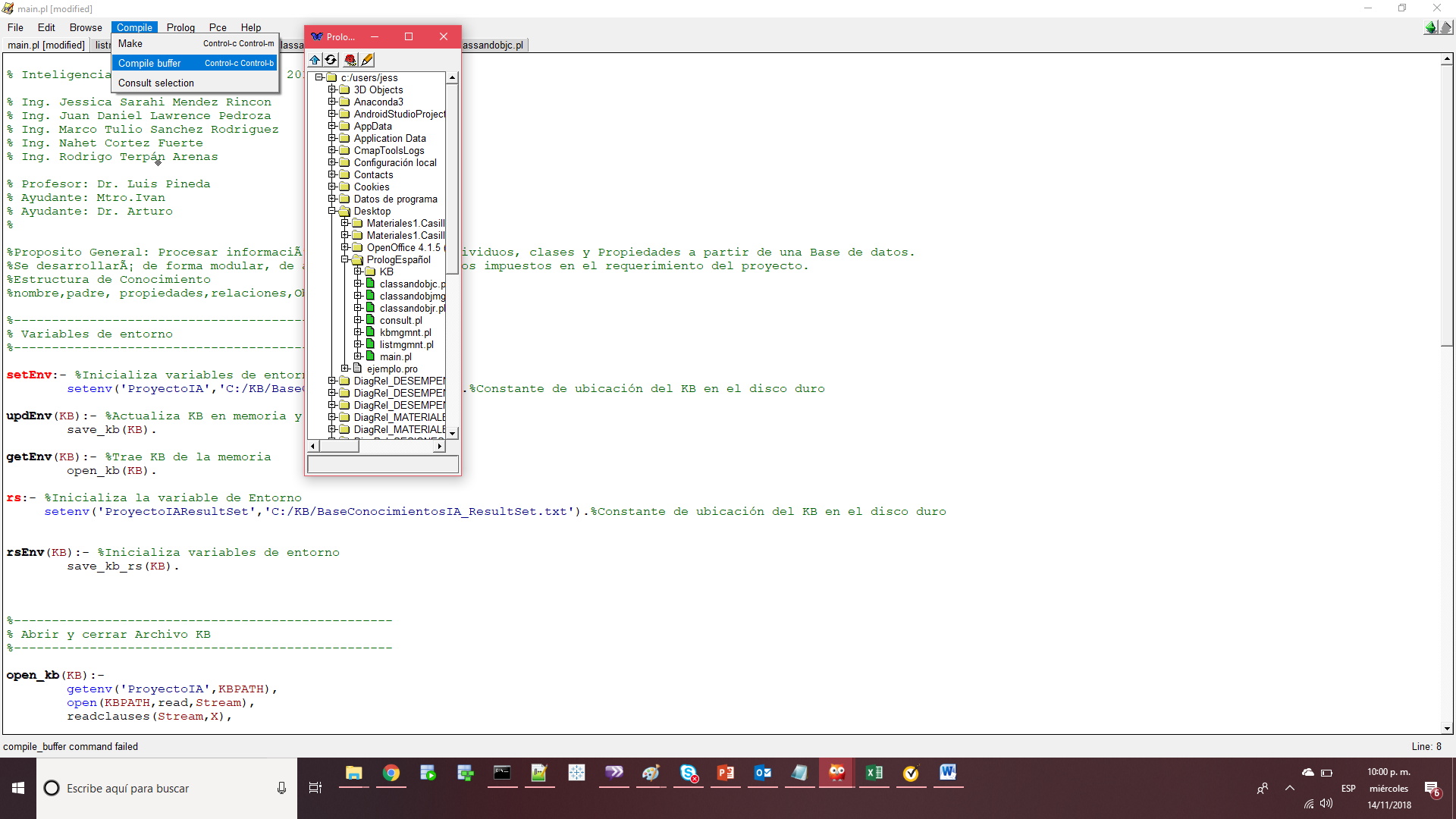
***Compile>Compile Buffer***

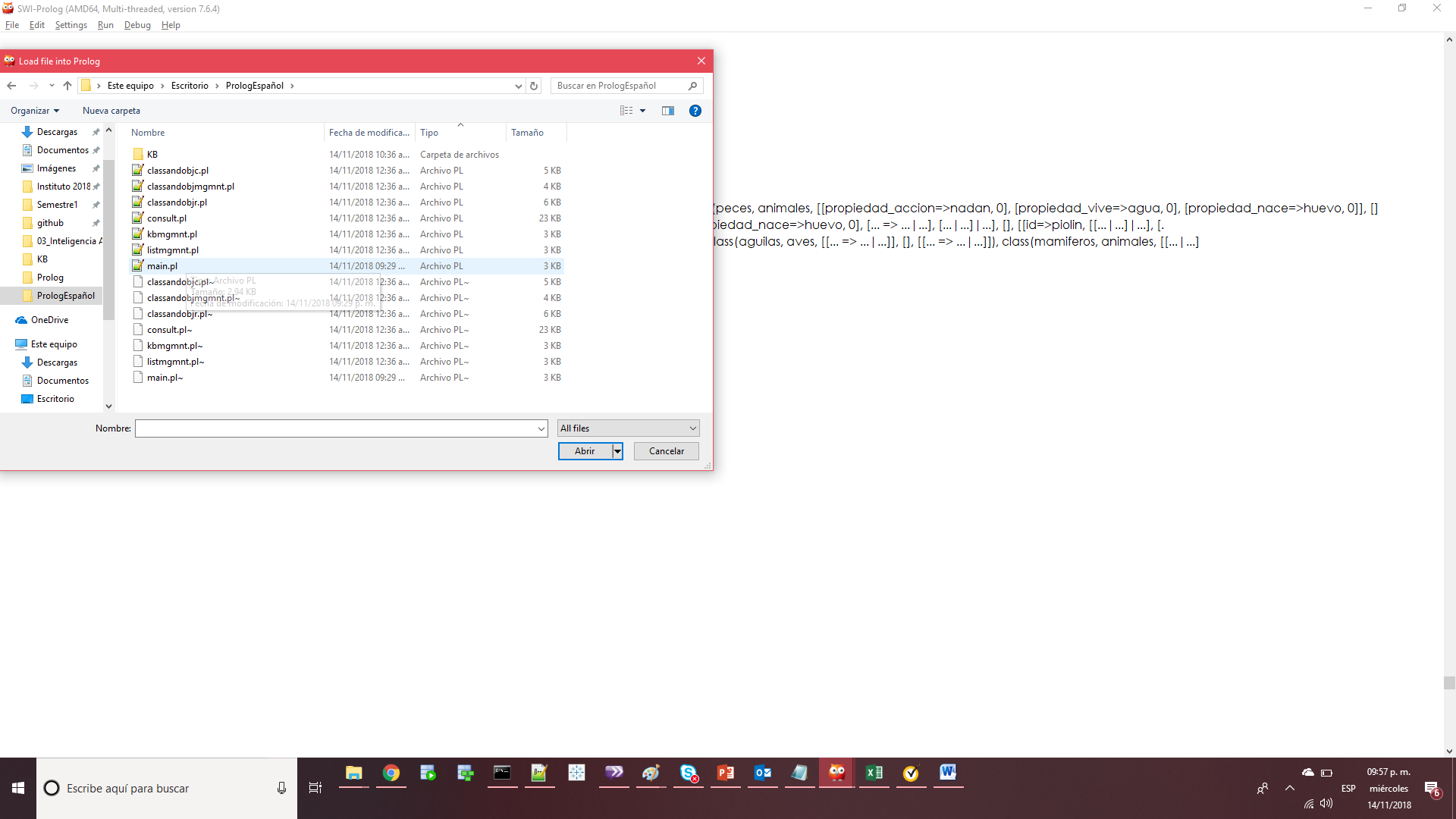
O también en la pantalla de Ejecución de Prolog seleccionando la opción de:

***File>Consult…***

Y buscar el archivo, o tambien una vez que se abrá el archivo se puede aplicar la función

***File>Reload modified files***





# Módulos del Sistema

A continuación se dará una descripción de las funciones de los módulos así como un ejemplo de ejecución.

# Módulo de Consulta

# Consulta de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Consulta de Propiedades de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Consulta de Preferencias de Propiedades de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Consulta de Relaciones de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Consulta de Preferencias de Relaciones de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Consulta de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Consulta de Propiedades de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Consulta de Preferencias de Propiedades de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Consulta de Relaciones de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Consulta de Preferencias de Relaciones de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Módulo de Añadir

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Propiedades de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Preferencias de Propiedades de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Relaciones de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Preferencias de Relaciones de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Propiedades de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Preferencias de Propiedades de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Relaciones de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Añadir de Preferencias de Relaciones de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Módulo de Eliminar

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Propiedades de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Preferencias de Propiedades de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Relaciones de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Preferencias de Relaciones de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Propiedades de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Preferencias de Propiedades de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Relaciones de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Eliminar de Preferencias de Relaciones de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Módulo de Modificar

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Propiedades de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Preferencias de Propiedades de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Relaciones de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Preferencias de Relaciones de Clases

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Propiedades de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Preferencias de Propiedades de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Relaciones de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |

# Modificar de Preferencias de Relaciones de Objetos

|  |  |
| --- | --- |
| Función | Descripción |
| clasesDeObjeto(Objeto,KB,ListaClases):-  verificarObjeto(Objeto,KB,yes),  obtenerHerencia(Objeto,KB,X),  obtenerAntecesores(X,KB,Y),  append([X],Y,ListaClases).  clasesDeObjeto(\_,\_,'no lo se'). | Función que muestra todas las clases a las que pertenece un objeto.  VerificarObjeto => verifica si existe el objeto en la base de conocimiento  obtenerHerencia=> verifica el nombre de la clase del objeto, el primer relacionado al objeto es decir, la clase padre  obtenerAntecesores => extrae las clases padre del objeto  Append => agrega los nombres a una lista para el result set. |
| Ejemplo | |
|  | |