Tutorial MessengerAmigos

A continuación se proporcionan algunas indicaciones para la creación de un nuevo verificador en la aplicación Messenge. Durante la primera sección se mostrará como agregar un nuevo autómata a la aplicación, mientras que en la segunda sección se mostrará como agregar un nuevo parser.

1. Agregando un nuevo autómata

1. En el paquete "uniandes.cupi2.messengerAmigos.verificador" de la aplicación, existe una interfaz denominada "IVerificadorMensajes.java" (ver Figura 1), abra este archivo. En esta interfaz se encuentran los nombres asociados a cada uno de los verificadores: ya sea un Parser, un autómata, etc. Allí existen dos extensiones que permiten agregar nuevos verificadores a la aplicación: Extension 1 y Extension 2. Se sustituirá el nombre de la extensión 1 por "Nuevo Autómata", con la finalidad de agregar un nuevo autómata a la aplicación (ver Figura 2)

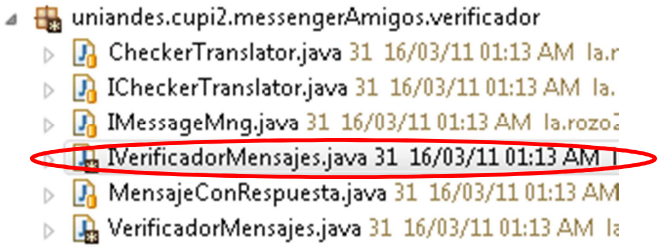

Figura 1. Paquete "uniandes.cupi2.messengerAmigos.verificador"

```
package uniandes.cupi2.messengerAmigos.verificador;

public interface IVerificadorMensajes {
    static final   String  names [] ={"Nada", "Auto Frase Aceptable",
        "* Auto Frase Aceptable(1)",  "ParserDR", "PArser JavaCC", "CAML",
       "Nuevo Automata", "Extension 2"} ;

}
```

Figura 2. Interfaz IVerificadorMensajes.java

2. El siguiente paso consiste en abrir el archivo denominado "VerficadorMensajes.java", que también se encuentra en el paquete "uniandes.cupi2.messengerAmigos.verificador". Al abrir esta clase, se observa que al inicio aparece una declaración de un arreglo de tipo "CheckerTranslator" llamado "checks", a través del cual son creados todos los verificadores empleados en la aplicación. Se modificará la penúltima posición del arreglo de verificadores (correspondiente a la extensión 1), enviando como parámetro una instancia del nuevo autómata (ver Figura 3)

```
public class VerificadorMensajes implements ICheckerTranslator, IVerificadorMensajes {


private CheckerTranslator checks [] = {
        new CheckerTranslator(),
        new CheckerTranslator(new FraseAceptable(0)),
        new CheckerTranslator(new FraseAceptable(1),new FraseAceptable(-1)),
        new CheckerTranslator(new Parser(),CheckOnly),
        new CheckerTranslator(new SimpleParser(System.in),CheckOnly),
        new CheckerTranslator(new ParserCAML(System.in),Code),

        /* extensiones actualmente asociadas a el verificador que no revisa ni traduce */
        /*  Al agregarlos acá deben agregar los nombres en IVerificadorMensajes */

        new CheckerTranslator(new NuevoAutomata()),

        new CheckerTranslator());
```

Figura 3. Clase "VerificadorMensajes"

3. Existen varias maneras de crear un nuevo verificador mediante la clase "CheckerTranslator". Se puede crear un verificador que simplemente verifique la estructura de la cadena de entrada, caso en el cual se especifica la opción "Copy" en el segundo parámetro del constructor de CheckerTranslator, este es el caso del primer verificador del arreglo "checks" que, a pesar de que no especifica el parámetro "Copy" al constructor, este se lo pone por defecto (si no se especifica algo en el constructor, ver Figura 4. Creación de los diferentes tipos de verificadores mediante el constructor de la clase "CheckerTranslator". a) Verificador que no revisa ni traduce; b) Verificador de revisión ("CheckOnly"); c) Verificador de codificación y decodificación; d) Verificador de revisión ("CheckOnly"); e) Verificador de codificación. a). Este también es el caso del nuevo autómata introducido a la aplicación.
   También se puede crear un verificador cuyo objetivo solo sea el de verificar la frase introducida por el usuario, es decir, de tipo "CheckOnly", para lo cual se introduce un único parámetro al constructor: la clase encargada de hacer la verificación; este es el caso del verificador que aparece en la posición dos del arreglo de verificadores, en este caso, la clase encargada de hacer la verificación es "FraseAceptable" (ver Figura 4. Creación de los diferentes tipos de verificadores mediante el constructor de la clase "CheckerTranslator". a) Verificador que no revisa ni traduce; b) Verificador de revisión ("CheckOnly"); c) Verificador de codificación y decodificación; d) Verificador de revisión ("CheckOnly"); e) Verificador de codificación b). Otra forma de crear un verificador de este tipo es introduciendo dos parámetros al constructor: la clase encargada de hacer la verificación y el tipo de verificador, que corresponde a un entero (ver Figura 4. Creación de los diferentes tipos de verificadores mediante el constructor de la clase "CheckerTranslator". a) Verificador que no revisa ni traduce; b) Verificador de revisión ("CheckOnly"); c) Verificador de codificación y decodificación; d) Verificador de revisión ("CheckOnly"); e) Verificador de codificación, en la que la clase encargada de hacer la verificación corresponde a "Parser"). Las clases introducidas como parámetro para hacer la verificación deben implementar la interfaz "IMessageMng", que se encuentra en el paquete "uniandes.cupi2.messengerAmigos.verificador", de tal manera que sean aceptadas por el constructor de la clase "CheckerTranslator"
   Otro tipo de verificador es aquel compuesto de un codificador y un decodificador, este es el caso del tercer verificador de la lista de verificadores "checks" (ver Figura 4. Creación de los diferentes tipos

de verificadores mediante el constructor de la clase “CheckerTranslator”. a) Verificador que no revisa ni traduce; b) Verificador de revisión (“CheckOnly”); c) Verificador de codificación y decodificación; d) Verificador de revisión (“CheckOnly”); e) Verificador de codificaciónc); para construir un verificador de este tipo a partir del constructor de la clase “CheckerTranslator”, se introducen dos parámetros: el codificador (que en la figura corresponde a la clase “FraseAceptable(1)”) y el decodificador (que en la figura es la clase “FraseAceptable(-1)”). Tanto el codificador como el decodificador introducidos como parámetros en el constructor, deben implementar la interfaz “IMessageMng” para que sean aceptados por la función

Otro caso sería la creación de un verificador cuyo objetivo sea el de codificar únicamente, en este caso, se introducen dos parámetros al constructor de la clase “CheckerTranslator”: la clase encargada de hacer la codificación y un parámetro de tipo entero que indicará el tipo de verificador que se va a crear, en este caso “Code”. En la Figura 4. Creación de los diferentes tipos de verificadores mediante el constructor de la clase “CheckerTranslator”. a) Verificador que no revisa ni traduce; b) Verificador de revisión (“CheckOnly”); c) Verificador de codificación y decodificación; d) Verificador de revisión (“CheckOnly”); e) Verificador de codificacióne se muestra un ejemplo en el que se construye un verificador de este tipo, la clase “ParserCAML” será la encargada en este caso de realizar la codificación

a.
```
new CheckerTranslator()
```
b.
```
new CheckerTranslator(new FraseAceptable(0))
```
c.
```
new CheckerTranslator(new FraseAceptable(1),new FraseAceptable(-1))
```
d.
```
new CheckerTranslator(new Parser(),CheckOnly)
```
e.
```
new CheckerTranslator(new ParserCAML(System.in),Code)
```

Figura 4. Creación de los diferentes tipos de verificadores mediante el constructor de la clase “CheckerTranslator”. a) Verificador que no revisa ni traduce; b) Verificador de revisión (“CheckOnly”); c) Verificador de codificación y decodificación; d) Verificador de revisión (“CheckOnly”); e) Verificador de codificación

4. El siguiente paso consiste en crear el nuevo autómata en el paquete “uniandes.cupi2.messengerAmigos.verificador.mundoAutomata” (ver Figura 5). El nuevo autómata debe extender la clase “Automata”, proporcionada como base para la creación de autómatas, y debe tener definido, como mínimo, uno o más estados finales, un estado inicial, cero o más estados intermedios, una función delta, y una función denominada “toString”, que retorna un String con el nombre del autómata

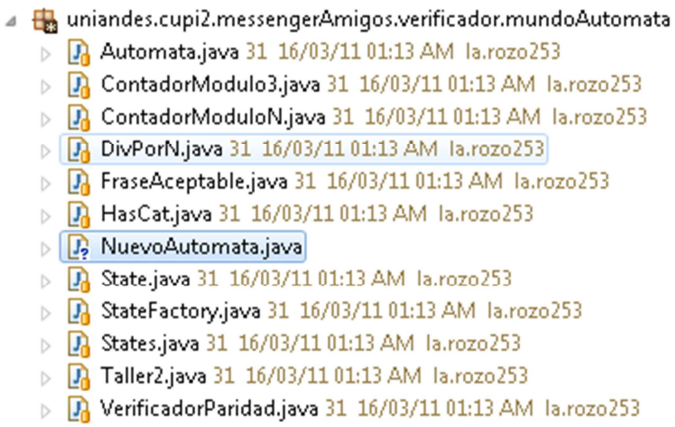

Figura 5. Creación del nuevo autómata en el paquete "uniandes.cupi2.messengerAmigos.verificador.mundoAutomata"

5. Una vez son seguidos los pasos anteriores, corremos la aplicación, para lo cual damos click derecho sobre la carpeta que contiene el proyecto, elegimos la opción "Run As", "Java Application" (ver Figura 6). En la ventana emergente, seleccionamos la interfaz asociada al servidor (ver Figura 7), de esta manera ya podremos correr la interfaz asociada al cliente

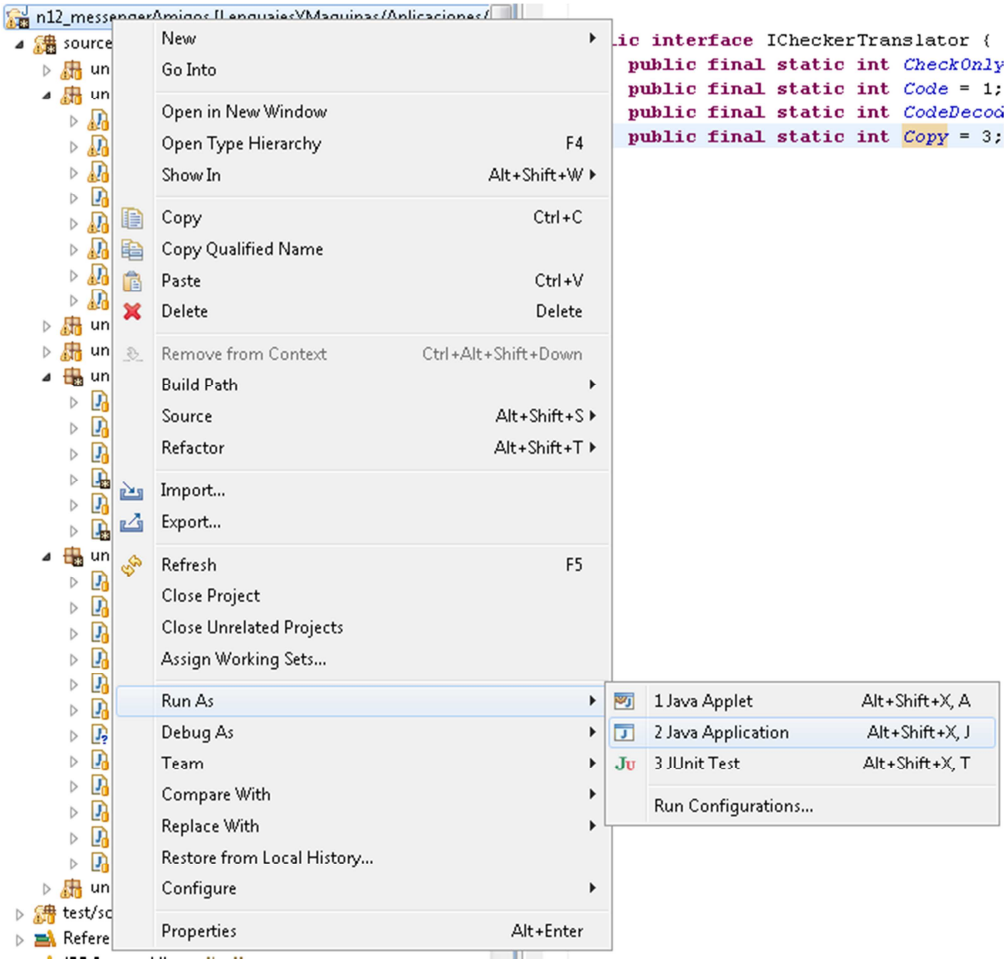

Figura 6. Instrucciones para correr la aplicación MessengerAmigos

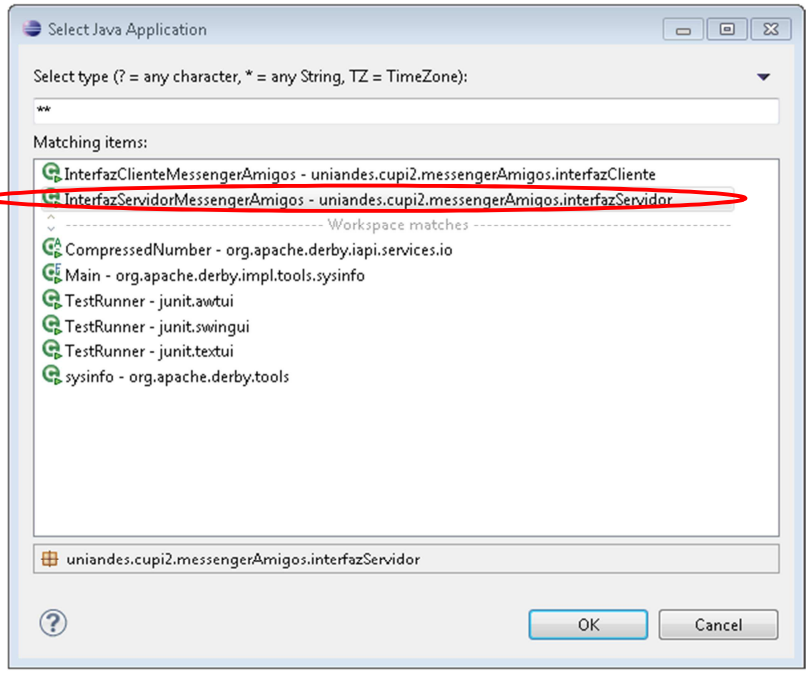

Figura 7. Selección de la interfaz asociada al servidor de la aplicación MessengerAmigos

6. El siguiente paso consisten en correr la aplicación asociada al cilente, para lo cual damos nuevamente click derecho sobre la carpeta que contiene el proyecto, y elegimos la opción “Run As”, “Java Application”; y en la ventana emergente seleccionamos la opción asociada al servidor del cliente (ver Figura 8)

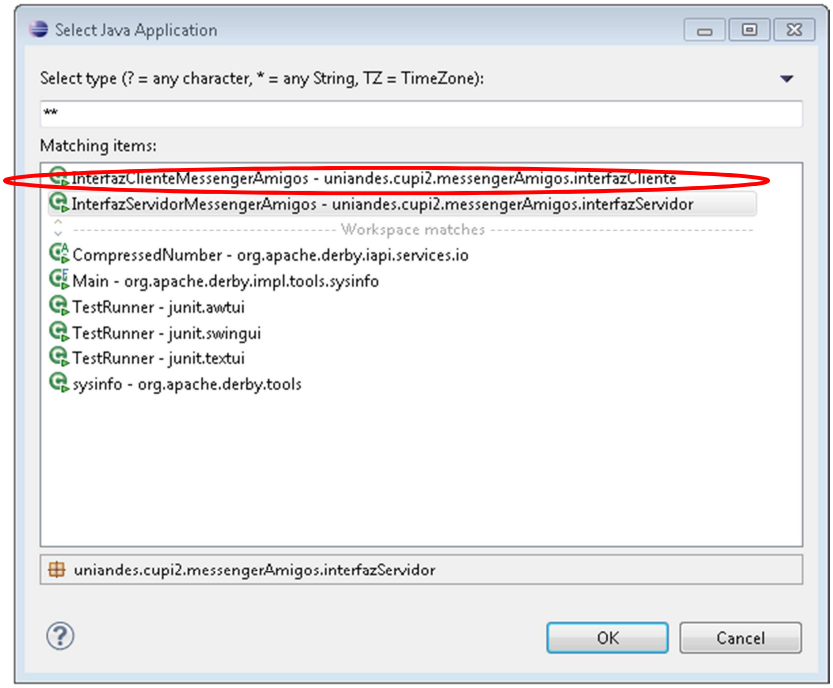

Figura 8. Selección de la interfaz asociada al cliente de la aplicación MessengerAmigos

7. Repetimos el paso anterior para abrir otra ventana, asociada a otro cliente
8. Una vez tenemos abiertas 3 ventanas: una asociada al servidor, y 2 más asociadas a los clientes; procedemos a conectar los cliente. Para lo cual nos dirigimos a una de las interfaces de los clientes, seleccionamos el menú "Messenger", y elegimos la opción "Conectar" (ver Figura 9). Aparecerá una ventana como la mostrada en la Figura 10, en donde se solicitará el nombre de usuario; una vez es introducido el parámetro solicitado, elegimos la opción "Aceptar". Repetimos lo anterior para la interfaz del otro cliente

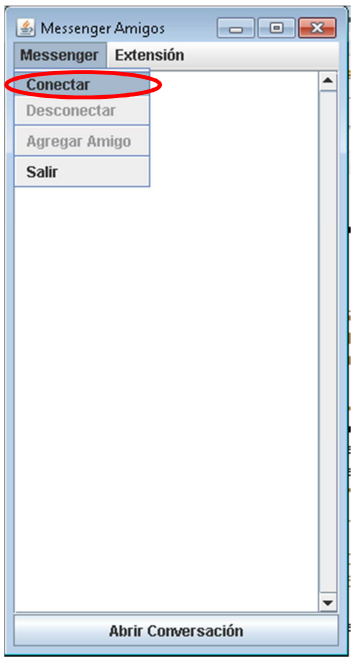

Figura 9. Conectar dos clientes

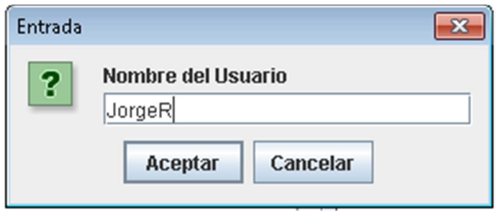

Figura 10. Especificación del nombre del cliente

9. El siguiente paso consiste en agregar un amigo (el otro cliente registrado), para lo cual nos dirigimos al menú "Messenger", y elegimos la opción "Agregar Amigo" (ver Figura 11). Se desplegará una ventana como la mostrada en la Figura 12, en la que se nos solicitará ingresar el nombre del nuevo amigo, que en este caso, será AngélicaR (correspondiente al nombre del otro cliente registrado)

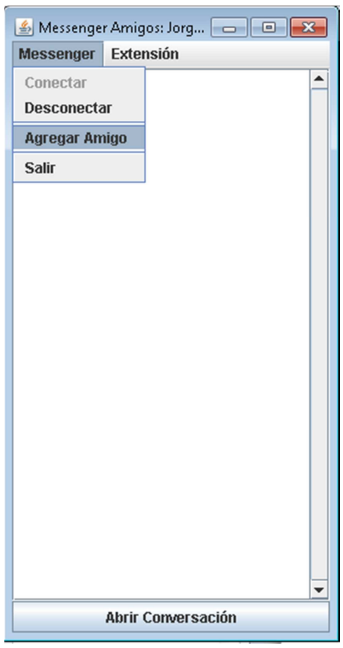

Figura 11. Agregar un amigo desde la interfaz de un cliente

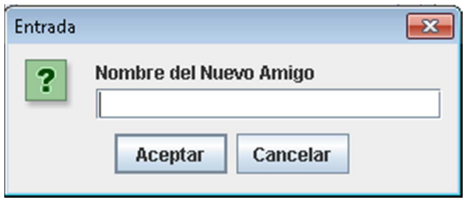

Figura 12. Especificación del nombre de un nuevo amigo

10. Abrimos una nueva conversación. Seleccionamos el nombre del amigo con el que queremos iniciar una nueva conversación y elegimos la opción “Abrir Conversación” (ver Figura 13). Se desplegará una ventana como la mostrada en la Figura 14, con el menú “Revisar”, en donde aparecerán todos los verificadores disponibles en la aplicación (incluido el agregado en este tutorial, “Nuevo Autómata”)

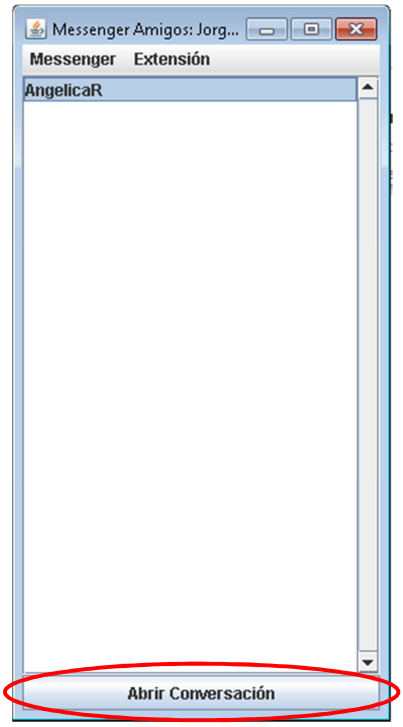

Figura 13. Instrucciones para abrir una nueva conversación

Figura 14. Ventana asociada a una nueva conversación

2. Agregando un nuevo parser

1. Seguir los pasos 1 a 3 de la anterior sección, y en lugar de nombrar el nuevo verificador "Nuevo Automata", nombrarlo como "Nuevo Parser". En la Figura 15 se muestra como instanciar el nuevo parser en el arreglo "checks" de la clase "VerificadorMensajes"

```
import uniandes.cupi2.messengerAmigos.verificador.mundoAutomata.*;

public class VerificadorMensajes implements ICheckerTranslator, IVerificadorMe



private CheckerTranslator checks [] = {
        new CheckerTranslator(),
        new CheckerTranslator(new FraseAceptable(0)),
        new CheckerTranslator(new FraseAceptable(1),new FraseAceptable(-1)),
        new CheckerTranslator(new Parser(),CheckOnly),
        new CheckerTranslator(new SimpleParser(System.in),CheckOnly),
        new CheckerTranslator(new ParserCAML(System.in),Code),

        /* extensiones actualmente asociadas a el verificador que no revisa ni
        /*  Al agregarlos acá deben agregar los nombres en IVerificadorMensaje

        new CheckerTranslator(new NuevoAutomata()),

        new CheckerTranslator(new NuevoParser(System.in), CheckOnly)};
```

Figura 15. Instanciación del nuevo parser en el arreglo checks

2. Crear un nuevo parser. Dar click derecho sobre el paquete "uniandes.cupi2.messengerAmigos.verificador.parsers" y seleccionar la opción "New", "Other" (ver Figura 16). Se desplegará una ventana como la mostrada en la Figura 17, allí elegimos la opción "Java CC Template File" y damos click en "Next"
3. Se especifica el nombre del nuevo parser, que en este caso será "NuevoParser" y se elige la opción "Non Static" (ver Figura 18)
4. Se realizan algunas modificaciones al archivo JavaCC creado en el paso anterior. Primero, se define el nombre del analizador, como "NuevoParser", ya que el nombre establecido por defecto es "eg1", y se borra la función main que aparece en la plantilla inicialmente. La clase NuevoParser, debe implementar la interfaz "IMessageMng", por lo que se debe importar el paquete "uniandes.cupi2.messengerAmigos.verificador.IMessageMng". Esta interfaz se compone de dos funciones: "public void setInput(String m)", la cual se implementará en la definición del parser; y "public Object input() throws Throwable;", la cual será implementada durante la definición sintáctica de la gramática. De esta manera se obtiene algo similar a lo mostrado en la Figura 19 (allí solo se observa una de las funciones de la interfaz "IMessageMng", setInput)
5. A continuación se define la especificación léxica del a gramática, que para el caso de este tutorial se dejará la que viene por defecto en el archivo. Aquí se especifican los tokens que harán parte de la gramática (empleando declaraciones de tipo TOKEN), así como aquellos caracteres que serán filtrados por el analizador léxico, como los espacios y los finales de línea (esto se hace empleando declaraciones de tipo SKIP)

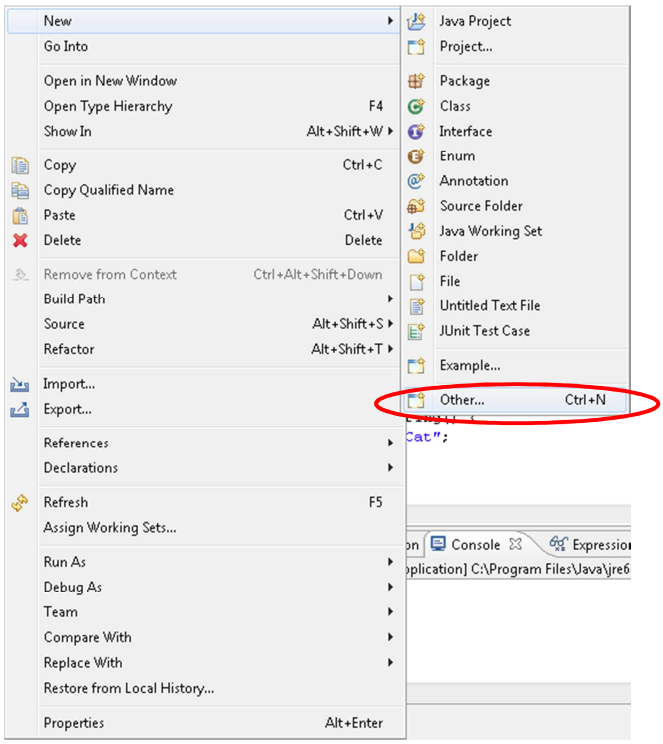

Figura 16. Creación de un nuevo parser

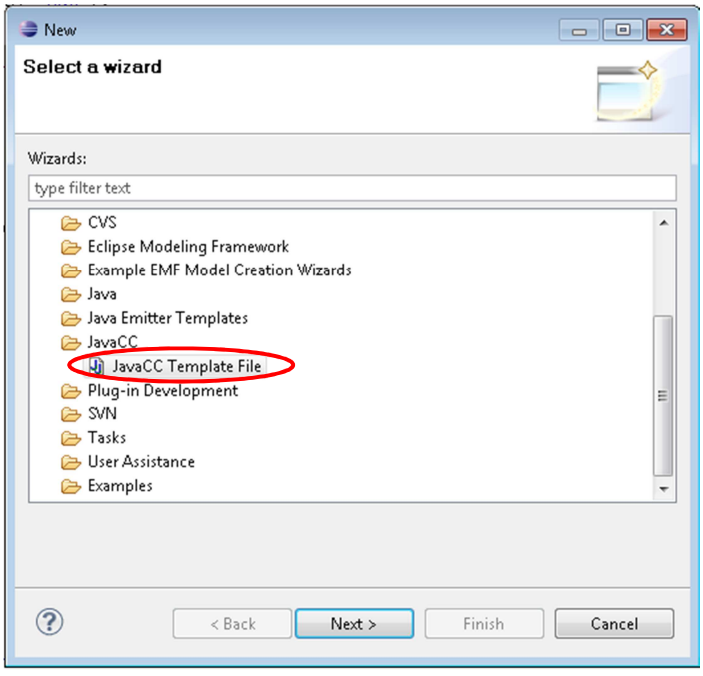

Figura 17. Creación de un nuevo parser empleando el pluggin de JavaCC

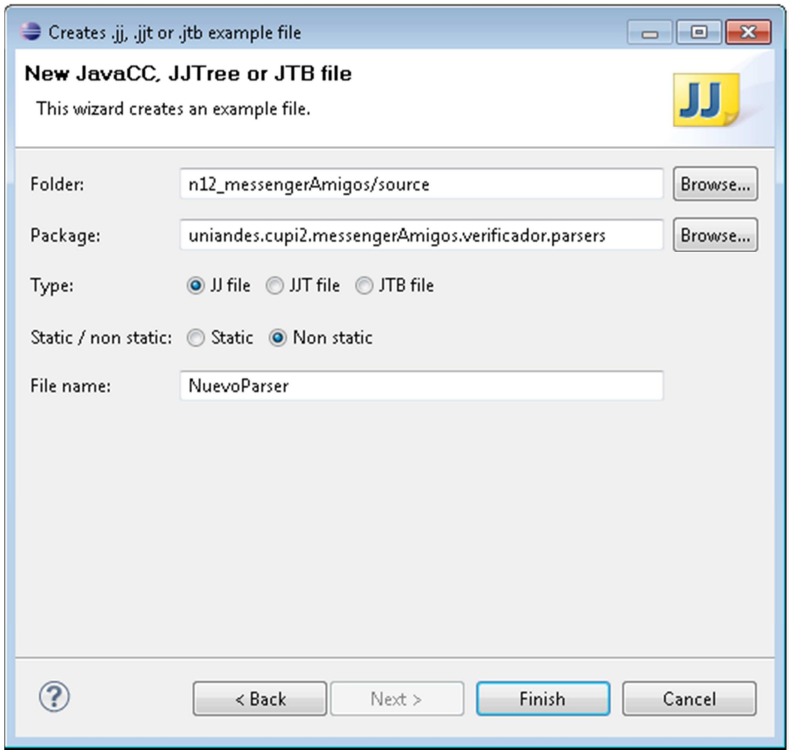

Figura 18. Creación de un nuevo parser empleando el pluggin de JavaCC

```
/**
 * JavaCC template file created by SF JavaCC plugin 1.5.17+ wizard for JavaCC
 */
options
{
  JDK_VERSION = "1.5";
  static = false;
}

PARSER_BEGIN(NuevoParser)
package uniandes.cupi2.messengerAmigos.verificador.parsers;
import uniandes.cupi2.messengerAmigos.verificador.IMessageMng;

public class NuevoParser implements IMessageMng
{
  public void setInput(String input) {
        ReInit(new java.io.StringReader(input));
    }
}

PARSER_END(NuevoParser)
```

Figura 19. Archivo JavaCC asociado al nuevo Parser

6. Se define ahora la especificación sintáctica de la gramática. En esta sección son definidos todos los símbolos no terminales de la gramática (incluido el símbolo distinguido). Para este tutorial, se dejará

la definición por defecto que viene con el archivo JavaCC, agregando una función más: input(), que corresponde a una de las funciones implementadas en la interfaz “IMessageMng”, mencionada en el paso 4, en esta función se especifica el símbolo distinguido de la gramática, que en este caso, debe tener el nombre “input” para satisfacer los requerimientos de la interfaz “IMessageMng” definida en la aplicación (ver Figura 20)

```
String input():
{}
{
 (one_line())+ {return "FRASE ACEPTADA";}
}

int one_line() :
{}
{
  sum() ";"
  {
    return 0;
  }
| ";"
  {
    return 1;
  }
}
```

Figura 20. Definición del símbolo distinguido del nuevo parser, que en este caso, debe tener el nombre “input”

7. Para correr la aplicación con el nuevo parser, siga los pasos 5 a 10 de la anterior sección