Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Curso: Organizacion de Lenguajes y Compiladores 2

Manual Técnico
Proyecto 1: Virtual Machine

Erik Vladimir Girón Márquez
Carnet # 200313492
Marlon Manzo
Carnet # 200313178
21/03/07
Sección B

Introducción

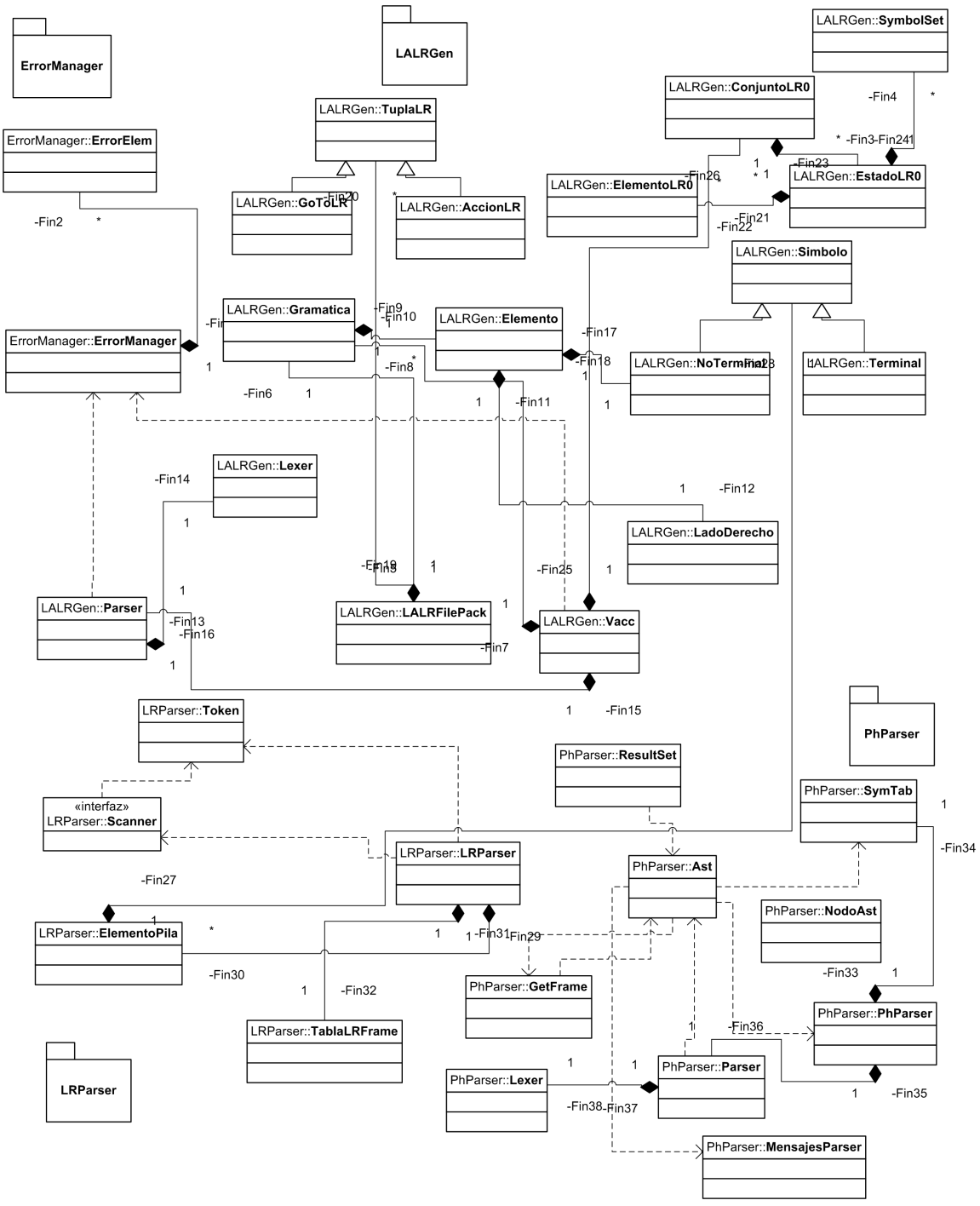
Virtual Machine es una aplicacion de interpretacion de lenguajes tipo 2, que pueden ser resueltos con analisis sintactico libre de contexto. permitiendo al usuario generar paquetes de lenguajes, con acciones asociadas a la gramatica, para que luego, estos sirvan de intérpretes para los archivos de entrada final, como por ejemplo, para realizar un paquete calculadora, se debera escribir la gramatica de que reconozca expresiones aritméticas, para luego generar el paquete, y esta pueda interpretar un archivo de entrada con la expresion aritmetica correspondiente.

El programa implementa un diseño puramente orientado a objetos, que junto con las características de programación por eventos de Java, simplificaron en gran parte el desarrollo visual e interactivo de la aplicación.

Se utilizó entonces para el desarrollo del proyecto, la plataforma Java. Permitiéndose compilar la aplicación bajo *Java JDK* v 1.5 y utilizando *netBeans* para la administración del proyecto, que gracias al sencillo editor de GUI con que cuenta este *IDE*, permitió generar de una manera rápida y segura el código necesario para la manipulación de eventos sobre el UI, tanto en GUI, como en CLI.

En este documento, el programador se podrá guiar por medio de descripcion de los paquetes a través del diseño del corazon del proyecto y de cada una de las clases que la componen, contando con descripciones de cada clase. Se recomienda que para ampliar la información dada aquí, se consulte el manual *JavaDoc* generado desde la documentación del código de la aplicación, y localizado en la distribución oficial, además de revisar el código fuente si se tiene alguna duda sobre la implantación del código.

Diseño interno generalizado:



Jerarquía de clases:

java.lang.Object

- phparser.**Ast**
- java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - java.awt.Container
        - java.awt.Window (implements javax.accessibility.Accessible)
            - java.awt.Frame (implements java.awt.MenuContainer)
                - javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
                    - phparser.**GetFrame**
- phparser.**MensajesParser**
- lrparser.**TablaLRFrame**
- lalrgen.**ConjuntoLR0**
- lalrgen.**Elemento** (implements java.util.Comparator<T>, java.io.Serializable)
- lalrgen.**ElementoLR0**
- lrparser.**ElementoPila**
- errormanager.**ErrorElem** (implements java.io.Serializable)
- errormanager.**ErrorManager**
- lalrgen.**EstadoLR0**
- lalrgen.**Gramatica** (implements java.io.Serializable)
- lalrgen.**LadoDerecho** (implements java.util.Comparator<T>, java.io.Serializable)
- lalrgen.**LALRFilePack** (implements java.io.Serializable)
- lalrgen.**Lexer** (implements java_cup.runtime.Scanner)
- phparser.**Lexer** (implements java_cup.runtime.Scanner)
- java_cup.runtime.lr_parser
    - lalrgen.**parser**
- phparser.**parser**

- lrparser.**LRParser**
- vmw.**Main**
- phparser.**NodoAst**
- phparser.**PhParser**
- phparser.**ResultSet**
- lalrgen.**Simbolo** (implements java.util.Comparator<T>, java.io.Serializable)
    - lalrgen.**NoTerminal** (implements java.io.Serializable)
- lalrgen.**Terminal** (implements java.io.Serializable)
- lalrgen.**sym**
- phparser.**sym**
- lalrgen.**SymbolSet**
- phparser.**SymTab**
- phparser.**SymTab.SymTabEntry**
- lrparser.**TestLexer** (implements lrparser.Scanner)
- lrparser.**Token**
- lalrgen.**TuplaLR** (implements java.io.Serializable)
    - lalrgen.**AccionLR** (implements java.io.Serializable)
- lalrgen.**GoToLR** (implements java.io.Serializable)
- lalrgen.**Vacc**
- vmw.**Vmw**

Descripcion de clases por paquete:

A continuacion se presenta una descripcion general de cada clase ordenada por paquetes, Para mayor informacion consultar los documentos JavaDoc incluidos

Package errormanager

Class Summary	
ErrorElem	elemento que describe a un error de lectura
ErrorManager	Manejador estatico de errores lexicos, sintacticos, semanticos.

Package lalrgen

Class Summary	
AccionLR	Clase derivada de TuplaLr que representa una accion en la tabla accion[] de los analizadores LR
ConjuntoLR0	Representa un conjunto de estados LR0, Incluye funciones para la creacion y manipulacion de estos.
Elemento	Representa una produccion sencilla (sin operador "|") de una sola linea
ElementoLR0	Provee una abstraccion de un elemento LR0 que contiene una produccion y su respectivo puntito
EstadoLR0	Representa un estado LR0 del conjunto de estados LR0 necesarios para generar las acciones y los lookaheads
GoToLR	Clase derivada de TuplaLr que representa una transicion de estados en la tabla goto[] de los analizadores LR
Gramatica	Representa una gramatica libre de contexto.
LadoDerecho	Provee abstraccion al lado derecho de una gramatica libre de contexto
LALRFilePack	Clase que sirve como contenedor de los datos necesarios para serializar una tabla LALR dada.
Lexer	
NoTerminal	Representacion de un no terminal de la gramatica dada.
parser	CUP v0.10k generated parser.
Simbolo	Superclase que representa los simbolos gramaticales.
sym	CUP generated class containing symbol constants.
SymbolSet	Clase base para el conjunto de elementos primero y siguiente.
Terminal	Representa un terminal en una gramatica.
TuplaLR	Representa una celda de la tabla LR
Vacc	Clase principal para el generador de analizadores sintacticos, contiene todos los metodos utiles para su utilizacion

Package lrparser

Interface Summary	
Scanner	Interface Scanner Declara el metodo next_token() que debe ser implementado por los scanners, el fin de archivo(EOF) puede ser indicada ya sea retornanto * `new Token($EOF$)` o `null`.

Class Summary	
ElementoPila	Define un elemento que se va a ingresar a la pila de analisis sintactico
LRParser	Clase que carga e interpreta un paquete que contiene una tabla LALR generada previamente, utilizando un scanner que use la interfaz Scanner y que tenga implementada la funcion next_token() que devuelva

	un token, para poder realizar un analisis sintactico LR.
TablaLRFrame	Ventana que muestra la tabla LALR dada por un paquete.
TestLexer	
Token	Clase que define a los tokens retornados por el lexer

Package phparser

Class Summary	
Ast	Clase que representa un arbol de sintaxis abstracta para la interpretacion dinamica del lenguaje dado
GetFrame	Ventana de obtencion de valor para la funcion get
Lexer	
MensajesParser	Ventanita de obtencion de valor de variable para la instruccion get
NodoAst	Representa un nodo abstracto de la cual se derivaran todos los diferentes tipos de nodos nodos del AST
parser	CUP v0.10k generated parser.
PhParser	Punto de entrada para el parsing del codigo en PHP
ResultSet	Simboliza un resultado devuelto por un recorrido hecho en un AST, y es usado para intercambios
sym	CUP generated class containing symbol constants.
SymTab	Tabla de simbolos a utilizar durante la interpretacion