فاز سوم پروژه :

هدف از این قسمت پروژه درست کردن یک درخت است (ast) که درواقع همان درخت تحلیل از کد است و که از برگ ها شروع میشود و به ریشه درخت میرسد که همان botton-up parsing است که بتوان با استفاده از ان کد را به صورت یک درخت دراورد و ان را انالیز کرد.

حال قسمت هایی از کد با توضیح ان ها :‌

feature\_list

:

{ $$ = nil\_Features(); }

| feature

{ $$ = single\_Features($1);}

| feature\_list feature

{ $$ = append\_Features($1,single\_Features($2));}

|feature\_list error

{ $$ = NULL; }

;

feature

: OBJECTID '(' formal\_list ')' ':' TYPEID '{' expr '}' ';'

{ $$ = method($1, $3, $6, $8); }

| OBJECTID ':' TYPEID ASSIGN expr ';'

{ $$ = attr($1,$3,$5); }

| OBJECTID ':' TYPEID ';'

{ $$ = attr($1,$3,no\_expr() ); }

| error ';'

{ $$ = NULL; };

درخت توابع و مجموعه ای از ان ها که در یک کلاس می ایند در بالا درست شده است که در ابتدا مجموعه ای از توابع همان یک تابع است حال اگر در کنار مجموعه توابع تابع دیگری بیاید به مجموعه توابع قبلی اضافه میشود و اگر بعد از ان ها تابعی با ارور بیاید برای اینکه کل برنامه متوقف نشود همه ان ها از درخت پاک میشوند و نادیده گرفته میشوند و ارور گزارش میشود .

حال خود تابع مشخصات خاصی دارد و میتواند فقط به چند صورت خاص بیاد که همه ان ها در بالا در نظر گرفته شده و اگر درست بود و مطابقت با یکی از حالات داشت پس به وصیله کانستراکتور های مخصوص ان یک راس ایجاد میکنیم که فرزندان ان مشخصات ان تابع باشد و اگر مطابقت نداشت پس ارور است و ان ارور را گزارش میدهیم و ان تابع راس را قرار میدهیم.

let\_expr

: OBJECTID ':' TYPEID ASSIGN expr IN expr

{ $$ = let($1,$3,$5,$7); }

| OBJECTID ':' TYPEID ASSIGN expr ',' let\_expr

{ $$ = let($1,$3,$5,$7); }

| OBJECTID ':' TYPEID IN expr

{ $$ = let($1,$3,no\_expr(),$5); }

| OBJECTID ':' TYPEID ',' let\_expr

{ $$ = let($1,$3,no\_expr(),$5); }

| OBJECTID error IN expr

{ $$ = $4; }

| OBJECTID error ',' let\_expr

{ $$ = $4; }

;

Let: میتواند به چند صورت بیاید و مهم ترین انها وقتی است که چند متغیر پشت سر هم تعریف میشوند پس اولین متغیر را دریافت کرده و بقیه ان ها را با اولی با کانستراکتور let یک راس میکنیم که درواقع با اینکار داریم letهای تودرتو و هر کدام با تعریف یک متغیر ایجاد میکنیم و اگر در ادامه با یک ارور برخورد کردیم به متغییر بعدی میرویم و ارور را گزارش میدهیم و از متغیر بعدی ادامه میدهیم.

در good.cl و bad.cl دو کد امده که اولی شامل قسمت هایی به همراه درست شده bad.cl است که درخت را میسازد و در bad.cl مشکلاتی وجود دارد.