مستند پروژه کامپایلر لولو ۲

- مستند فاز دوم پروژه طراحی کامپایلر زبان LULU
- ترم اول سال تحصیلی ۹۸-۱۳۹۷

اعضای گروه:

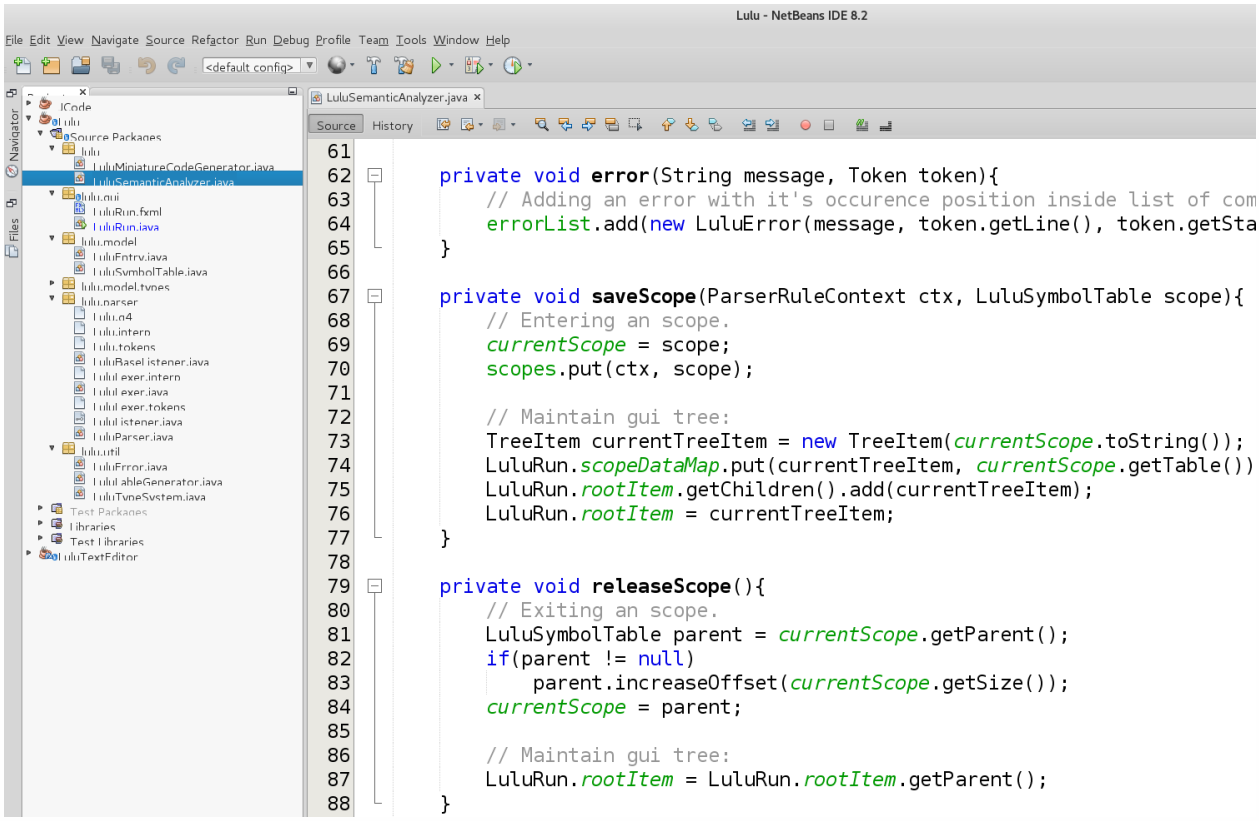
- محسن دهباشی ۹۵۳۶۱۱۱۳۳۰۲۸
- سید محمد هاشمی ۹۵۳۶۱۱۱۳۳۰۸۴
- پوریا زمانی نژاد ۹۵۳۶۱۱۱۳۳۰۳۶

مقدمه

در این فاز از پروژه از API جاوا ابزار $antlr$ برای تجزیه و تحلیل معنایی(شناسایی خطاهای معنایی، آشکارسازی درخت جدول نشانه های برنامه) و تولید کد میانی زبان برنامه نویسی لولو استفاده شد که مراحل استفاده از این ابزار برای رسیدن به این هدف به ترتیب تیتر های درج شده در مستند می باشد.

مرحله اول: بارگیری و نصب

- در این مرحله پس از مطالعه رفرنس، نوع پیمایش درخت از نوع $listener$ انتخاب شد و از ابزار توسعه یکپارچه $netbeans$ جهت توسعه تحلیل گر معنایی از روی کدهای پارسر تولید شده در فاز قبلی پروژه استفاده شد، همچنین در هنگام توسعه از کتابخانه $javafx$ نیز جهت تعریف رابط کاربری گرافیکی استفاده شد.



- یک برنامه *main* جهت یکپارچه سازی عملیات کامپایل:

```
File input = new File("program.lulu");
StringBuilder program = new StringBuilder();
Scanner scan = new Scanner(input);
while(scan.hasNextLine())
    program.append(scan.nextLine()).append("\n");
LuluLexer lexer = new LuluLexer(new ANTLRInputStream(program.toString()
LuluParser parser = new LuluParser(new CommonTokenStream(lexer));
ParserRuleContext programCtx = parser.program();
ParseTreeWalker walker = new ParseTreeWalker();
// LuluSemanticAnalyzer is our extension of LuluBaseListener
LuluSemanticAnalyzer analyzer = new LuluSemanticAnalyzer();
walker.walk(analyzer, programCtx);
```

- لازم بذکر است که از ابتدا این پروژه بر روی گیت هاب تعریف شد و اعضای گروه در تمامی مراحل انجام آن، تغییرات خود بر روی پروژه را از طریق گیت اعمال کردند که آدرس *repository* های مربوطه عبارت اند از:

- https://github.com/mdsinalpha/Lulu-Compiler
- https://github.com/mdsinalpha/Lulu-Text-Editor

مرحله دوم: شناسایی خطاهای معنایی

- در مرحله شناسایی خطا ها، کلاس $LuluSemanticAnalyzer$ وظیفه جمع آوری خطاهای معنایی برنامه را دارد، در صورتی که برنامه ورودی حداقل یک خطای معنایی داشته باشد برای نمونه خطاها به صورت زیر در خروجی چاپ می شوند:

```
Compile error at line 10: Cannot assign a to name. {133|135}
```

- در غیر اینصورت در خروجی پیغام $CompiledSuccessfully$ چاپ می شود.

مرحله سوم: آشکار سازی درخت جدول نشانه ها

- پس از اینکه کامپایلر هیچ خطایی برای یک برنامه ورودی تشخیص نداد، وقت آن است که درخت جدول نشانه و محتویات هر جدول نشانه برنامه آشکار شود، برای اینکار هنگام صدا زدن کامپایلر در `shell` سیستم باید از یک $flag$ استفاده کرد.

```
java -jar Lulu.jar program.lulu ic.asm -reveal
```

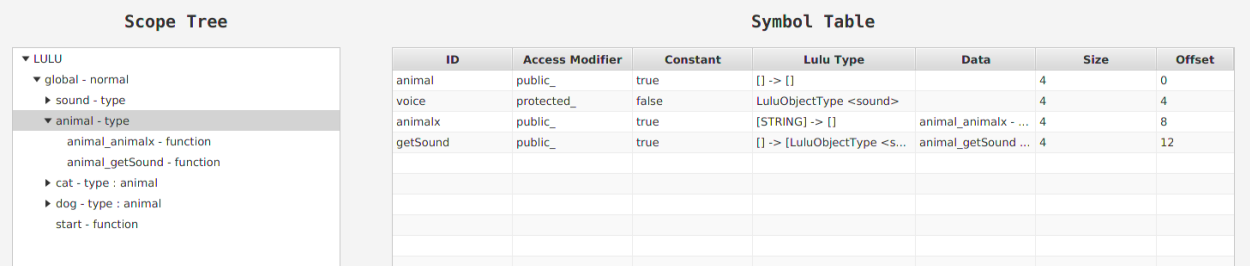
- بعنوان مثال، درخت جدول نشانه های یک برنامه نمونه در ادامه آمده است:

```
declare{
    animal;
    sound;
    cat;
}
type sound{
    public string voiceStr;
    public function soundc(string voiceStr){
        int x = - 2 * 8 % 6 - 1 / (2 -3);
    }
}
type animal{
```

```
    protected sound voice;
    public (sound s) = function getSound () {
        int y = ~ 78 & 1;
    }
}
type cat:animal{
    public function catc(){

    }
}

(int res) = function start(){
    int z = 56 + 89 || 58 - 9;
}
```



مرحله چهارم: تولید کد میانی

- در این مرحله سعی شد تا حدودی کد میانی برنامه لولو را بدست آورد، برای نمونه کد میانی مثال مرحله قبل که کامپایلر لولو قادر به تولید آن است در پایین آمده است:

```
global: sound: sound_soundc: T0 = 2
T1 = - T0
T2 = 8
T3 = T1 * T2
T4 = 6
T5 = T3 % T4
T6 = 1
T7 = 2
T8 = 3
T9 = T7 - T8
T10 = T6 / T9
T11 = T5 - T10
```

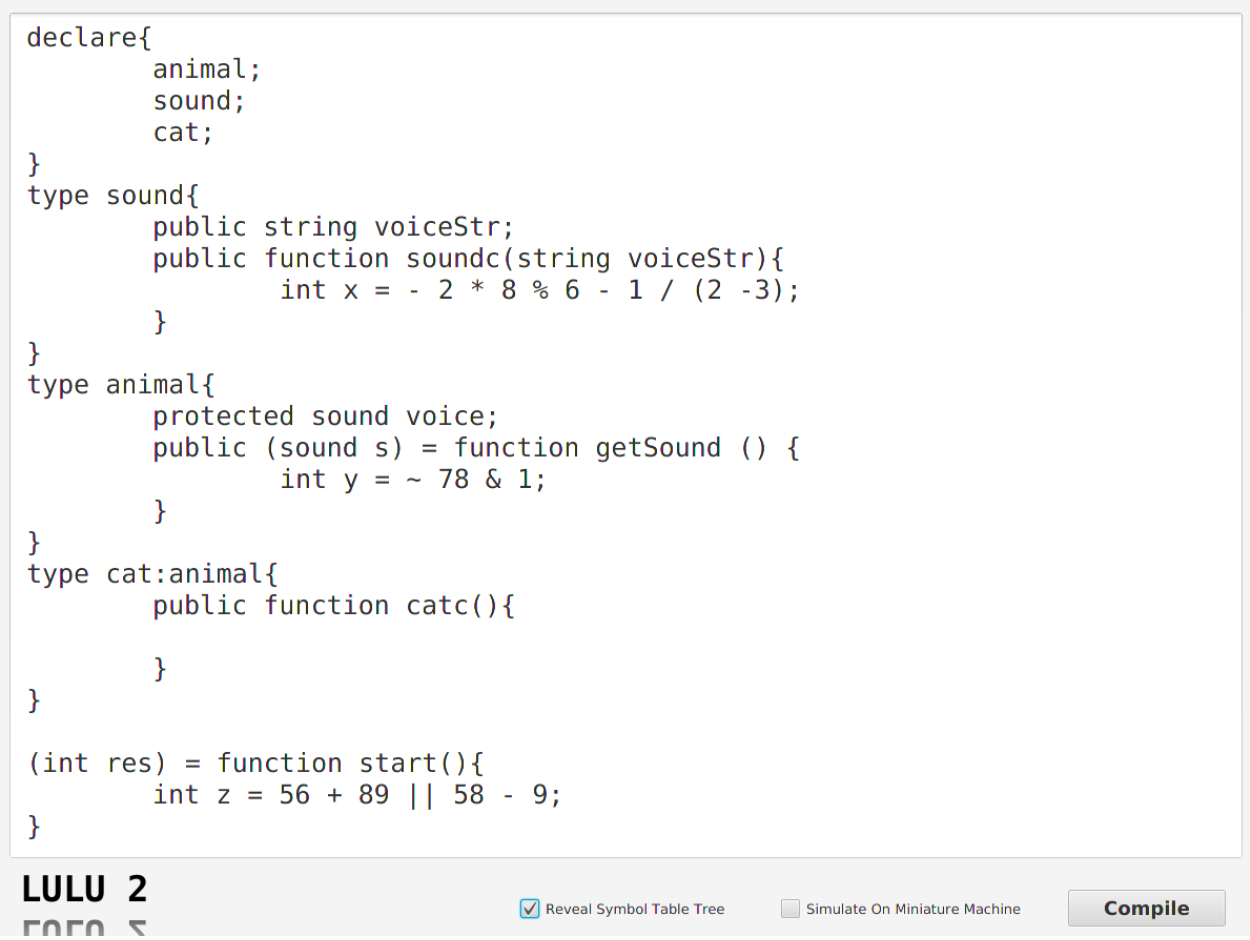
```
animal: animal_getSound: T12 = 78
T13 = ~ T12
T14 = 1
T15 = T13 & T14
cat: cat_catc: start: T16 = 56
T17 = 89
T18 = T16 + T17
T19 = 58
T20 = 9
T21 = T19 - T20
T22 = T18 || T21
```

- این کد هنگام فراخوانی کامپایلر در صورتی که برنامه خطای معنایی نداشته باشد به صورت خودکار ایجاد می‌شود، بنابراین هنگام فراخوانی کامپایلر باید در آرگومان های فراخوانی ابتدا فایل برنامه ورودی و سپس اسم فایل کد میانی خروجی را نیز وارد کنید.

مرحله پنجم: آزمون درستی

- در این مرحله برای دیباگ کردن کامپایلر ۶ برنامه نوشته شد که در پوشه $luluTest$ پروژه موجود می باشند.

مرحله ششم: ویرایشگر متن



```
declare{
        animal;
        sound;
        cat;
}
type sound{
        public string voiceStr;
        public function soundc(string voiceStr){
                int x = - 2 * 8 % 6 - 1 / (2 -3);
        }
}
type animal{
        protected sound voice;
        public (sound s) = function getSound () {
                int y = ~ 78 & 1;
        }
}
type cat:animal{
        public function catc(){

        }
}

(int res) = function start(){
        int z = 56 + 89 || 58 - 9;
}
```

LULU 2

Reveal Symbol Table Tree Simulate On Miniature Machine Compile

روند کار گروهی

- تقسیم کار در کل بصورتی بود که همه اعضای گروه در روند پیاده سازی پروژه حضور فعال داشتند.

تعهدنامه اخلاقی

ما(محسن دهباشی، سید محمد هاشمی، پوریا زمانی نژاد) تعهد می نماییم که پروژه تحویل داده شده نتیجه کار گروهی ما بوده و در هیچ یک از بخش های انجام شده از کار دیگران کپی برداری نشده است. در صورتی که مشخص شود که پروژه تحویل داده شده کار این گروه نبوده است، طبق ظوابط آموزشی با ما برخورد شده و حق اعتراض نخواهیم داشت.