دانشکده مهندسی کامپیوتر

گزارش پروژه درس کامپایلر

استاد درس:
دکتر پارسا

اعضای گروه:
مهدیه نادری
هدیه اسحقی
سارینا شیبانی
محمد عرفان زارع
حسنا کاظمیان

چکیده:

ما برای انجام این پروژه ۳ تابع اصلی را در main برنامه خود نوشتیم که هر یک از آنها یک فاز از پروژه را در بر میگیرند:

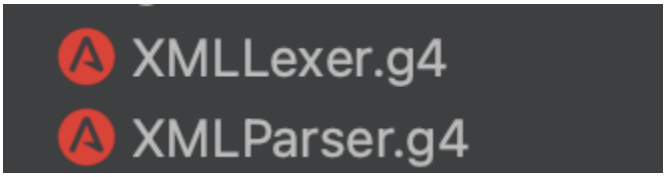
```
def getQFontComboBoxList(input_address):...

def getAstList(QFONTBOXLIST):...

def codeGenerator(ASTLIST):...
```

سپس در main برنامه ابتدا آدرس فایل xml را به تابع getQFontComboBoxList دادیم و parsetree را دریافت کردیم. پس از آن parsetree را به getAstList دادیم و AST را دریافت کردیم و در نهایت AST را به تابع codeGenerator دادیم و کد پایتون ساخته شده را در یک فایل خروجی ذخیره کردیم:

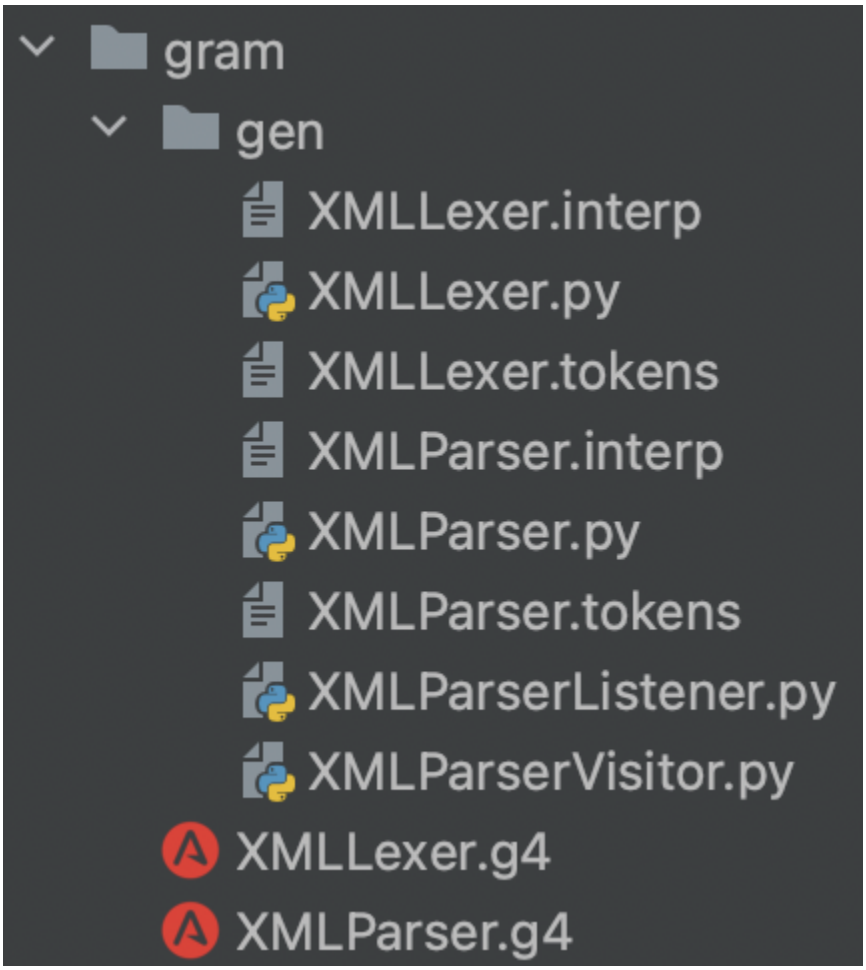
```
if __name__ == "__main__":
    QFONTBOXLIST=getQFontComboBoxList("Myxml.xml")
    ASTLIST = getAstList(QFONTBOXLIST)
    final = codeGenerator(ASTLIST)
```

فاز اول پروژه:

ابتدا فایل های XMLLexer.g4 و XMLParser.g4 را از اینترنت دانلود کردیم:



سپس آنها را generate کردیم و فایلهای زیر از روی آن ساخته شدند:



سپس اقدام به تکمیل تابع اول(getQFontComboBoxList) کردیم:
ساختار این تابع در تصویر زیر مشخص شده است:

```
def getQFontComboBoxList(input_address):
    stream = FileStream(input_address)
    lexer = XMLLexer(stream)
    tokens = lexer.getAllTokens()
    QFONTBOXLIST = []
    i = 0
    while i < len(tokens):
        currentToken = tokens[i]
        # print(currentToken.text)
        if currentToken.text == "QFontComboBox":
            s = "<QFontComboBox "
            i += 1
            currentToken = tokens[i]
            while currentToken.text != "QFontComboBox":
                s += f" {currentToken.text}"
                i += 1
                currentToken = tokens[i]
            s += f" {currentToken.text}>"
            QFONTBOXLIST.append(s)
        i += 1
    return QFONTBOXLIST
```

همانطور که مشاهده میشود ابتدا از روی آدرس فایل xml توانستیم محتوای آن را به صورت stream بخوانیم. سپس با استفاده از XMLLexer توانستیم lexer را ایجاد کنیم. پس از آن توکن ها را با استفاده از تابع getAllTokens به دست آوردیم. سپس با کمک توکن ها و نوع و تکست آنها توانسیم بخش هایی که مرتبط با ویجت ما بود که همان QFontComboBox میباشد را به صورت درخت های جداگانه ذخیره کنیم و آنها را به عنوان خروجی برگردانیم.

فاز دوم پروژه:

در این بخش اقدام به تکمیل بخش تبدیل parse tree هر یک از اعضای لیست قبلی به AST کردیم:

```
def getAstList(QFONTBOXLIST):
    ASTLIST = []
    for QFONTBOX in QFONTBOXLIST:
        istream = InputStream(QFONTBOX)
        lexer = XMLLexer(istream)
        stream = CommonTokenStream(lexer)
        parser = XMLParser(stream)
        parse_tree = parser.document()
        listener = ast_creator()
        walker = ParseTreeWalker()
        walker.walk(listener, parse_tree)
        ASTLIST.append(listener)
        listener.show_tree()
        print(f"root: {listener.root}, attrs: {listener.attrs}")
    return(ASTLIST)
```

همانطور که در تصویر بالا مشاهده میشود در این بخش یک لیست تعریف کردیم که قرار است ast مربوط به تمام QFontComboBox های داخل فایل xml ما را نگه دارد و در آخر به عنوان خروجی برگرداند.

سپس به ازای تک تک اعضای داخل لیست QFontComboBox ها آن را به صورت stream خواندیم و سپس از روی آن lexer را ایجاد کردیم. پس از آن تمام توکن ها را ایجاد کردیم و از روی آن parser و پس از آن parsetree را ایجاد کردیم. سپس listener خود را ast_creator قرار دادیم(در فایل AST.py) قرار دارد و پارس تری خود را با استفاده از از پیمایش کردیم و پس از هر پیمایش AST را به لیست خود اضافه کردیم و درخت مربوطه را در کنسول چاپ نمودیم.

```
labellabellabel
"Arial"                "B-nazanin"                "test"
root: , attrs: [label = "Arial", label = "B-nazanin", label = "test"]

labellabellabel
"Arial"                "B-nazanin"                "test"
root: , attrs: [label = "Arial", label = "B-nazanin", label = "test"]
```

نمونه ای از آن در تصویر بالا قابل مشاهده است که بر روی یک فایل xml تستی اجرا شده است.

فاز سوم پروژه:

در این فاز سعی کردیم که با استفاده از AST بتوانیم به قطعه کد پایتون معادل برسیم.
برای این کار تابع زیر را نوشتیم:

```
def codeGenerator(ASTLIST):
    imports = "from PyQt5.QtWidgets import *\nfrom PyQt5 import Qttore, QtGui\nfrom PyQt5.QtGui import *\nfrom Pyqt5.QtCore import *\nimport sys\n"
    base = "app = QApplication([])\nwindow = QWidget()\nwindow.setWindowTitle(QFONTCOMBOBOX)\n"
    final_out = imports+base
    final_out = get_ast_str(ASTLIST, start_str=final_out, num_of_space=8)
    final_out += "window.setLayout(QFCB)\nwindow.show()\nsys.exit(app.exec())"

    print(final_out)

    with open("final_out.py", "w") as f:
        f.write(final_out)
```

که در آن ابتدا یک متغیر تعریف کردیم که تمامی import های لازم را شامل میشود سپس قسمت هایی که بین تمامی فایل های xml مشترک میباشد را قرار دادیم و در آخر نیز با استفاده از تابع get_ast_str توانستیم بخش های داخل فایل ast خود را به فایل پایتون تبدیل کنیم.
برای این کار ابتدا یک object از QFontComboBox ساخته میشود و سپس به ازای تمام آیتم های داخل آن به صورت زیر یک خط به کد پایتون نهایی اضافه میشود:

```
_attributes_factory = {
    "label": "{obj_name}.Items.Add(\"{value}\")\n",
    # "fontSize":"{obj_name}.Items.setFontSize({value})\n"
}
```

در آخر فایل xml را به برنامه دادیم و توانستیم به قطعه کد پایتون زیر برسیم:

```
from PyQt5.QtWidgets import *
from PyQt5 import Qttore, OtGui
from PyOt5.OtGui import *
from Pyot5.OtCore import *
import sys
app = QApplication([])
window = QWidget()
window.setWindowTitle(QFONTCOMBOBOX)
QFCB = QFontComboBox()
QFCB.Items.Add("Arial")
QFCB.Items.Add("B-nazanin")
QFCB.Items.Add("test")
QFCB = QFontComboBox()
QFCB.Items.Add("Arial")
QFCB.Items.Add("B-nazanin")
QFCB.Items.Add("test")
window.setLayout(QFCB)
window.show()
sys.exit(app.exec())
```