(به نام خداوند بخشنده‌ی مهربان)

گزارش فاز اول پروژه‌ی درس کامپایلر

OpenUnderstand

گروه ۷:

دیار حامدی

سید عماد موسوی

افشین زنگنه

مرتضی شهرابی فراهانی

فهرست مطالب

مقدمه:

Understand ابزاری قدرتمند برای تجزیه و تحلیل کدها است. در واقع در این نرم‌افزار، تمامی موجودیت‌ها که شامل فایل‌ها، پکیج‌ها، توابع، متغیرها و ... است را در پروژه بررسی می‌کند و همچنین نوع رابطه‌ای که بین هر کدام از این موجودیت‌ها وجود دارد، خط و نام فایلی که این رابطه در آن رخ داده‌است و سایر موارد این‌چنینی را بررسی می‌کند. موارد گفته‌شده توسط برنامه‌ی Understand انجام می‌شود اما هدف از این پروژه پیاده‌سازی متن‌باز از Understand Python API برای تجزیه و تحلیل کدهای منبع است. این کار برای کدهای جاوا با استفاده از ابزارهای Python و Antlr انجام می‌شود. با اجرای این پروژه، کارکردهای برنامه‌ی Understand به صورت رایگان و متن باز در اختیار برنامه‌نویسان قرار می‌گیرد.

نتایج حاصل شده:

در پایان این پروژه API مربوط به define/definIn پیاده‌سازی شده‌است و در نهایت اطلاعات مربوط به موجودیت‌ها و مراجع در پایگاه داده ذخیره می‌شود.

خلاصه‌ای از کارهای انجام شده:

در فاز اول پروژه، وظیفه‌ی گروه ۷ پیاده‌سازی API مربوط به بخش define / defineIn پروژه‌ی OpenUnderstand بوده‌است. در ابتدا مکان پروژه‌ای که قصد بررسی آن را داریم به برنامه داده می‌شود. سپس برای هرکدام از فایل‌ها و بخش‌های مختلف آن پروژه، بخش‌های define/ defineIn که کد آن‌ها در این بخش زده شده‌است، پیدا می‌شود و در دیتابیس در نظر گرفته‌شده برای پروژه ریخته می‌شود.

ساختار ادامه‌ی گزارش:

در ادامه‌ی گزارش، بخش‌های مختلفی از پروژه که در این فاز توسط گروه ۷ تغییر داده شده‌است با جزئیات بیشتر و توضیح بیشتر به ترتیب بررسی می‌شود و توضیحات هر بخش به صورت جداگانه داده می‌شود.

نحوه‌ی تقسیم کار:

بیشتر کار پروژه به صورت تیمی و مشورتی انجام شد. در ابتدا اعضای گروه در چند جلسه‌ی حضوری در سایت دانشکده با هم به بررسی پروژه پرداختند و کدهای هر بخش ما مشارکت همه‌ی اعضای گروه زده شده‌است. اگر بخواهیم به صورت ریزتر، برای هر فرد مشخص کنیم که روی چه بخشی تمرکز بیشتری داشته‌است، نحوه‌ی تقسیم کار بدین صورت می‌شود.

دیار حامدی و سید عماد موسوی: بیشتر بررسی کد define / defineIn

افشین زنگنه: نوشتن گزارش پروژه و بخشی از listener

مرتضی شهرابی فراهانی: نوشتن گزارش و بخش تست و مقایسه‌ی نتایج با نرم افزار Understand

* البته تمامی اعضای گروه در تمام بخش‌های پروژه همکاری داشتند و تقسیم‌بندی بالا تنها بیانگر بخشی است که هریک از افراد در آن بخش تاثیر بیشتری داشته‌اند.

روش پیشنهادی:

در ابتدا برنامه‌ی Understand توسط هریک از اعضای گروه نصب شد و سعی شد تا با بررسی آن با بخش‌های مختلف آن آشنا شویم. پس از به دست آمدن آشنایی نسبی با برنامه و نحوه‌ی کار آن، چند پروژه‌ی نمونه به برنامه داده‌شد و خروجی آن‌ها برای define و defineIn بررسی شد و حالت‌های مختلفی که define و defineIn رخ می‌داد یادداشت شد تا در بخش کدزنی به آن‌ها پرداخته شود. در ادامه به سراغ پروژه‌ی OpenUnderstand رفتیم.

برای توضیح روش پیشنهادی، ابتدا فایل main.py که فایل ابتدایی پروژه است بررسی می‌شود و عملکرد آن توضیح داده می‌شود. این فایل شامل توابع مختلفی است که در این بخش توابع مرتبط با پروژه‌ی گروه ما هم بررسی می‌شوند. سپس در ادامه سایر فایل‌هایی که توسط گروه ما به پروژه اضافه شده‌اند نیز بررسی می‌شوند و نقش هریک از آن‌ها توضیح داده می‌شود.

فایل main.py:

این فایل، فایل اصلی و ابتدایی پروژه است. در این فایل ابتدا در بخش main آن، یک شی از کلاس اصلی این فایل که کلاس Project است ساخته می‌شود. سپس دیتابیس مرتبط با این اجرا ساخته‌می‌شود و در ادامه توسط تابع getListOfFiles که به عنوان ورودی آدرس پروژه را دریافت می‌کند لیست آدرس تمام فایل‌های جاوای پروژه برگردانده می‌شود. سپس به ازای هر کدام از آدرس‌های فایل‌های جاوا، یک بار تابع getFileEntity را صدا می‌زنیم. این تابع به عنوان ورودی آدرس یک فایل جاوا را دریافت می‌کند و در ادامه Entity مربوط به آن فایل را تشخیص می‌دهد و آن را بر می‌گرداند.

در ادامه برای همین فایل تابع Parse را صدا می‌زنیم و با استفاده از این فایل درخت تجزیه‌ی این فایل ساخته می‌شود و برگردانده می‌شود. (توضیحات بیشتر مربوط به هریک از این توابع گفته‌شده در خطوط بالا در صفحات جلوتر آمده‌است. جهت دسترسی سریع‌تر می‌توانید به فهرست مطالب مراجعه کنید.)

پس از ساختن درخت مربوط به این فایل، باید listener های مربوط به این فایل یک به یک صدا زده‌شوند. در این بخش listener های مختلفی صدا زده می‌شوند که هریک وظیفه‌ی خاص خود را دارند و برای تکمیل اطلاعات یک بخش از پایگاه داده کلی پروژه به کار برده می‌شوند. توضیح همه‌ی این موارد خارج از اهداف تسک محول‌شده به گروه ۷ است. بنابراین در این بخش تنها به توضیح listener مربوط به بخش خودمان یعنی بخش define / defineby اکتفا می‌کنیم. توضیح مربوط به listener ی که برای define / defineby ساخته شده‌است، در بخش بعدی به صورت کامل آمده‌است.

تابع getListOfFiles:

این تابع به عنوان ورودی، آدرس پروژه را دریافت می‌کند. یک لیست به نام allFiles ساخته می‌شود و در ادامه فایل‌های مختلف موجود در آدرس داده‌شده بررسی می‌شوند و آدرس تمامی فایل‌هایی که به پسوند java. ختم می‌شوند (که همان فایل‌هایی هستند که ما قصد بررسی آن‌ها را داریم)، به لیست allFiles اضافه می‌شوند.

تابع getFileEntity:

در این تابع ابتدا روی آدرسی که به عنوان ورودی به تابع داده می‌شود، یک سری اصلاحات انجام می‌شود تا به فرمت مورد نظرمان در بیاید. در ادامه این فایل را به صورت یک Entity ذخیره می‌کنیم به طوری که برای فیلد kind مقدار ۱، برای فیلد name نام فایل که از روی آدرس آن به دست می‌آید، برای path آدرسی که در تابع به فرمت درست ساخته‌بودیم و در نهایت در contents هم محتوای فایل ذخیره می‌شود و Entity مورد نظر ساخته می‌شود و در نهایت به عنوان خروجی توسط تابع برگردانده می‌شود.

تابع Parse:

هدف نهایی این تابع ساختن درخت تجزیه‌ی فایلی است که آدرس آن به عنوان ورودی به این تابع داده می‌شود. برای این مهم، لازم است مراحل زیر طی شود. این مراحل به ترتیب عبارتند از:

- بازکردن فایل با استفاده از آدرس فایل که به عنوان ورودی به تابع داده شده‌است.
- فراخوانی تابع JavaLexer مربوط به فایل ساخته‌شده و ذخیره خروجی آن در متغیر lexer
- دریافت توکن‌های مربوط به lexer ی که در بخش قبل برگردانده شده‌بود توسط تابع CommonTokenStream
- فراخوانی parser مربوط به زبان جاوا توسط تابع JavaParserLabeled
- ساخت درخت مربوطه توسط فراخوانی parser.compilationUnit()
- برگرداندن درخت ساخته‌شده به عنوان خروجی تابع

اجرای listener:

```
try:
    # define
    listener = DefineListener()
    p.Walk(listener, tree)
    p.addDefineRefs(listener.defines, file_ent)
except Exception as e:
    print("An Error occurred for reference define in file:" + file_address +
"\n" + str(e))
```

کد ۱ فایل main.py. فراخوانی listener مربوط به بخش Define

در این بخش از کد که در انتهای فایل main وجود دارد، عملیات پیداکردن روابط را فراخوانی می‌کنیم.

تابع addDefineRefs:

هدف از این تابع افزودن روابط پیداشده به پایگاه داده است و کد آن به شرح زیر است:

```
def addDefineRefs(self, ref_dicts, file_ent):
    for ref_dict in ref_dicts:
        if ref_dict["scope"] is None:  # the scope is the file
            scope = file_ent
        else:  # a normal package
            scope = self.getPackageEntity(file_ent, ref_dict["scope"],
ref_dict["scope_longname"])

        ent = self.getPackageEntity(file_ent, ref_dict["ent"],
ref_dict["ent_longname"])

        # Define: kind id 194
        define_ref = ReferenceModel.get_or_create(_kind=194, _file=file_ent,
_line=ref_dict["line"],
                                                  _column=ref_dict["col"],
_ent=ent, _scope=scope)

        # Definein: kind id 195
        definein_ref = ReferenceModel.get_or_create(_kind=195,
_file=file_ent, _line=ref_dict["line"],
                                                    _column=ref_dict["col"],
_scope=ent, _ent=scope)
```

کد ۲ تابع addDefineRefs برای ذخیره سازی در پایگاه داده

define_definein.py:

کلاس DefineListener:

```
class DefineListener(JavaParserLabeledListener):
    def __init__(self):
        self.defines = []
        self.package = ""
```

در این کلاس listener های مرتبط با هر کدام از قسمت‌ها را پیاده‌سازی می‌کنیم. هم‌چنین در ابتدا لیستی از پکیج‌ها و define ها را ایجاد می‌کنیم و در هر listener نتایج مربوطه را در این لیست‌ها ذخیره می‌کنیم.

انواع listener ها:

```
def enterPackageDeclaration(self,
ctx:JavaParserLabeled.PackageDeclarationContext):
    self.package = [str(i) for i in ctx.qualifiedName().IDENTIFIER()]
    ent_start = ctx.qualifiedName().IDENTIFIER()[0]
    ent_name = ctx.qualifiedName().IDENTIFIER()[-1].getText()
    ent_longname = ".".join(self.package)
    line = ent_start.symbol.line
    column = ent_start.symbol.column
    self.defines.append({
        "scope": None, "ent": ent_name,
        "scope_longname": None, "ent_longname": ent_longname,
        "line": line, "col": column
    })

def enterClassDeclaration(self,
ctx:JavaParserLabeled.ClassDeclarationContext):
    ent = ctx.IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.addDefineInfo(ent, ent_parents)

def enterInterfaceDeclaration(self,
ctx:JavaParserLabeled.InterfaceDeclarationContext):
    ent = ctx.IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.addDefineInfo(ent, ent_parents)

def enterMethodDeclaration(self,
ctx:JavaParserLabeled.MethodDeclarationContext):
    ent = ctx.IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.addDefineInfo(ent, ent_parents)

def enterAnnotationTypeDeclaration(self,
ctx:JavaParserLabeled.AnnotationTypeDeclarationContext):
```

```
    ent = ctx.IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.addDefineInfo(ent, ent_parents)

def enterConstructorDeclaration(self,
ctx:JavaParserLabeled.ConstructorDeclarationContext):
    ent = ctx.IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.addDefineInfo(ent, ent_parents)

def enterVariableDeclarator(self,
ctx:JavaParserLabeled.VariableDeclaratorContext):
    ent = ctx.variableDeclaratorId().IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.addDefineInfo(ent, ent_parents)

def enterEnumConstant(self, ctx:JavaParserLabeled.EnumConstantContext):
    ent = ctx.IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.addDefineInfo(ent, ent_parents)

def enterEnumDeclaration(self, ctx:JavaParserLabeled.EnumDeclarationContext):
    ent = ctx.IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.addDefineInfo(ent, ent_parents)
```

```
    self.add_define_info(ent,   ent_parents   +   [ent.getText()],   'values')
    self.add_define_info(ent, ent_parents + [ent.getText()], 'valueOf')
```

```
def enterFormalParameter(self,
ctx:JavaParserLabeled.FormalParametersContext):
    ent = ctx.variableDeclaratorId().IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.addDefineInfo(ent, ent_parents)
```

```

```

```
def enterLambdaParameters0(self,
ctx:JavaParserLabeled.LambdaParameters0Context):
    self.lambda_expression_count += 1
    ent = ctx.IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    ent_name = f'(lambda_expr_{self.lambda_expression_count})'
    self.add_define_info(ent, ent_parents, ent_name)
    self.add_define_info(ent, ent_parents + [ent_name])

def enterLambdaParameters2(self,
ctx:JavaParserLabeled.LambdaParameters2Context):
    self.lambda_expression_count += 1
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    ent_name = f'(lambda_expr_{self.lambda_expression_count})'
    identifiers = ctx.IDENTIFIER()
    self.add_define_info(identifiers[0], ent_parents, ent_name)
```

```
    for ent in identifiers:
        self.add_define_info(ent, ent_parents + [ent_name])

def enterEnhancedForControl(self,
ctx:JavaParserLabeled.EnhancedForControlContext):
    ent = ctx.variableDeclaratorId().IDENTIFIER()
    ent_parents = class_properties.ClassPropertiesListener.findParents(ctx)
    self.add_define_info(ent, ent_parents)
```

کد ۳ فایل define_defineby.py

در ادامه به توضیح هر کدام از listener ها می‌پردازیم:

- متد enterPackageDeclaration:

  این متد برای یافتن تعریف فایل‌ها توسط package ها به کار می‌رود.

- متد enterClassDeclaration:

  این متد برای یافتن تعریف class به کار می‌رود.

- متد enterInterfaceDeclaration:

  این متد برای یافتن تعریف interface به کار می‌رود.

- متد enterClassDeclaration:

  این متد برای یافتن تعریف کلاس‌ها به کار می‌رود.

- متد enterInterfaceDeclaration:

  این متد برای یافتن تعریف اینترفیس به کار می‌رود.

- متد enterMethodDeclaration:

  این متد برای یافتن تعریف متدها به کار می‌رود.

- متد enterAnnotationTypeDeclaration:

  این متد برای یافتن تعریف annotation به کار می‌رود.

- متد enterConstructorDeclaration:

  این متد برای یافتن تعریف متد constructor به کار می‌رود.

- متد enterVariableDeclarator:

  این متد برای یافتن تعریف متغیرها به کار می‌رود.

- متد enterEnumConstant:

  این متد برای یافتن تعریف enum به کار می‌رود.

- متد enterEnumDeclaration:

  این متد برای یافتن متغیرهای تعریف‌شده در enum به کار می‌رود.

- متد enterFormalParameter:

  این متد برای یافتن تعریف پارامترهای توابع به کار می‌رود.

- متد enterLambdaParameters0:

  این متد برای یافتن عبارات لاندایی است که یک پارامتر دارند.

- متد enterLambdaParameters2:

  این متد برای یافتن عبارات لاندایی است که بیش از یک پارامتر دارند.

- متد enterEnhancedForControl:

  این متد برای یافتن پارامترهایی است که داخل foreach جاوا تعریف می‌شوند.

تابع addDefineInfo:

```
def addDefineInfo(self, ent, ent_parents):
    ent_name = ent.getText()
    line = ent.symbol.line
    column = ent.symbol.column
    scope_longname = ".".join(self.package + ent_parents)
    ent_longname = scope_longname + "." + ent_name
    if len(ent_parents) == 0:
        scope_name = None
    else:
        scope_name = ent_parents[-1]
    self.defines.append({
        "scope": scope_name, "ent": ent_name,
        "scope_longname": scope_longname, "ent_longname": ent_longname,
        "line": line, "col": column
    })
```

کد ٤ تابع addDefineInfo در فایل main.py

در این تابع اطلاعات پیداشده توسط listener ها را در لیست defines اضافه می‌کنیم.

لازم به ذکر است به علت اینکه فایل class_properties.py به طور کامل نتایج مورد نظر ما را پیدا نمی‌کرد، تابع findParents آن را کمی تغییر دادیم. بعد از تغییر این تابع به صورت زیر درآمد.

```
@staticmethod
def findParents(c: ParserRuleContext):  # includes the ctx identifier
    parents = []
    current = c.parentCtx
    while current is not None:
        if current.getRuleIndex() in [
            JavaParserLabeled.RULE_classDeclaration,
            JavaParserLabeled.RULE_methodDeclaration,
            JavaParserLabeled.RULE_enumDeclaration,
            JavaParserLabeled.RULE_interfaceDeclaration,
            JavaParserLabeled.RULE_constructorDeclaration,
            JavaParserLabeled.RULE_annotationTypeDeclaration,
        ]:
            parents.append(current.IDENTIFIER().getText())
        current = current.parentCtx
    return list(reversed(parents))
```

کد ٥ تابع findParents در فایل class_properties.py

ارزیابی:

در ابتدای امر، برای ارزیابی برنامه چند فایل کوچک را به Understand و OpenUnderstand دادیم و نتایج آن‌ها را با بررسی یک به یک نتایج این دو برنامه، مقایسه کردیم. سپس بعد از اینکه ایرادات مربوط به این فایل‌های کوچک برطرف شدند، برای بررسی فایل‌ها و پروژه‌های بزرگ‌تر نیاز بود تا یک فایل برای بررسی دقیق‌تر نتایج Understand ایجاد شود. برای این کار از api هایی که خود نرم‌افزار Understand در اختیار ما قرار می‌داد استفاده کردیم.

ابتدا لازم است تا پروژه را به نرم افزار Understand بدهیم تا فایلی با پسوند udb. توسط نرم افزار ساخته شود. سپس با استفاده از دستور زیر می‌توانیم به محتوای تولید شده توسط نرم‌افزار understand دسترسی داشته باشیم

```
try:
   import understand
except ImportError:
   print("Can not import understand")
```

```
db=understand.open(r"D:\works\university\term6\compiler\Project\Open Understand\benchmark\JSON\JSON.udb")
```

در ادامه می‌توانیم به نتایج به شیوه‌ی دلخواه دسترسی داشته باشیم. برای این منظور گروه ما نتایج را در کنسول پرینت می‌کرد. همچنین یک متغیر counter هم برای محاسبه‌ی تعداد فیلدهای ساخته شده تعریف شده است. محتوای کامل فایل بالا با نام test.py در زیر آورده شده‌است.

```
from pprint import pprint

try:
    import understand
```

```
except ImportError:
    print("Can not import understand")

db =
understand.open(r"D:\works\university\term6\compiler\Project\OpenUnderstand\b
enchmark\calculator_app\calculator_app\calculator_app1.udb")

counter = 0

for ent in db.ents():
    for ref in ent.refs():

        if ref.kindname() == "Define":
            counter += 1

            print("++++++++++++++++++++++++")

            print(f"entity: {ent}\n, ref: {ref}\n ref.scope: {ref.scope()},
ref.ent: {ref.ent()}\n"
                  f"ref.line: {ref.line()}, ref.col: {ref.column()},
ref.file: {ref.file().name()}")
            print("-------------------------------------------------------")
            print(f"ref.ent.name:{ref.ent().name()},
ref.ent.longname:{ref.ent().longname()} ,ref.ent.kind:{ref.ent().kind()}\n"
                  f"ref.ent.parent:{ref.ent().parent()},
ref.ent.value:{ref.ent().value()},ref.ent.type:{ref.ent().type()}\n")
            print("-------------------------------------------------------")

print(counter)
```

کد ٦ فایل test.py

در ادامه نتایج بدست آمده از اجرای برنامه‌ی calculator_app توسط برنامه‌ی Understand و همچنین توسط کد نوشته‌شده‌ی ما آورده شده‌است.

```
D:\works\university\term6\compiler\Project\OpenUnderstand\docs>python test.py
++++++++++++++++++++++++++
entity: printLog.java
, ref: Define app.method printLog.java(1)
 ref.scope: printLog.java, ref.ent: app.method
ref.line: 1, ref.col: 8, ref.file: printLog.java
----------------------------------------------------------
ref.ent.name:app.method, ref.ent.longname:com.calculator.app.method ,ref.ent.kind:Package
ref.ent.parent:basic_operation.java, ref.ent.value:None,ref.ent.type:None

----------------------------------------------------------
++++++++++++++++++++++++++
entity: printLog.java
, ref: Define method.printLog printLog.java(8)
 ref.scope: printLog.java, ref.ent: method.printLog
ref.line: 8, ref.col: 13, ref.file: printLog.java
----------------------------------------------------------
ref.ent.name:method.printLog, ref.ent.longname:com.calculator.app.method.printLog ,ref.ent.kind:Public Class
ref.ent.parent:printLog.java, ref.ent.value:None,ref.ent.type:None

----------------------------------------------------------
++++++++++++++++++++++++++
entity: method.printLog
, ref: Define printLog.java printLog.java(8)
 ref.scope: method.printLog, ref.ent: printLog.java
ref.line: 8, ref.col: 13, ref.file: printLog.java
----------------------------------------------------------
ref.ent.name:printLog.java, ref.ent.longname:D:\works\university\term6\compiler\Project\OpenUnderstand\benchmark\calculat
ref.ent.parent:None, ref.ent.value:None,ref.ent.type:None

----------------------------------------------------------
++++++++++++++++++++++++++
entity: method.printLog
, ref: Define printLog.print printLog.java(10)
 ref.scope: method.printLog, ref.ent: printLog.print
ref.line: 10, ref.col: 16, ref.file: printLog.java
----------------------------------------------------------
ref.ent.name:printLog.print, ref.ent.longname:com.calculator.app.method.printLog.print ,ref.ent.kind:Public Method
ref.ent.parent:method.printLog, ref.ent.value:None,ref.ent.type:void

----------------------------------------------------------
++++++++++++++++++++++++++
entity: printLog.print
, ref: Define method.printLog printLog.java(10)
 ref.scope: printLog.print, ref.ent: method.printLog
ref.line: 10, ref.col: 16, ref.file: printLog.java
----------------------------------------------------------
ref.ent.name:method.printLog, ref.ent.longname:com.calculator.app.method.printLog ,ref.ent.kind:Public Class
ref.ent.parent:printLog.java, ref.ent.value:None,ref.ent.type:None

----------------------------------------------------------
```

شکل ۱: نتایج مربوط به دیتابیس Understand

در این پروژه در مجموع ۱۴۸ مورد define, defineby پیدا شد. به علت اینکه پوشش دادن تمام این موارد در فایل باعث زیاد شدن بیهوده‌ی حجم داکیومنت می‌شود و شاید از کیفیت لازم هم برخوردا نباشد، بنابراین عکس سایر موارد به طور کامل آورده نشده است و در عکس بعدی تعداد define, defineby ها آورده شده که بتوانیم با بخش بعدی که نتایج openUnderstand است بهتر مقایسه انجام بدهیم.

```
++++++++++++++++++++++++++
entity: tag
, ref: Define println.print println.java(14)
 ref.scope: tag, ref.ent: println.print
ref.line: 14, ref.col: 29, ref.file: println.java
-------------------------------------------------------
ref.ent.name:println.print, ref.ent.longname:com.calculator.app.display.println.print ,ref.ent.kind:Public Method
ref.ent.parent:display.println, ref.ent.value:None,ref.ent.type:void

-------------------------------------------------------
++++++++++++++++++++++++++
entity: text
, ref: Define println.print println.java(14)
 ref.scope: text, ref.ent: println.print
ref.line: 14, ref.col: 41, ref.file: println.java
-------------------------------------------------------
ref.ent.name:println.print, ref.ent.longname:com.calculator.app.display.println.print ,ref.ent.kind:Public Method
ref.ent.parent:display.println, ref.ent.value:None,ref.ent.type:void

-------------------------------------------------------
++++++++++++++++++++++++++
entity: println.print_fail
, ref: Define display.println println.java(25)
 ref.scope: println.print_fail, ref.ent: display.println
ref.line: 25, ref.col: 16, ref.file: println.java
-------------------------------------------------------
ref.ent.name:display.println, ref.ent.longname:com.calculator.app.display.println ,ref.ent.kind:Public Class
ref.ent.parent:println.java, ref.ent.value:None,ref.ent.type:None

-------------------------------------------------------
++++++++++++++++++++++++++
entity: println.print_fail
, ref: Define pf println.java(27)
 ref.scope: println.print_fail, ref.ent: pf
ref.line: 27, ref.col: 19, ref.file: println.java
-------------------------------------------------------
ref.ent.name:pf, ref.ent.longname:com.calculator.app.display.println.print_fail.pf ,ref.ent.kind:Variable
ref.ent.parent:println.print_fail, ref.ent.value:print_fail(),ref.ent.type:print_fail

-------------------------------------------------------
++++++++++++++++++++++++++
entity: pf
, ref: Define println.print_fail println.java(27)
 ref.scope: pf, ref.ent: println.print_fail
ref.line: 27, ref.col: 19, ref.file: println.java
-------------------------------------------------------
ref.ent.name:println.print_fail, ref.ent.longname:com.calculator.app.display.println.print_fail ,ref.ent.kind:Public Method
ref.ent.parent:display.println, ref.ent.value:None,ref.ent.type:void

-------------------------------------------------------
148
```

شکل ۲: ادامه‌ی نتایج مربوط به دیتابیس Understand

در عکس‌های بعدی، نتایج مربوط به دیتابیس OpenUnderstand آورده شده‌است.

	_id	_kind_id	_file_id	_line	_column	_ent_id	_scope_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	194	1	1	8	2	1
2	2	195	1	1	8	1	2
3	3	194	1	8	13	4	3
4	4	195	1	8	13	3	4
5	5	194	1	10	16	6	5
6	6	195	1	10	16	5	6
7	7	194	1	10	29	7	5
8	8	195	1	10	29	5	7
9	9	194	1	14	16	6	5
10	10	195	1	14	16	5	6
11	11	194	1	14	29	8	5
12	12	195	1	14	29	5	8
13	13	194	1	14	41	7	5
14	14	195	1	14	41	5	7
15	15	194	1	20	16	10	9
16	16	195	1	20	16	9	10
17	17	194	1	25	16	12	11
18	18	195	1	25	16	11	12
19	19	194	1	27	19	13	11
20	20	195	1	27	19	11	13
21	21	194	14	1	8	15	14
22	22	195	14	1	8	14	15
23	23	194	14	8	13	17	16
24	24	195	14	8	13	16	17
25	25	194	14	9	16	19	18
26	26	195	14	9	16	18	19
27	27	194	20	1	8	21	20
28	28	195	20	1	8	20	21
29	29	194	20	7	13	23	22
30	30	195	20	7	13	22	23
31	31	194	20	9	23	25	24
32	32	195	20	9	23	24	25
33	33	194	26	1	8	27	26
34	34	195	26	1	8	26	27
35	35	194	26	8	13	29	28
36	36	195	26	8	13	28	29
37	37	194	26	10	23	31	30
38	38	195	26	10	23	30	31
39	39	194	26	10	37	32	30
40	40	195	26	10	37	30	32
41	41	194	26	12	17	33	30
42	42	195	26	12	17	30	33
43	43	194	26	17	16	34	30
44	44	195	26	17	16	30	34
45	45	194	26	18	15	35	30
46	46	195	26	18	15	30	35
47	47	194	36	1	8	37	36
48	48	195	36	1	8	36	37
49	49	194	36	5	13	39	38
50	50	195	36	5	13	38	39

	_id	_kind_id	_file_id	_line	_column	_ent_id	_scope_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
51	51	194	36	6	22	41	40
52	52	195	36	6	22	40	41
53	53	194	36	6	30	42	40
54	54	195	36	6	30	40	42
55	55	194	36	6	36	43	40
56	56	195	36	6	36	40	43
57	57	194	36	10	25	41	40
58	58	195	36	10	25	40	41
59	59	194	36	10	36	42	40
60	60	195	36	10	36	40	42
61	61	194	36	10	46	43	40
62	62	195	36	10	46	40	43
63	63	194	36	14	22	45	44
64	64	195	36	14	22	44	45
65	65	194	36	14	41	46	44
66	66	195	36	14	41	44	46
67	67	194	36	14	48	47	44
68	68	195	36	14	48	44	47
69	69	194	36	18	25	45	44
70	70	195	36	18	25	44	45
71	71	194	36	18	47	46	44
72	72	195	36	18	47	44	46
73	73	194	36	18	57	47	44
74	74	195	36	18	57	44	47
75	75	194	36	22	25	49	48
76	76	195	36	22	25	48	49
77	77	194	36	22	36	50	48
78	78	195	36	22	36	48	50
79	79	194	36	22	46	51	48
80	80	195	36	22	46	48	51
81	81	194	52	1	8	53	52
82	82	195	52	1	8	52	53
83	83	194	52	3	13	55	54
84	84	195	52	3	13	54	55
85	85	194	52	4	25	57	56
86	86	195	52	4	25	56	57
87	87	194	52	4	47	58	56
88	88	195	52	4	47	56	58
89	89	194	52	4	58	59	56
90	90	195	52	4	58	56	59
91	91	194	52	4	66	60	56
92	92	195	52	4	66	56	60
93	93	194	52	5	15	61	56
94	94	195	52	5	15	56	61
95	95	194	52	6	17	62	56
96	96	195	52	6	17	56	62
97	97	194	63	1	8	64	63
98	98	195	63	1	8	63	64
99	99	194	63	6	13	66	65
100	100	195	63	6	13	65	66

	_id	_kind_id	_file_id	_line	_column	_ent_id	_scope_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
101	101	194	63	7	19	67	65
102	102	195	63	7	19	65	67
103	103	194	63	8	19	68	65
104	104	195	63	8	19	65	68
105	105	194	63	9	19	69	65
106	106	195	63	9	19	65	69
107	107	194	63	11	18	71	70
108	108	195	63	11	18	70	71
109	109	194	63	11	37	72	70
110	110	195	63	11	37	70	72
111	111	194	63	11	50	73	70
112	112	195	63	11	50	70	73
113	113	194	63	11	58	74	70
114	114	195	63	11	58	70	74
115	115	194	63	13	15	75	70
116	116	195	63	13	15	70	75
117	117	194	63	14	15	76	70
118	118	195	63	14	15	70	76
119	119	194	63	15	17	77	70
120	120	195	63	15	17	70	77
121	121	194	63	23	18	79	78
122	122	195	63	23	18	78	79
123	123	194	63	23	43	80	78
124	124	195	63	23	43	78	80
125	125	194	63	23	56	81	78
126	126	195	63	23	56	78	81
127	127	194	63	23	64	82	78
128	128	195	63	23	64	78	82
129	129	194	63	25	15	83	78
130	130	195	63	25	15	78	83
131	131	194	63	26	15	84	78
132	132	195	63	26	15	78	84
133	133	194	63	27	17	85	78
134	134	195	63	27	17	78	85
135	135	194	63	35	18	87	86
136	136	195	63	35	18	86	87
137	137	194	63	35	42	88	86
138	138	195	63	35	42	86	88
139	139	194	63	35	53	89	86
140	140	195	63	35	53	86	89
141	141	194	90	1	8	91	90
142	142	195	90	1	8	90	91
143	143	194	90	8	13	93	92
144	144	195	90	8	13	92	93
145	145	194	90	10	16	95	94
146	146	195	90	10	16	94	95
147	147	194	90	10	29	96	94
148	148	195	90	10	29	94	96

شکل ۳: مجموعه شکل‌های نتایج دیتابیس OpenUnderstand

با بررسی نتایج متوجه می‌شویم که هم دیتابیس OpenUnderstand به ما ۱۴۸ خروجی می‌دهد و هم خروجی فایل test.py. همچنین در صورتی که دقیق‌تر بررسی کنیم، هر یک از سطرهای این دو خروجی باهم مطابقت دارند. بنابراین برنامه به درستی کار می‌کند.

مشکلات و چالش‌ها:

یکی از مشکلات به‌وجود آمده در پروژه، تلاش برای ساخت فایل تست بود. در ابتدا ما فکر می‌کردیم که امکان دسترسی به پایگاه داده Understand وجود ندارد و به همین علت زمان زیادی را صرف بررسی دستی نتایج برنامه‌ی خودمان با Understand کردیم.

پس از آن‌که متوجه شدیم امکان بررسی بهتر نتایج وجود دارد، با بررسی API های سایت Understand متوجه شدیم که اگر آدرس فایل udb ساخته شده را به متد open از کتابخانه‌ی understand بدهیم، می‌توانیم به داده‌ها دسترسی داشته باشیم. اما در ابتدا با دادن پروژه‌های مختلف به برنامه‌ی Understand فایل udb توسط برنامه ساخته نمی‌شد. در اینجا متوجه شدیم که باید نسخه‌های قدیمی‌تر از Understand را نصب می‌کردیم و ورژن کنونی با کرک، همچین فایلی نمی‌سازد.

پس از نصب دوباره‌ی نسخه‌های قدیمی‌تر برنامه، وقتی تلاش کردیم تا فایل test.py را اجرا کنیم، این بار برنامه متد open را شناسایی نمی‌کرد. پس از پرس‌وجو از سایر دانشجویان و همچنین دستیاران آموزشی، متوجه شدیم که بعد از نصب برنامه‌ی Understand لازم بود تا یک فایل از پوشه‌ی Scripts به نام api_install_test.py را اجرا کنیم و پس از رفع ارورهای آن امکان شناسایی متد open وجود دارد.

پس از اجرای مراحل بالا، درنهایت توانستیم فایل test.py را اجرا کنیم.

نتیجه‌گیری و کارهای آتی:

با استفاده از بخش پیاده‌سازی‌شده می‌توان رابطه define/definein را در کدهای جاوا پیدا و تجزیه و تحلیل کرد. در آینده می‌توان راهی برای مقایسه نتایج ابزار Open Understand و ابزار Understand ارائه داد تا بتوان به صورت اتوماتیک نتایج را مقایسه کرد و از صحت نتایج اطمینان یافت.