Разработать инструмент командной строки для визуализации графа зависимостей, включая транзитивные зависимости. Сторонние средства для получения зависимостей использовать нельзя.
Зависимости определяются по имени пакета языка Java (Maven). Для описания графа зависимостей используется представление PlantUML.
Визуализатор должен выводить результат в виде сообщения об успешном выполнении и сохранять граф в файле формата png. Ключами командной строки задаются:
• Путь к программе для визуализации графов. (Флаг отброшен, т.к. используется pythonplantuml)
• Имя анализируемого пакета.
• Путь к файлу с изображением графа зависимостей.
• Максимальная глубина анализа зависимостей.
Все функции визуализатора зависимостей должны быть покрыты тестами.

Описание раюоты программы

Программа включает в себя несколько файлов интерпритируемого языка python. Для запуска приложения следует открыть файл `main.py` при помощи комманды:

```
python3 main.py
```

Дополнительные библиотеки для работы проекта не требуются. Используются лишь стандартные.

Флаги программы

```
-l, --library LIBRARY Путь к plantuml
-p, --package PACKAGE Имя анализируемого пакета
-f, --file FILE       Путь к файлу с изображением графа зависимостей
-d, --depth DEPTH     Максимальная глубина анализа зависимостей
```

Описание функций программы

Программа включает в себя четыре файла:

1. `main.py`
2. `generateoutput.py`
3. `reqgen.py`
4. `argsniffer.py`

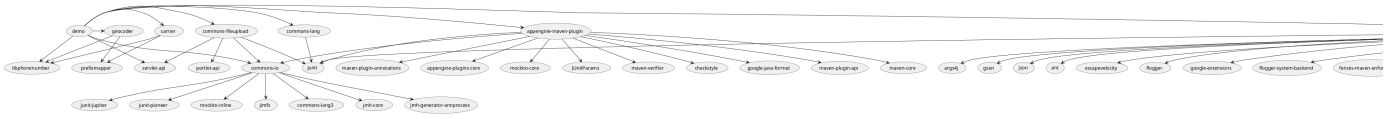
Все остальные файлы вспомогательные. Главным файлом является `reqgen.py`, реализующий запись в файл, как и парсинг

reqgen.py

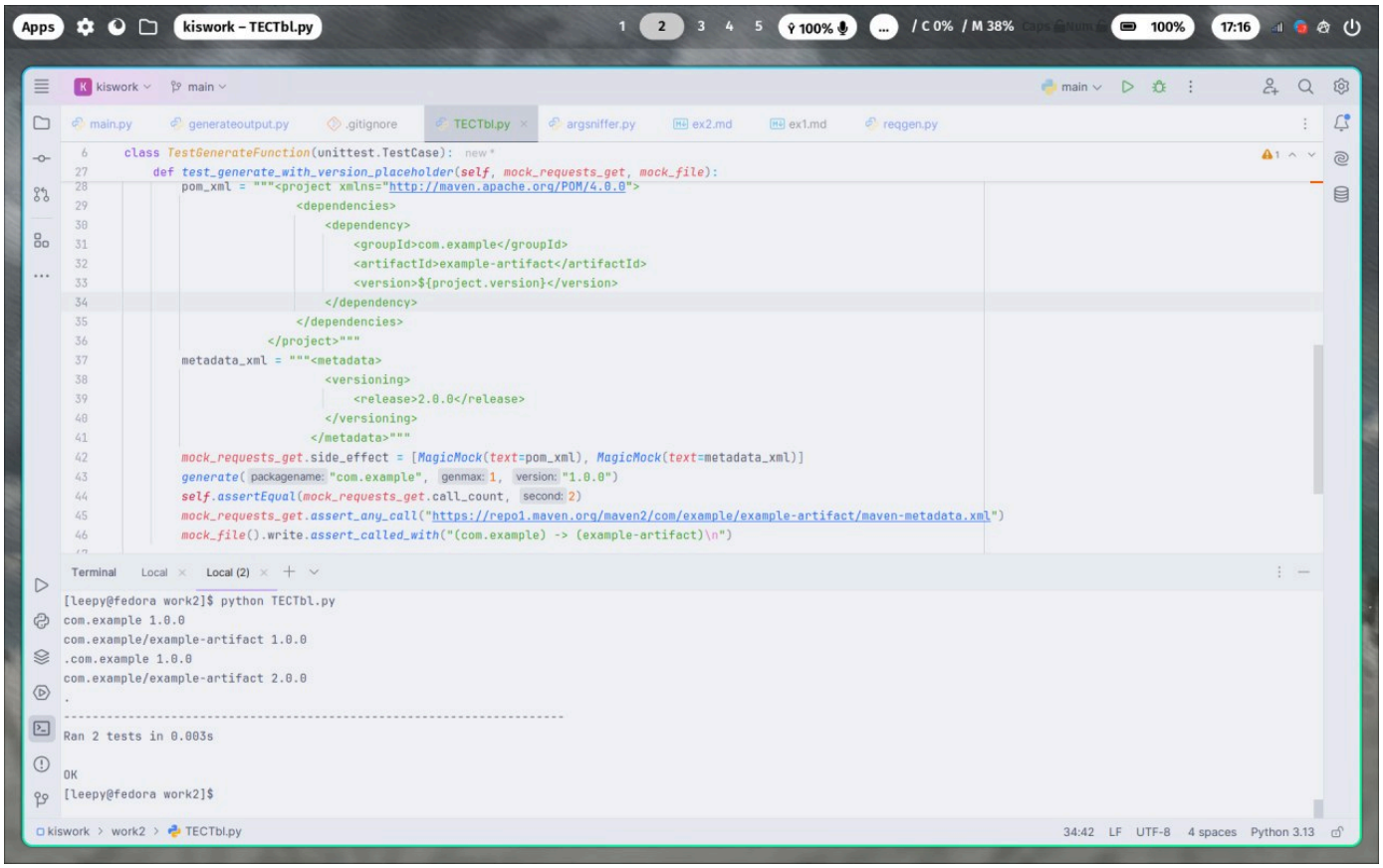
1. generate(packagename, genmax, version = 'latest', generation = 0, outfile = "test")
   Парсинг зависимостей и запись их в файл

Демонстрация работы программы:

```
$ python main.py -l plantuml-1.2024.7.jar -p pkg:maven/com.googlecode.libphonenumber/demo -f a.jpg -d
com.googlecode.libphonenumber/demo 8.13.50
com.google.template/soy 2022-07-20
com.google.code.findbugs/jsr305 3.0.2
com.google.errorprone/error_prone_annotations 2.14.0
...
```



Тестирование программы:


Apps
kiswork – TECTbl.py
1 2 3 4 5
100%
/ C 0% / M 38%
100%
17:16
kiswork
main
main.py
generateoutput.py
.gitignore
TECTbl.py
argsniffer.py
ex2.md
ex1.md
reqgen.py
class TestGenerateFunction(unittest.TestCase):
def test_generate_with_version_placeholder(self, mock_requests_get, mock_file):
pom_xml = """<project xmlns="http://maven.apache.org/POM/4.0.0">
<dependencies>
<dependency>
<groupId>com.example</groupId>
<artifactId>example-artifact</artifactId>
<version>${project.version}</version>
</dependency>
</dependencies>
</project>"""
metadata_xml = """<metadata>
<versioning>
<release>2.0.0</release>
</versioning>
</metadata>"""
mock_requests_get.side_effect = [MagicMock(text=pom_xml), MagicMock(text=metadata_xml)]
generate(packagename: "com.example", genmax: 1, version: "1.0.0")
self.assertEqual(mock_requests_get.call_count, second: 2)
mock_requests_get.assert_any_call("https://repo1.maven.org/maven2/com/example/example-artifact/maven-metadata.xml")
mock_file().write.assert_called_with("(com.example) -> (example-artifact)\n")
Terminal
Local
Local (2)
[leepy@fedora work2]$ python TECTbl.py
com.example 1.0.0
com.example/example-artifact 1.0.0
.com.example 1.0.0
com.example/example-artifact 2.0.0
.
Ran 2 tests in 0.003s
OK
[leepy@fedora work2]$
kiswork > work2 > TECTbl.py
34:42 LF UTF-8 4 spaces Python 3.13