

**عنوان گزارش :**

گزارش فاز دوم پروژه کامپایلر

**استاد:**

**دکتر سعید پارسا**

**اعضای تیم:**

محمد جواد مهدی تبار

معین کاظمی

امیرحسین سماوات

علی رهنما علمداری

فهرست مطالب

[مقدمه 3](#_Toc107320437)

[کد اجرا شده برای پر کردن دیتابیس 3](#_Toc107320438)

[CountDeclClassMethod 3](#_Toc107320439)

[مقایسه خروجی understand با ما 4](#_Toc107320440)

[CountDeclClassVariable 6](#_Toc107320441)

[مقایسه خروجی understand‌ با ما 7](#_Toc107320442)

[CountDeclExecutableUnit 8](#_Toc107320443)

[مقایسه خروجی understand با ما 9](#_Toc107320444)

[CountDeclFile 10](#_Toc107320445)

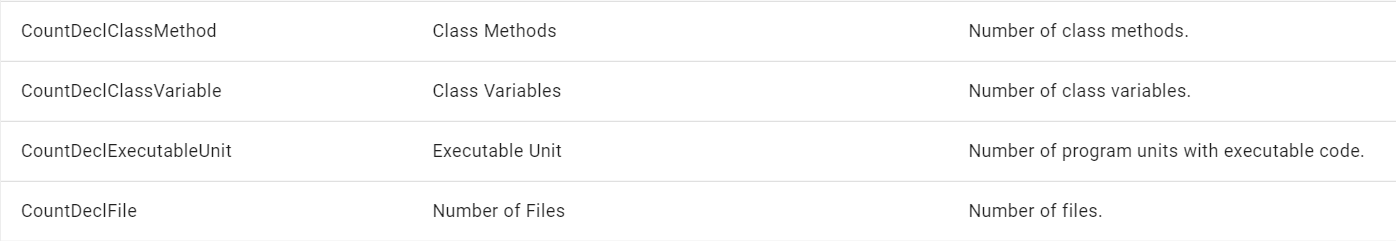
[مقایسه خروجی understand با ما 11](#_Toc107320446)

# مقدمه

در این فاز پروژه قرار است کدهایی که برای فاز اول زده بودیم را تست کنیم و چک کنیم تا چه حد درست کار می‌کند در واقع دیتابیس‌های درست شده در فاز اول را با دیتابیس تولیدی خود understand چک می‌کنیم که آیا دقیقا تمام آیتم‌ها برابر می‌باشند یا خیر.

این بار باید با metrics کار کنیم که این metrics‌ یک سری api از پیش تعریف‌شده دارد که understand به ما نشان می‌دهد هر کدام از آنها دقیقا چه معیاری را می‌خواهند.

متریک‌های خواسته شده برای گروه ما به شرح زیر می‌باشد

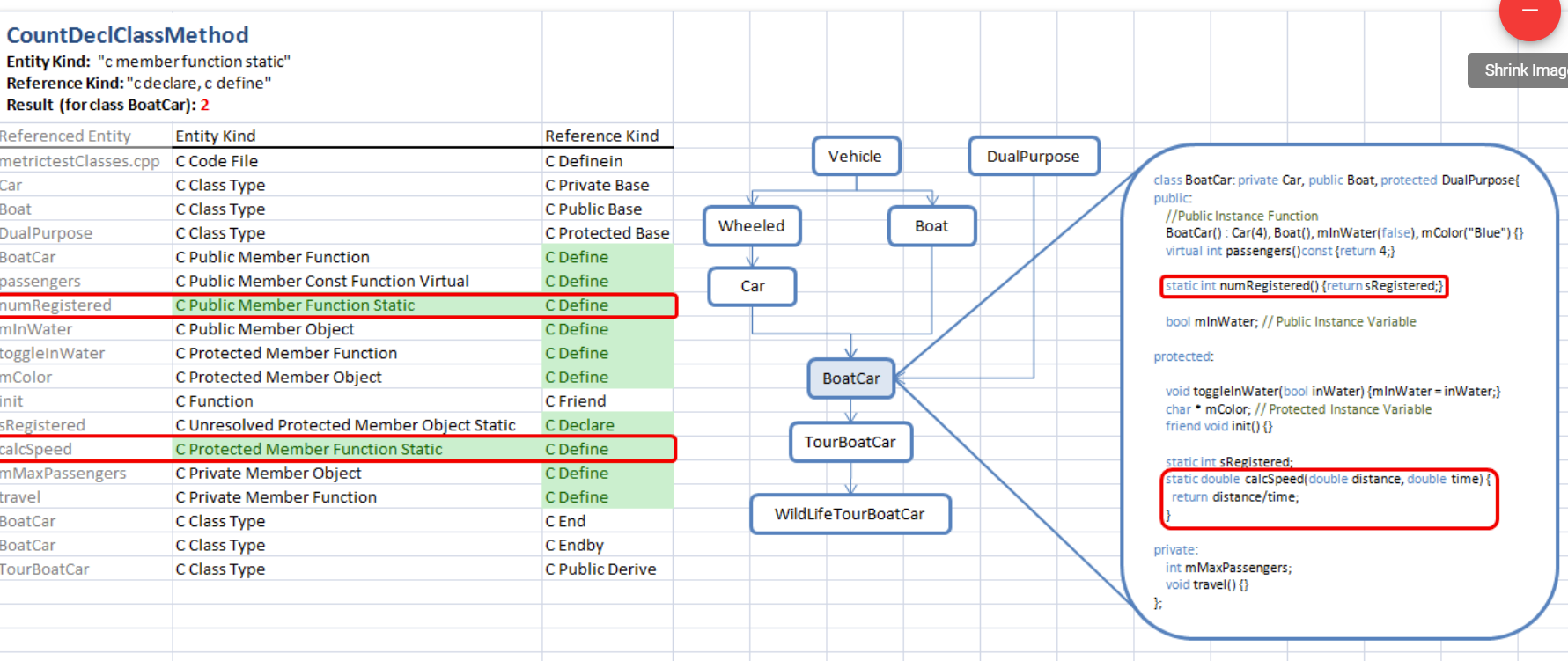


# کد اجرا شده برای پر کردن دیتابیس

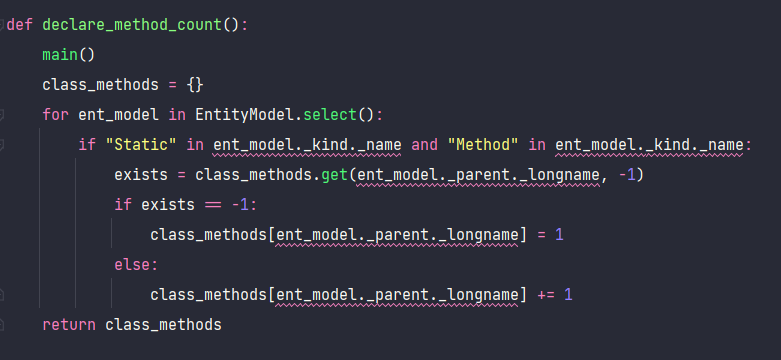
برای پر کردن دیتابیس از کد فاز ۱ گروه خودمان استفاده کردیم که define را پیاده‌سازی کرده بودیم که تقریبا تمام entity هایی موجود در پروژه را در دیتابیس می‌سازد.

نام فایل : define\_and\_definein.py

# CountDeclClassMethod



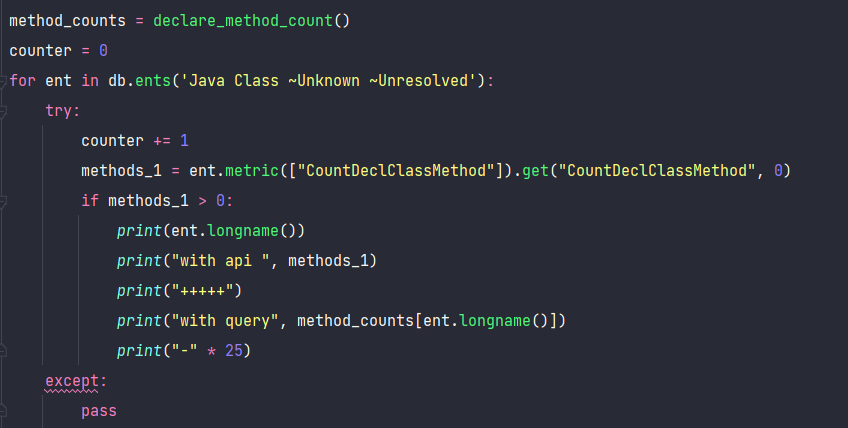
برای پیدا کردن این بخش باید تمام function های static را برگردانیم .



ابتدا برای پر شدن table‌هایمان تابع main ، فایل define را ران می‌کنیم .

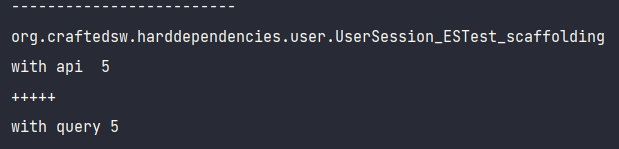
برای پیدا کردن این نوع از تابع‌ها روی تمام آیتم‌های تیبل EntityModel لوپ میزنیم و آنهایی که Kind شان Static و Method هر دو را با هم داشت آن‌ها را به دیکشنری‌مان اضافه می‌کنیم که دیکشنری ما در واقع اینگونه پر می‌شود به ازای هر class که key دیکشنری ما می‌باشد تمام method هایی که در آن کلاس موجود باشند را در می‌اورد و تعداد آن method ها value هر key می‌باشد.

## مقایسه خروجی understand با ما



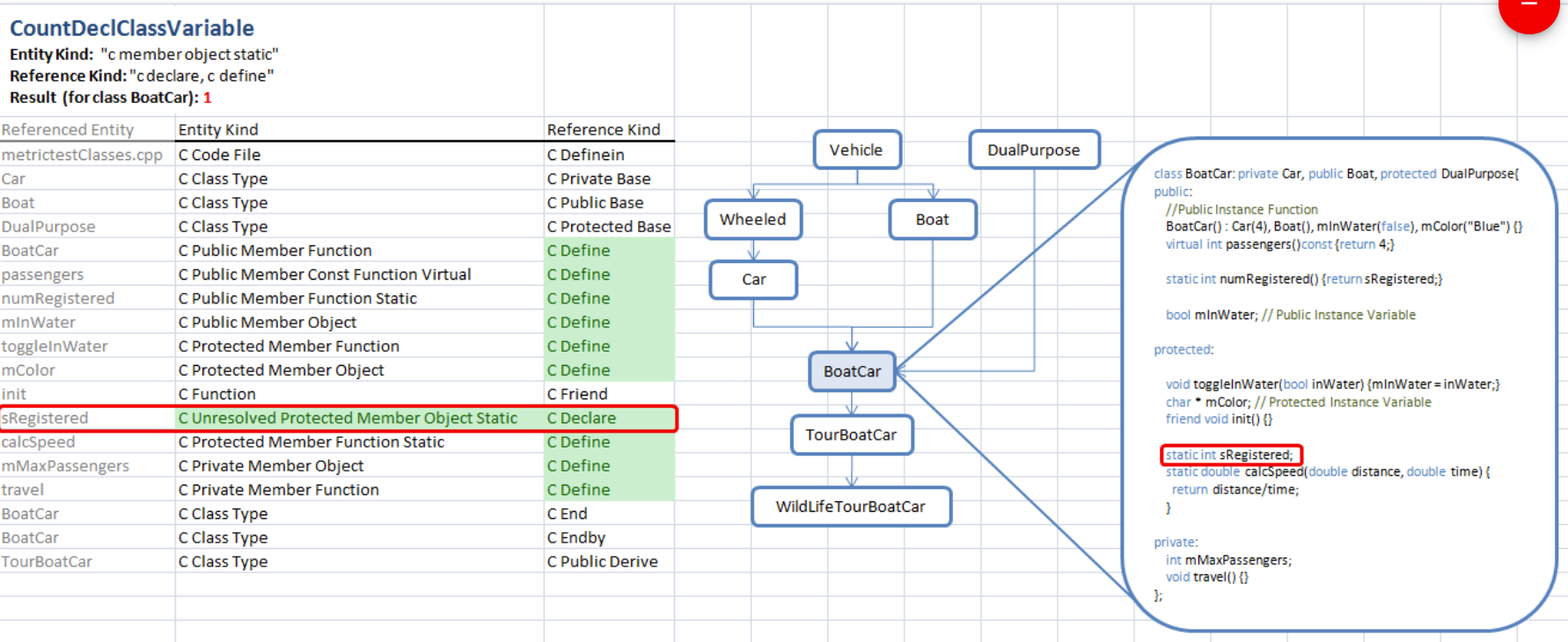
همانطور که می‌بینید ابتدا تابع خود را کال کرده تا جواب‌هایمان را بگیریم سپس با تابع built-in understand که متریک ها را به ما برمی‌گرداند مقایسه می‌کنیم جواب‌هایمان را



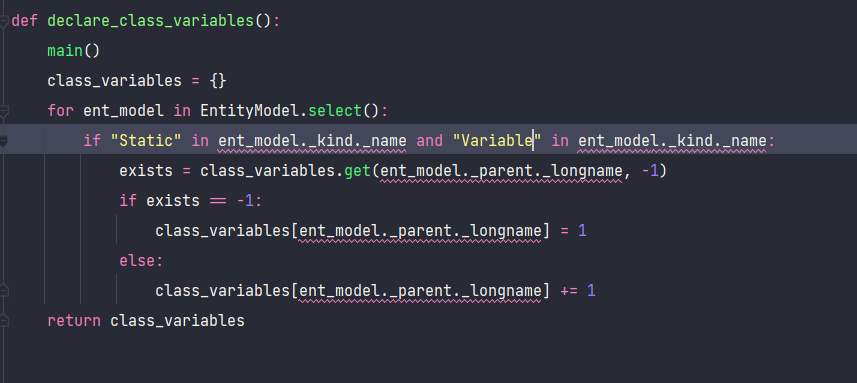


تعداد به دست آمده از دیتابیس understand با دیتابیس ما برابر است .

# CountDeclClassVariable



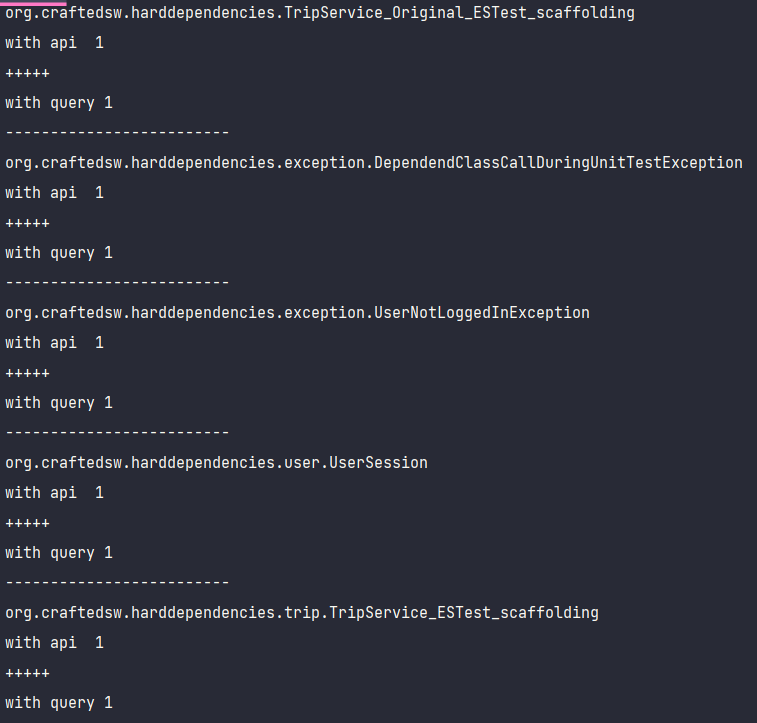
برای پیدا کردن باید تمام variable های static را برگردانیم.

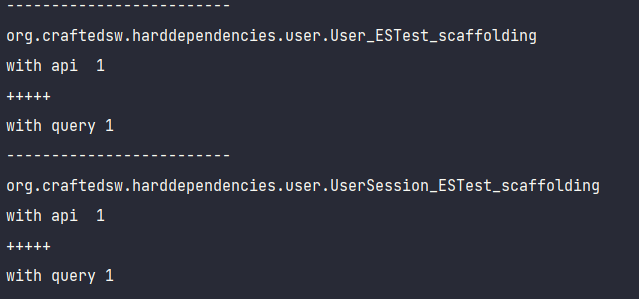


این کد هم همانند کد بالا کار می‌کند و به جای method باید Varaible ها را برگردانیم.

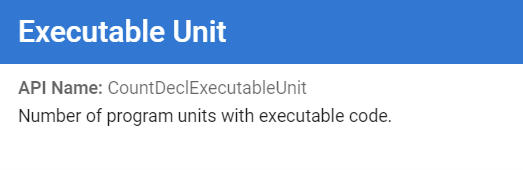
## مقایسه خروجی understand‌ با ما







# CountDeclExecutableUnit

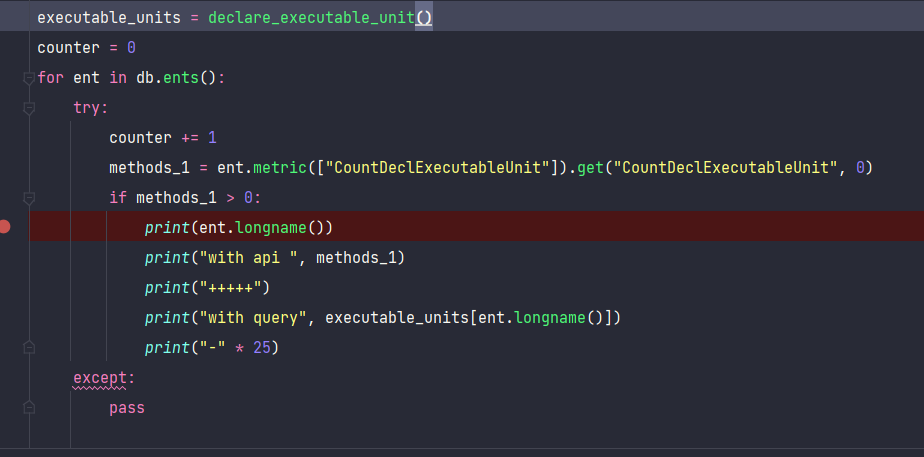


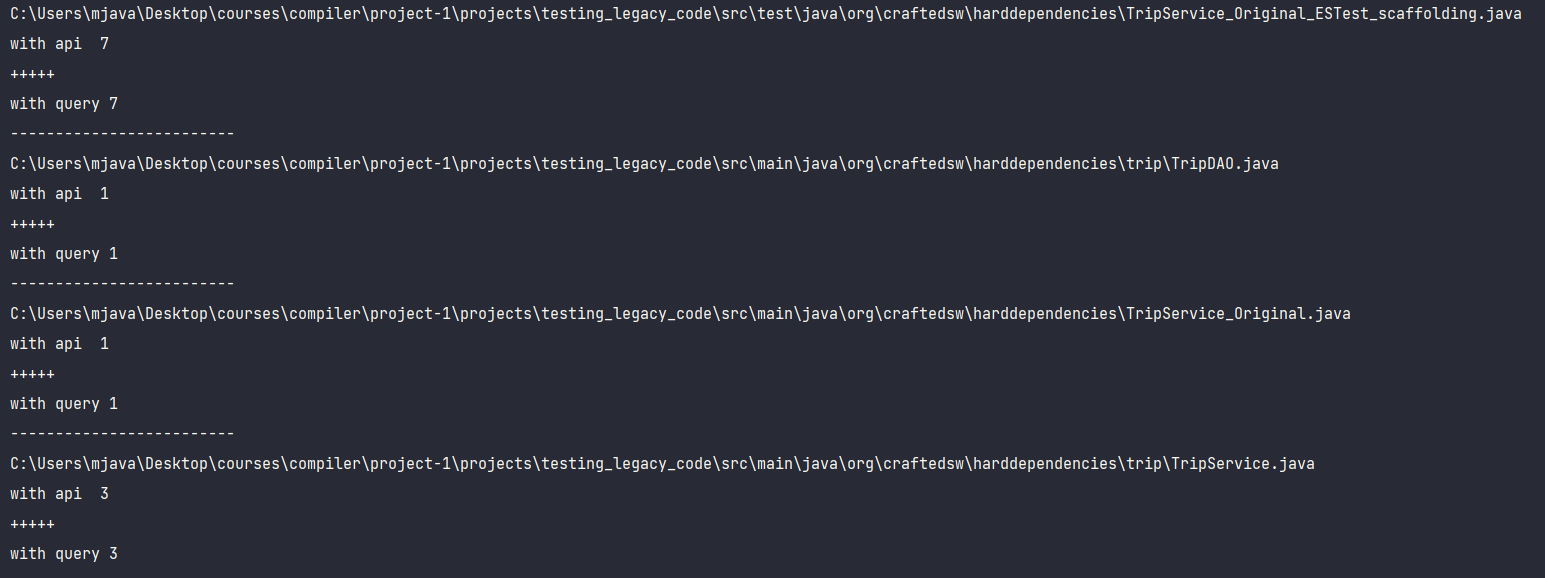
باید تمام برنامه‌های قابل اجرا را برگردانیم

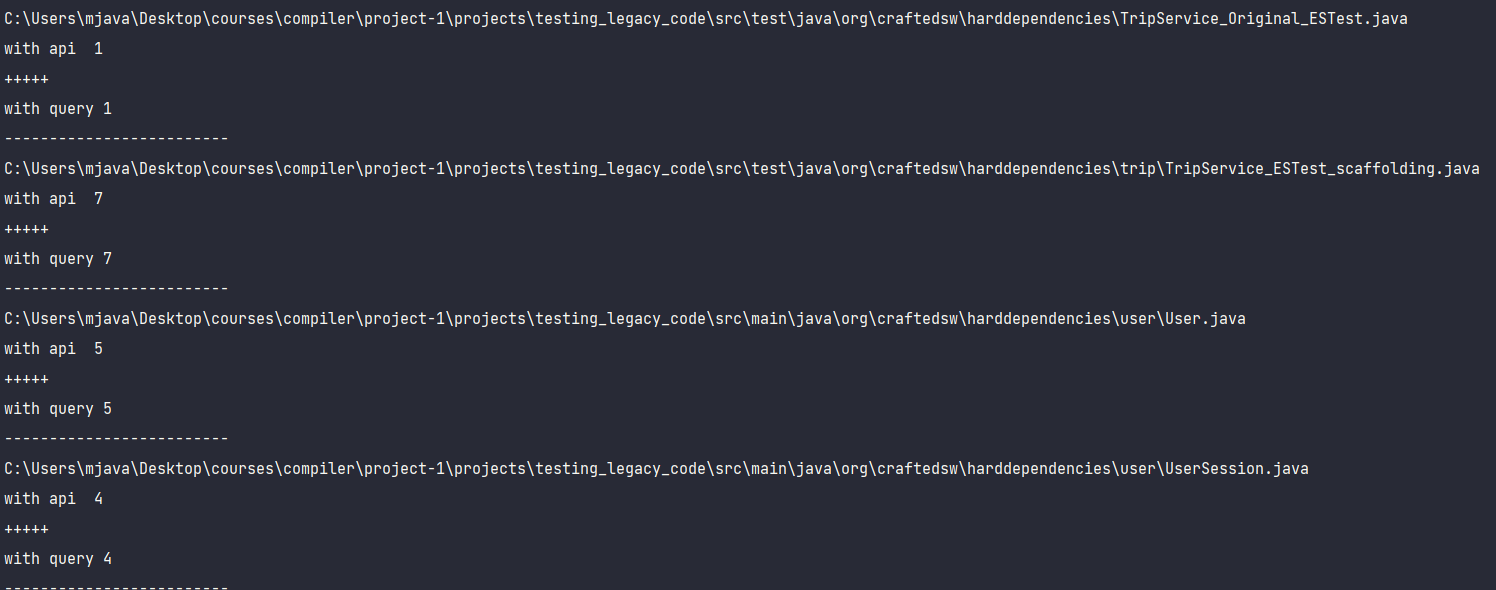


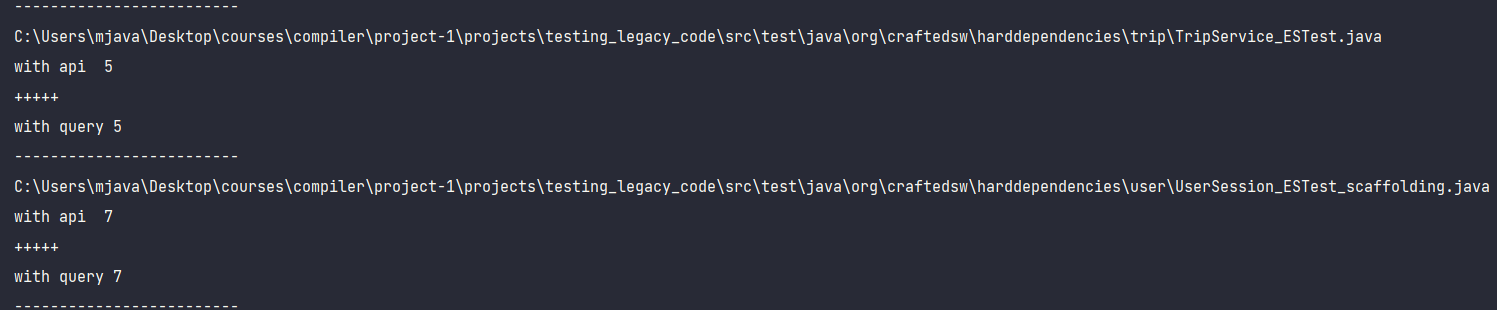
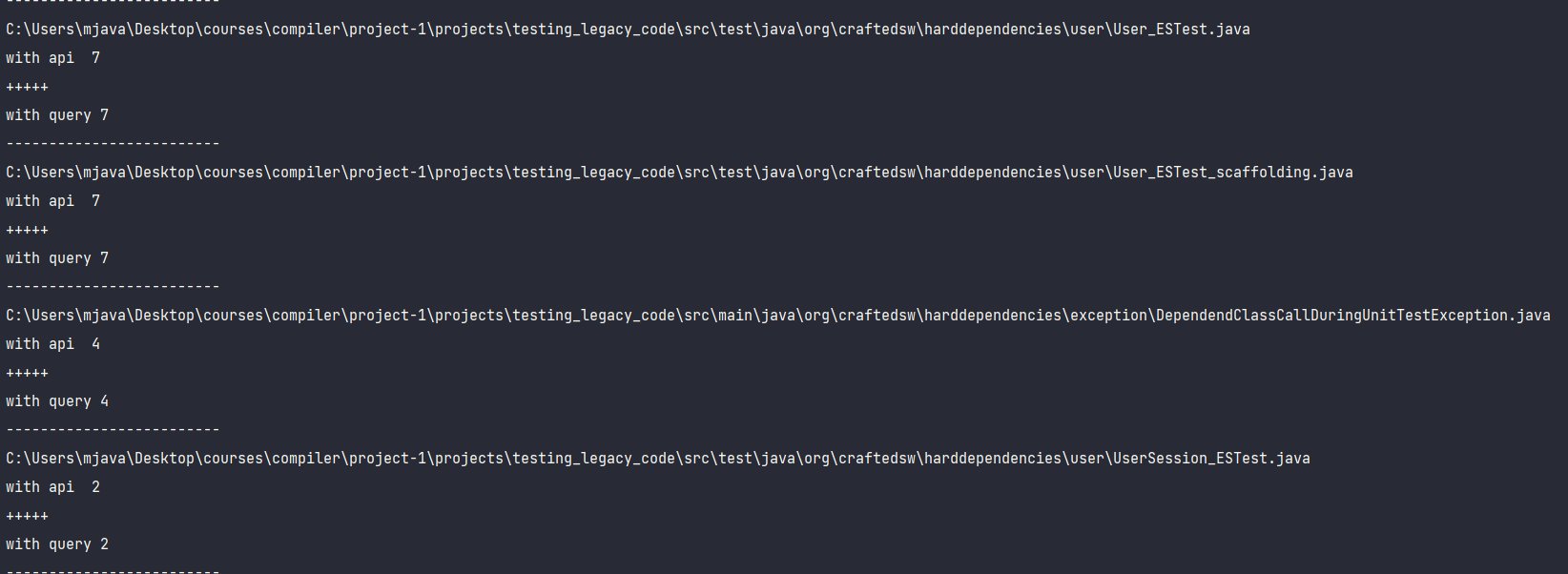
که بعد از ران کردن understand‌فهمیدیم که executable unit‌ها همان Method هایمان می‌باشند که در واقع قابل اجرا هستند و ما باید در هر فایل پیدا کنیم چند method وجود دارد و آن‌ها را برگردانیم

## مقایسه خروجی understand با ما



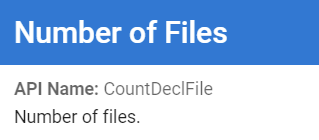




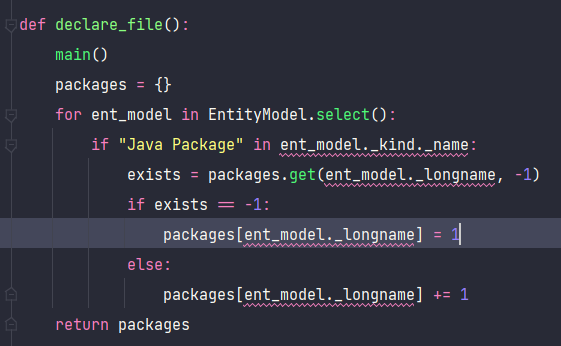


**تماما یکسان می‌باشند.**

# CountDeclFile



که فایل های موجود در پروژه را می‌شمارد. که در واقع باید تعداد package های به کار رفته را شمرد.



## مقایسه خروجی understand با ما





**که تماما یکسان می‌باشند.**