به نام خدا

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

درس آزمون نرم افزار

پروژه دوم

نام و نام خانوادگی	دانیال سعیدی(810198571) محمد قره حسنلو(810198461)
تاریخ ارسال گزارش	1401/9/6
ریپو گیتهاب	
Commit IDآخرین	7b09d6263df12b3dc3ca01274dcea482ec76dd6c

فهرست گزارش سوالات

سوال 1

در تست اول از stub استفاده شده است؛ زیرا در stub دیتاهای از پیش تعریف شده را نگهداری میکنیم و از آنها به عنوان جواب به فراخوانی ها در تست ها استفاده میکنیم. در این مثال ما تعریف میکنیم که چه دیتایی باید از فراخوانی متود getCredicts از course به عنوان خروجی بازگردد و بعد از آن از خروجی به دست آمده در تست در assertEquals استفاده میکنیم. در واقع داریم از state verification استفاده میکنیم. زمانی استفاده میشود که نمیخواهیم object هایی را درگیر کنیم که با داده های واقعی پاسخ دهند یا عوارض جانبی دارند. مثلا یک object که باید از دیتابیس دیتا بردارد تا به یک method call جولب دهد، به جای داده واقعی(مانند این مثال که) یک stub تعریف میکنیم و معین میکنیم که چه دیتایی را باید برگرداند.

در تست دوم از mock استفاده شده است؛ زیرا در mock از verify استفاده میکنیم تا آن اکشن های موردنظر که مدنظرمان بوده که انجام شوند را چک و verify میکنیم. در واقع داریم از behavior verification استفاده میکنیم. در اینجا میخواهیم چک کنیم که در هنگام انجام شدن متود isPassed از StudyRecord، آیا از getMinValidGrade در استفاده از graduateLevel فراخوانی شده و این را با verify تایید میکنیم. به این دلیل از mock استفاده میکنیم که نخواهیم به production code استناد کنیم یا راه ساده برای verify کردن وجود ندارد یا ممکن است return value وجود نداشته باشد.

در تست سوم در خط دوم از رویکرد stub استفاده شده است که یک مقداری را بخواهیم به عنوان خروجی از متود getMinValidGrade داشته باشیم و همچنین graduateLevel که در خط سوم خارج میشود، همانی باشد که با متود mock در خط اول تعریف شده است و در نهایت با این خروجی هایی که انتظار داریم خارج شوند، متود isPassed از studyRecord را چک میکند. این همان رویکرد stub را دارد که خروجی هایی از چند متود را تعریف کردیم(بدون توجه به ورودی هایی که میدیم) و خروجی مورد نظر از متود isPassed را با آن شرایط سنجیدیم که درست کار میکند یا خیر.

سوال 3

در این قسمت باید از رفتاری که در تابع خصوصی checkLoop اتفاق می افتد مطلع شویم و برای این کار باید از behavior verification استفاده کنیم که باعث میشود که ما از ماک استفاده کنیم. چون به طور مستقیم به checkLoop دسترسی نداریم، باید از تعداد exception ها و تعداد حالت هایی که در پیش نیاز ها باعث این لوپ میشوند را بشماریم و مقایسه کنیم که یکی هستند یا نه، اگر یکی بودند یعنی تست pass میشود.