Software Testing

CA1 Report

810197636

سوال اول

تست کردن متدهای پرایوت bad practice محسوب می‌شود.

یکی از دلایل این می‌باشد که اگر متد پرایوت را تست کنیم به redundancy خواهیم خورد. از آن‌جایی که متدهای پرایوت در متدهای پابلیک ظاهر می‌شوند و به تنهایی ظاهر نمی‌شوند، این که ما دوباره متد پرایوت را تست کنیم باعث افزونگی می‌شود.

یکی دیگه از دلایل این هست که ممکنه یک dead code در متد پرایوت وجود داشته باشد که در واقعیت هیچ‌وقت به آن دسترسی پیدا نکنیم از طریق پابلیک متدها. بنابراین این تست نشان می‌ده که متد پرایوت ما درست کار می‌کند در حالی که هیچ وقت در واقعیت به آن لاجیک برخورد نخواهیم کرد.

بهترین شرایط تست کردن متدهای پرایوت، تست کردن به صورت indirectly و از طریق تست کردن متدهای پابلیک می‌باشد.

سوال دوم

تست کردن correctness کدهای multi-threaded توسط یونیت تست، کار سختی می‌باشد. چالش اصلی آن هم به دلیل non-deterministic بودن نتیجه می‌باشد. ممکن است که کد دچار race condition و یا deadlock شود، که این کار را چالش برانگیز می‌کند.

یونیت تست کردن بیشتر برای آن می‌باشد که بخشی از تکه‌های کد را ایزوله کنیم و از درستی کارایی آن

بخش کد، اطمینان حاصل کنیم. در کدهای multi-threaded به دلیل اینکه تردها باید با هم کامیونیکیت کنند، باعث می‌شود که یونیت تست کردن آن بخش تبدیل به کار پیچیده‌ای شود.

در نهایت اگر کد پرایوت ما خیلی پیچیده باشد و یا بفهمیم که خوب کار نمی‌کند، در واقع در شرایط ضروری، می‌توان در آن صورت برای آن تست نوشت.

سوال سوم

۱. مشکلات این تکه کد:

۱. در خط ۴ از System.out.println استفاده شده که تنها به کاربر نشان می‌دهد که expected چی بوده و actual چی هست. در اینجا باید از assertEquals استفاده کنیم. به صورت زیر:

assertEquals(10, result)

۲. مشکلات این تکه کد:

۱. باید خط expects Exception پاک شود.

۲. در نهایت نیز خط زیر باید اضافه شود به آن:

assertThrows(Exception, new AnotherClass().process(badInput)

۳. مشکلات این تکه کد:

۱. در تست کردن اگر قرار است که مراحل initialization قبل از هر تست یکی باشد، بهتر است از دو متد beforeEach و setup استفاده شود.
