گزارش کار پروژه تست 2

سید علی امام زاده 810199377، محمدحسین عقیلی 810199576

شناسه اخرین کامیت: de152b82ce2b5dcd8b0055ec71cf665d252ae955

سوال 1)

Dependency Injection by Constructor :

این الگو زمانی ایده‌آل است که می‌خواهید اطمینان حاصل کنید که یک کلاس همیشه وابستگی‌های مورد نیاز خود را از لحظه شروع آن دارد.

مزایا:

باعث می شود که کلاس پس از ایجاد، در وضعیت معتبری باشد.

این یک قرارداد روشن و صریح برای وابستگی های کلاس فراهم می کند.

معایب:

اگر وابستگی های زیادی وجود داشته باشد، Constructor ها می توانند پیچیده شوند.

تغییر وابستگی ها ممکن است نیاز به تغییر Constructor داشته باشد.

Dependency Injection by Setter :

این الگو زمانی مناسب است که وابستگی های اختیاری دارید یا زمانی که نیاز به تغییر وابستگی در زمان اجرا دارید.

مزایا:

امکان انعطاف پذیری با وابستگی های اختیاری و تغییرات زمان اجرا را فراهم می کند.

بدون نیاز به تغییر Constructor هنگام افزودن یا تغییر وابستگی ها.

معایب:

تا زمانی که همه تنظیم‌کننده‌های مورد نیاز فراخوانی شوند، کلاس ممکن است در وضعیت معتبری نباشد که منجر به مشکلاتی می‌شود.

تست کردن در این شرایط می تواند چالش برانگیزتر باشد زیرا باید مطمئن شوید که همه setter های مورد نیاز فراخوانی شده اند.

Dependency Injection by Field:

این الگو معمولاً به دلیل اتصال محکم و محدودیت در تست برای تزریق وابستگی، ایده آل نیست.

مزایا:

سادگی و مختصر بودن در کد.

معایب:

بالا ذکر شد.

- Dependency Injection by Constructor انتخاب ایده‌آل تری برای تزریق وابستگی است، زیرا یک حالت واضح و قابل پیش‌بینی را برای کلاس به محض نمونه‌گیری اعمال می‌کند و یک قرارداد کاملاً تعریف شده برای وابستگی‌های آن ارائه می‌کند.
-
- **سوال 2)**

الف) test doubles رفتار وابستگی‌های واقعی را تقلید می‌کنند، اما پیاده‌سازی واقعی تولید نیستند. Test Doubles برای جداسازی کد تحت آزمایش و ایجاد محیط های آزمایش کنترل شده استفاده می شود.

اصطلاح «imposter» هم همین ایده را دارد که این جایگزین‌ها وانمود می‌کنند که چیز واقعی برای اهداف آزمایشی هستند.

ب) انواع مختلف Test Double:

Dummy Objects: مکان‌هایی هستند که در مواقعی که پارامتر مورد نیاز است استفاده می‌شوند اما در واقع در مورد آزمایشی استفاده نمی‌شوند. آنها معمولاً به‌عنوان آرگومان‌ها به متدها یا سازنده‌ها ارسال می‌شوند، اما تأثیری بر رفتار آزمون ندارند.

Stubs: به فراخوانی‌های متد پاسخ‌های آماده ارائه می‌کنند و داده‌های از پیش تعریف‌شده را برمی‌گردانند. آنها برای شبیه سازی رفتار روش های یک شی واقعی استفاده می شوند. هنگامی که نیاز دارید کد مورد آزمایش را از وابستگی های پیچیده جدا کنید، Stub ها به ویژه مفید هستند.

Mocks: ماک ها برای تأیید اینکه روش های خاص با آرگومان های خاص یا تعداد معینی بار در طول یک آزمون فراخوانی می شوند استفاده می شوند. ماک ها کمک می کنند تا اطمینان حاصل شود که تعاملات بین اشیاء همانطور که انتظار می رود رخ می دهد.

Spies: جاسوس ها شبیه به mock هستند اما برای نظارت و ضبط تعاملات بین کد مورد آزمایش و وابستگی ها استفاده می شوند. آنها به شما این امکان را می دهند تا تعاملاتی را که در طول یک آزمون اتفاق افتاده است بدون الزاماً اعمال انتظارات بررسی کنید.

سوال 3)

Mockist Testing:

تمرکز بر تست تعامل:

تست ساختگی بین اشیا موثر است. مخصوصاً برای سناریوهایی که اشیا به درستی با هم همکاری می کنند مناسب است.

استفاده از اشیاء ساختگی:

در تست ماک، به شدت برای شبیه سازی رفتار وابستگی ها استفاده می شوند.ماک ها به شما این امکان را می دهند که انتظارات را در فراخوانی متدها تنظیم و تعاملات را تأیید میکند.

مزایا:

کنترل دقیق و دقیقی را بر روی تعاملات و رفتار ارائه می دهد و تشخیص دقیق علت مشکلات را آسان تر می کند. با وادارکردن شما به طراحی کد به گونه ای که تست پذیرتر و ماژولارتر باشد، اتصال شل را تشویق می کند. برای تأیید اینکه یک شی به درستی با وابستگی هایش همکاری می کند عالی است.

معایب:

می تواند منجر به تست های بیش از حد مشخص شود که به شدت با جزئیات پیاده سازی مرتبط هستند و نگهداری آنها را شکننده و دشوار می کند. نوشتن تست ماک گسترده می تواند زمان بر باشد و تست ها ممکن است بیش از حد پیچیده شوند.

Classical Testing:

تمرکز بر وضعیت و نتایج:

آزمایش کلاسیک بر تأیید وضعیت و نتایج رفتار یک شی متمرکز است و بیشتر به نتیجه فراخوانی متدها می پردازد تا تعاملات دقیق.

استفاده محدود از mock:

آزمایش کلاسیک معمولاً از پیاده‌سازی واقعی وابستگی‌ها استفاده می‌کند و برخلاف ماک، بر رفتار انتها به انتها سیستم تمرکز می‌کند.

مزایا:

به دلیل حساسیت کمتر نسبت به تغییرات پیاده سازی، تمایل به تولید تست های قوی تر را دارد.

معمولاً منجر به تست های کمتر و کد تمیزتر و ساده‌تر می‌شود.

معایب:

ممکن است بینش دقیقی در مورد تعاملات بین اشیاء نداشته باشد.