گزارش‌کار تمرین کامپیوتری دوم آزمون نرم افزار

اولدوز نیساری 810199505
مجید فریدفر 810199569

لینک ریپازیتوری: https://github.com/maj2idfar/Baloot
هش آخرین کامیت: 89c5355

سوال 1

1. Dependency injection by Constructor:

در این روش dependency ها توسط constructor کلاس تامین می‌شوند. این روش تضمین می‌کند که قبل از آن که instance ای از کلاسی ساخته شود ، تمام وابستگی های مورد نیاز آن را داریم و وابستگی ها را غیر قابل تغییر می‌کند. این روش می تواند منجر به زیاد شدن تعداد پارامتر ها در constructor و پیچیده شدن ساختن تست و ماک شود .

2. Dependency injection by Setter:

در این روش کاربر از یک متد setter که injector استفاده میکند که dependency ها را inject کند استفاده می‌کند. این روش امکان optional dependency ها و امکان اضافه کردن dependency ها به صورت پویا در زمان اجرا را فراهم می‌کند .زمانی که ترتیب صدا شدن متد ها اهمیت داشته باشد این روش می تواند منجر به temporal coupling شود (یعنی وابستگی برنامه به زمان) و باعث تغییر پذیری کلاس و ایمنی کمتر آن شود.

3. Dependency injection by Field :

در این روش injector به طور مستقیم مقدار مشخصی را به یک فیلد public یا annotated نسبت می‌دهد.این روش با این که ساده و مختصر است اصل encapsulation را زیر پا می‌گذارد و تست کردن کلاس یا ماک ساختن برای آن را دشوار تر می‌کند.

سوال 2

الف) به test doubleها اصطلاحا imposter گفته می شود زیرا آن ها object های جعلی هستند که رفتار object های واقعی را در سیستم تقلید می کنند. Test double ها برای جدا کردن unit under test از وابستگی هایش و verify کردن تعاملات بین آن ها می باشد .
هم چنین imposters اسم یک ابزار mountbank رایگان است، یک ابزار محبوب open source شبیه سازی که test double می سازد. Mountebank این امکان را به ما می دهد که object های جعلی برای پروتکل های HTTP, TCP , .. بسازیم و response های آن ها configure کنیم. با mount bank می توانیم ماک هایی از پروتکل ها بسازیم و response های واقعی آن ها را با انتظارات مان مقایسه کنیم.

ب)

1. Dummy object :

اشیا خیلی ساده ای هستند که فقط new می شوند و مثلا به عنوان پارامتر به کلاس داده می شوند که از compile error ها جلوگیری شود . این اشیا مورد استفاده یا دسترسی قرار نمی گیرند و هیچ عملکرد منطقی ندارند. مثالی از آن ها نام کاربری و رمز تصادفی برای پر کردن فیلد های ثبت نام در سایتی است.

2. Stub:

اشیایی هستند که return value خاص و پاسخ های predefined برای SUT فراهم میکنند. برای کنترل indirect input ها به کار می روند و جلوگیری از side effect ها و error ها از اهداف آن است.

3. Spy

اشیایی هستند که interactionها با آبجکت‌ها در طول تست را به همراه اطلاعاتشان ثبت میکنند. بیشتر برای verify کردن indirect output و برای آن که چک کنیم که متد هایی که انتظار اجرا یا دسترسی داشتیم ، اجرا شده یا به آنها دسترسی شده است استفاده می شود.یک مثال معروف از spy ها object هایی هستند که تعداد کال شدن متد های دیگر را می شمارند.

4. Mock

اشیایی پیچیده تر از stub ها و spy ها که در واقع هم عملکرد stub را دارند و هم عملکرد spy را .Mock ها در خودشان state نگه می دارند و خروجی هم که می دهند مبتنی بر state

است .Mock ها نه تنها response های از پیش تعیین شده می دهند این امکان را هم دارند که انتظاراتی از نحوه استفاده از آن ها تعیین کنیم . هم چنین ماک ها برای تایید اجرای متد ها با آرگومان های مشخص و با تربیت معلوم به کار می روند .ماک ها برای verify کردن رفتار SUT به کار می روند .

5.Fake object

پیاده سازی ساده شده از اشیا واقعی هستند که برای تست کردن استفاده می شود. اصطلاحا برای ساده کردن و سریع تر کردن تست ها به میانبر یا hard code نیاز دارند.در کل از آن ها برای کنترل شده تر کردن و بهینه تر کردن تست ها استفاده می شود.

سوال 3

در کل دو رویکرد می توانیم به تست کردنمان داشته باشیم : دیدگاه کلاسیک و دیدگاه mockist

دیدگاه کلاسیک تمرکز بیشتری روی تست کردن ورودی و خروجی دارد در حالی که mockist تمرکزی بیشتری روی تست کردن تعاملات بین object ها دارد .

روش کلاسیک ساده تر و آسان تر است و کمتر شکننده است ، یعنی احتمال آن که تست ها در نتیجه تغییرات کد تغییر کنند کمتر است . با این حال انعطاف پذیری کمتری دارد زیرا اجازه تعاملات بین اشیا را نمی دهد .

روش mockist در سوی دیگر انعطاف پذیری بیشتری دارد و اجازه تعاملات را میدهد.درکل استفاده از آن ها باعث می شود که کد بهتری بنویسیم چون باعث می شود با تعاملات بین اشیا مواجه شویم . اما نوشتن آن ها دشوار و پیچیده است چرا که تست ها خیلی مرتبط به پیاده سازی هستند.