به نام خدا

ایجاد مبتنی بر آزمون (TDD)

آزمایش سوم

اهداف

هدف از انجام این آزمایش، آشنایی با ضرورت آزمون نویسی و آشنایی با رویکرد Test-Driven Development در آزمون نرم افزار است.

پیش نیازهای آزمایش

آشنایی با زبان جاوا و مقداری آشنایی با ابزار آزمون JUnit برای انجام این آزمایش ضروری است.

همچنین لازم است که پروژه‌ی اولیه‌ی درس را از صفحه‌ی گیت‌هاب درس دریافت نمایید.

https://github.com/ssc-public/Software-Engineering-Lab/tree/main/base-projects/AccountManagementTDD

شرح آزمایش

شرح برنامه

در این آزمایش، یک برنامه ساده به منظور محاسبات مالی تراکنش‌های بانکی به شما داده شده است. در این برنامه، یک کلاس به نام Account Balance Calculator وجود دارد که در آن موارد زیر موجود است :

- لیستی از تاریخچه‌ی تراکنش‌های محاسبه شده.
- متدی برای محاسبه‌ی وضعیت نهایی حساب بانکی.

- این متد یک لیستی از تراکنش‌های مالی را دریافت کرده و بر اساس آن، موجودی نهایی حساب بانکی را محاسبه می‌کند.
- برای سادگی آزمایش، موجودی حساب به عنوان سقف تراکنش در نظر گرفته نمیشود. اما برداشت از حساب نباید به گونه‌ای باشد که منجر به موجودی منفی شود.
- پس از محاسبه (و نه قبل از آن) نتیجه‌ی آخرین محاسبه باید در قالب لیست تاریخچه قابل بازیابی باشد.

روش انجام کار و پرسش‌ها

در ابتدا، برنامه را دانلود کرده و تست‌ها را اجرا نمایید. در صورتی که برنامه را به درستی اجرا کنید، تمامی موارد آزمون باید پاس شوند.

بخش اول - کشف خطا

شاید از پاس شدن تست‌ها خوشحال شویم :) اما باید دقت کنید که در این کد، یک خطا وجود دارد که در این موارد آزمون مورد ارزیابی قرار نگرفته‌است.

- **پرسش اول - در این که چه خطایی وجود دارد؟ و به نظر شما چرا دیده نشده‌است؟ (در دو خط توضیح دهید).**
- **پرسش دوم - پس از یافتن خطا، یک آزمون برای آن بنویسید که منجر به کشف آن خطا شود. سپس آن را به گونه‌ای اصلاح کنید که آن مورد آزمون، پاس شود. (مورد آزمون مردود و همچنین پاس شدن آن باید در گزارش شما مشخص باشد).**
- **پرسش سوم - بنظر شما و بر اساس تجربه‌ی بدست آمده، نوشتن آزمون پس از نوشتن برنامه، چه مشکلاتی را می‌تواند بسازد؟ (در سه خط توضیح دهید).**

بخش دوم - به کارگیری TDD

همان طور که گفته شد، علاقه مند هستیم که تاریخچه‌ی تراکنش‌های مورد استفاده در آخرین محاسبه‌ی موجودی (به معنای آخرین فراخوانی متد Account Balance Calculator) در قالب لیست تاریخچه‌ی تراکنش‌ها قابل دریافت باشد. اما این قابلیت در برنامه پیاده‌سازی نشده‌است. قصد داریم که پیاده‌سازی این روش را با رویکرد TDD انجام دهیم. بنابراین لازم است که در ابتدا، آزمون‌هایی را برای آن بنویسیم که در ابتدا fail شوند. با توجه به این که قصد داریم وقت شما فرهیختگان را زیاد نگیریم، برای شما چند مورد آزمون نوشته‌ایم و آن‌ها را کامنت کرده ایم. در ابتدا لازم است که آن‌ها را از حالت کامنت خارج کنید. بدیهی است که این موارد آزمون fail میشوند و لازم است که تغییراتی را در کلاس AccountBalanceCalculator بدهید که آن موارد آزمون پاس شوند.

سپس به پرسش های زیر پاسخ دهید :

پرسش چهارم - نوشتن موارد آزمون پیش از کدنویسی، چگونه کار ساخت برنامه را (نسبت به روش قبل که در بخش قبل انجام داده اید) تسهیل کرد؟

پرسش پنجم - در نهایت به نظر شما، روش ایجاد مبتنی بر آزمون، چه مزایا و معایبی دارد؟ لطفا بر اساس تجربه بدست آمده به این سوال پاسخ دهید (این سوال پاسخ صحیح مشخص ندارد).

خروجی‌های مورد درخواست و نحوه‌ی تحویل

در نهایت لازم است که لطف کنید و خروجی نهایی کار خود را در یک مخزن عمومی گیت‌هاب بارگذاری کنید. برنامه‌ی شما بایستی حاوی تغییرات خواسته شده باشد. توجه کنید که تحت هیچ شرایطی نباید موارد آزمون را تغییر دهید (در صورت تغییر موارد آزمون، نمره‌ی کل بخش عملی صفر خواهد شد). تنها تغییرات مجاز عبارتند از:

1. افزودن مورد آزمون خواسته شده.
2. خارج کردن سه مورد آزمون آخر از کامنت.

در کنار کد داده شده، گزارش شما باید حاوی گزارشی از تغییرات و گزارشی از موارد آزمون پاس شده باشد و لازم است که تمامی موارد آزمون نوشته شده، پاس شده‌باشند؛ در پایان گزارش، بایستی به پنج سوال نهایی، به صورت مختصر و مفید پاسخ دهید.

پیروز، شاد و تندرست باشید