آزمون نرم‌افزار

تمرین شماره ۱

مدرس: دکتر احسان خامس‌پناه

طراحان: امیرعلی وحیدی، پاشا براهیمی

مهلت تحویل: دوشنبه ۳۰ مهر ۱۴۰۳، ساعت ۲۳:۵۹

مقدمه

هدف از این تمرین کسب آشنایی با آزمون واحد (Unit Testing) و چارچوب JUnit می‌باشد. آزمون واحد یک مرحله در آزمون نرم‌افزار می‌باشد که در آن کوچک‌ترین بخش‌های قابل آزمایش یک برنامه، به طور جداگانه، برای عملکرد صحیح مورد بررسی قرار می‌گیرند. همچنین JUnit یک چارچوب آزمون نرم‌افزار در زبان جاوا می‌باشد که به برنامه‌نویسان این امکان را می‌دهد تا با یک روش استاندارد آزمون‌های خود را بنویسند و آن‌ها را اجرا کنند.

بخش پیاده‌سازی

برای شروع، یک پوشه به نام "test/java/mizdooni" در پوشه "src" ایجاد کنید. خوب است آزمون‌هایی که برای کلاس‌های مختلف می‌نویسید را بر اساس نام کلاس در فایل‌های جداگانه پیاده‌سازی کنید. به عنوان مثال، اگر قصد دارید برای کلاس "User" از پوشه "model" آزمون بنویسید، یک پوشه به نام "model" در "test/java/mizdooni" ایجاد کنید و یک کلاس به نام "UserTest" در آن قرار دهید.

در ادامه، آزمون‌های واحد مربوط به متدهای موجود در کلاس‌های **User** و **Table** و **Restaurant** و **Rating** را پیاده‌سازی کنید. توجه داشته باشید که بعضی از متدها قابلیت آزمون شدن با روش Parameterized را دارند که در این صورت از این روش استفاده کنید. اطمینان حاصل کنید که کارکردهای مختلف این متدها به درستی و حداقل یک‌بار آزموده شده باشند.

در پیاده‌سازی آزمون‌ها به موارد زیر توجه کافی داشته باشید:

- طبق اسلایدهای درس حتما از چارچوب JUnit5 استفاده کنید.
- از نوشتن آزمون‌های تکراری خودداری کنید.
- سعی کنید آزمون‌ها را مستقل از یکدیگر بنویسید.
- سعی کنید هر آزمون را فقط برای آزمودن یک بخش از کد پیاده‌سازی کنید.

توجه کنید که نیازی به نوشتن آزمون برای constructor و یا متدهای getter و setter که هیچ منطقی در آن‌ها پیاده‌سازی نشده، وجود ندارد.

بخش تئوری

سوال اول

در مورد متدهای Assume و به طور خاص کاربرد متد AssumeTrue تحقیق کنید و به طور خلاصه نتیجه را بنویسید.

سوال دوم

فرض کنید قصد داریم Thread Safe بودن کلاس ReviewService را بررسی کنیم. برای مثال ممکن است رفرنس آبجکتی در این کلاس در Threadهای مختلف برای خواندن و نوشتن استفاده شود. به نظر شما می‌توانیم از آزمون واحد برای اطمینان از درستی یک کد Multi-threaded استفاده کنیم؟

سوال سوم

ایراد استفاده از کنسول برای پرینت خروجی آزمون (مانند شبه‌کد زیر) به جای استفاده از Assert را بیان کنید.

```
@Test
public void testA() {
  Integer result = new SomeClass().firstMethod();
  System.out.println("Expected result is 10. Actual result is " + result);
}
```

سوال چهارم

مشکلات هر یک از آزمون‌های زیر را در صورت وجود بیان کنید و راه‌حل(های) احتمالی برای تصحیح هر یک از آن‌ها را ارائه دهید.

الف)

```
@Test
public void testB() expects Exception {
    int badInput = 0;
    new AnotherClass().process(badInput);
}
```

ب)

```
public class TestCalculator() {
    Calculator fixture = Calculator.getInstance();

    @Test
    public void testAccumulate() {
      fixture.setInitialValue(0);
      int result = fixture.accumulate(50);
      Assertions.assertEquals(50, result);
    }

    @Test
    public void testSubsequentAccumulate() {
      int result = fixture.accumulate(100);
      Assertions.assertEquals(150, result);
    }
}
```

نکات پایانی

- پروژه در قالب گروه‌های **حداکثر دو نفره** انجام می‌شود.
- برای پیاده‌سازی ابتدا پروژه را از این لینک clone کرده و سپس یک مخزن[1] در صفحه شخصی خود به صورت خصوصی[2] ایجاد کرده و تغییرات لازم را بر روی آن اعمال کنید.
- کاربر **SWT-UT** را به مخزن خود اضافه کنید.
- پاسخ سوالات بخش تئوری را در قالب یک فایل PDF در صفحه درس بارگذاری کنید. توجه داشته باشید که نیازی به ذکر کدهای بخش پیاده‌سازی در این فایل نیست؛ تنها لازم است در ابتدای این فایل، **آدرس مخزن** و **شناسه آخرین کامیت** خود را بنویسید.
- برای تحویل کافی‌ست یکی از اعضای گروه فایل PDF را در صفحه درس بارگذاری نماید.
- هدف از این تمرین، یادگیری شماست؛ لطفا تمرین را خودتان انجام دهید. در صورت مشاهده مشابهت بین کدهای دو گروه، از نمره هر دو گروه مطابق سیاست ذکر شده در کلاس، کسر خواهد شد.

[1] Repository

[2] Private