تمرین اول[1]

آشنایی با JUnit

در این تمرین می‌خواهیم با پیاده‌سازی چند آزمون واحد در فریم‌ورک JUnit، با این چهارچوب در عمل آشنا شویم. برای این کار مخزن تمرین اول را دریافت کرده و از آخرین کامیت شاخه‌ی اصلی برای پاسخ به سوالات استفاده نمایید.

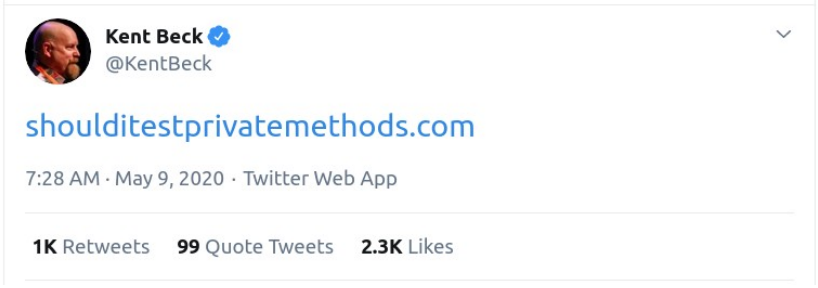
1. تست‌های مربوط به کلاس Pet را در فایل PetTest.java پیاده‌سازی نمایید. برای سنجش کفایت تعداد تست‌ها ملاک پوششی مد نظر نیست، کافی است کارکرد(های) هر متد حداقل یک‌بار تست شده باشد.

2. برای کلاس پیش‌گفته:
   a. متد getVisitsBetween را با روش Theories تست نمایید.
   b. متد getVisitsUntilAge را با روش Parameterized تست نمایید.

   طراحی تست‌های معنی‌دار و بکارگیری درست روش خواسته شده (مثلا بکارگیری assume و assert های مورد نیاز) و نام‌گذاری مناسب برای متدها و کلاس‌های تعریف‌شده مطلوب است.

در پاسخ به سوالات زیر، در صورتی که حین پیاده‌سازی به آن برخورد کردید، آن را در کدتان کامنت کنید یا در این جا به راهکارتان اشاره نمایید.

3. تست کردن متدهای Private بحث‌برانگیز است. این مساله را تحلیل کنید و اگر در این تمرین به آن برخورد کردید بگویید تصمیم شما چه بود و چرا؟
   کنت بک (از پیشروان TDD) اخیرا با اشتراک این لینک در توییتر نظر خودش را در این باره تصریح کرد. با جستجوی بیشتر می‌توانید نظرات دیگران را نیز در این زمینه مشاهده نمایید.



4. فرض کنید می‌خواهیم از Thread Safe بودن کلاس پیش‌گفته اطمینان حاصل کنیم. مثلا ممکن است رفرنس یک شی در Threadهای مختلف برای خواندن و نوشتن استفاده شود و نیاز داریم اطمینان حاصل کنیم به خطای ConcurrentModificationException برنمی‌خوریم. توضیح دهید آیا می‌توان با یونیت تست از درستی یک کد Multi-Thread مطمئن شد؟ (پیاده‌سازی تست امتیازی است).

---

1 طراحان: کیارش آذرنیا، محمدعلی عادلی‌منش و محمدرضا یزدانی‌فر

5. در شبه‌کد تست‌های زیر از Assert استفاده نشده است. آیا هر کدام از تست‌ها اشکالی دارند؟ چرا؟

```
@Test
public void testA() {
   Integer result = new SomeClass().aMethod();
   print(“expected result is 10. Actual result is ” + result);
}
```

```
class Foo() {
   public void bar() throws Exception {
      Integer result = doProcess();
      return result;
   }
}
@Test
public void testB() {
   new Foo().bar();
}
```

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

نکات

- هش آخرین کامیت ریپازیتوری و فایل گزارش خود را در محل مربوطه ثبت نمایید. کاربر ut_itst را در گیت‌لب به مخزن پرایوت خود اضافه نمایید. برای این کار پیشنهاد ما این است که ابتدا مخزن تمرین را Fork کنید و سپس تنظیمات لازم را در Settings ریپازیتوری گیت‌لب خود انجام دهید.
- پیدا کردن باگ در کلاس مربوطه حائز نمره امتیازی خواهد بود.
- نمره‌ی این پروژه تا ۱۰۵٪ قابل محاسبه است.
- در این تمرین، هم‌افزایی (اشتراک ایده، لینک‌های مفید، مشارکت در بحث‌های مربوطه، کمک به حل مشکلات حاشیه‌ای مثلا کانفیگ لازم برای IDE) در گروه کلاس توصیه می‌شود اما همکاری (اشتراک کد و پاسخ سوالات) تنها در گروه دو نفره‌ی تعریف‌شده قابل انجام است.