مدرس: دکتر احسان خامس‌پناه
طراحان: امیرعلی وحیدی، پاشا براهیمی

بخش تئوری

سوال اول

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

پاسخ:

متدهای Assume به طور کلی شروط اجرای یک تست را بیان می‌کنند و مشخص می‌کنند که یک تست در چه حالتی معنادار خواهد بود. در واقع، در صورتی که شرایط وجود نداشته باشند، به این معنی نیست که کد مشکلی دارد؛ بلکه به این معنی است که تست در این شرایط هیچ اطلاعات مفیدی در اختیار ما قرار نمی‌دهد. به طور خلاصه، می‌توان گفت متدهای Assume مشخص می‌کنند که در شرایط فعلی یک تست اجرا شود یا خیر.

متد AssumeTrue نیز یکی از پرکاربردترین متدهای این گروه است که صحیح بودن یک عبارت boolean-ای را به عنوان شرط اجرای تست بررسی می‌کند.

سوال دوم

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

پاسخ:

در حالی که با استفاده از یک تست یونیت fail شده می‌توانیم نشان دهیم که یک کلاس Thread Safe نیست، با هر تعداد تست یونیت pass شده نمی‌توانیم Thread Safe بودن یک کلاس را اثبات کنیم. این مورد به این دلیل است که اپلیکیشن‌های Multi-Threaded می‌توانند در تعداد بی‌شماری زمان‌بندی مختلف اجرا شوند که بررسی همه این زمان‌بندی‌ها از عهده unit testing خارج است.

برای بررسی Thread Safe بودن یک کلاس می‌توان از روش‌های صوری[1] مانند Model Checking استفاده کرد که به کمک روابط ریاضی، می‌توانند یک خاصیت را در یک تکه کد اثبات کنند.

[1] Formal Methods

سوال سوم

ایراد استفاده از کنسول برای پرینت خروجی تست (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);
}
```

پاسخ:

مشکل این روش این است که امکان بررسی تست به صورت خودکار وجود ندارد و لازم است یک نفر به صورت دستی نتایج تست‌ها را بررسی کند.
این روش در صورتی که تعداد تست‌ها خیلی زیاد باشد یا در مواردی که تست‌ها در پایپ‌لاین CI/CD اجرا می‌شوند، می‌تواند مشکلات زیادی را ایجاد کند.

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);
}
```

پاسخ:

مشکل این تست این است که کلیدواژه expects در جاوا وجود ندارد و این تست به صورت کلی کامپایل نخواهد شد. برای نوشتن صحیح این تست در JUnit5 از متد assertThrows و در ورژن‌های قدیمی‌تر JUnit از آرگومان expected استفاده می‌شود.

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);
    }
}
```

پاسخ:

مشکل این تست این است که در هنگام اجرای تست، ترتیب اجرا لزوما از تست اول نخواهد بود و ممکن است تست دوم پیش از تست اول اجرا شود. در این صورت، این تست fail می‌شود.