به نام خدا



دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

**آزمون نرم افزار**

**گزارش كار شماره 1**

**محمد پويا افشاري ‌(‌810198577)**

**مصطفی ابراهیمی (810199575)**

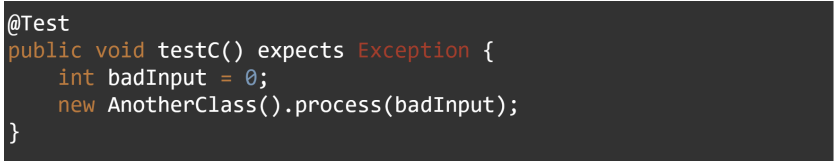
# بخش دوم - گزارش كار

1. سوال اول:

* معمولا Private method ها، به این علت استفاده می‌شوند كه جزييات آنها در دسترس كاربر ها قرار نگيرد اين در حالي است كه نوشتن Test به صورت Documentation ميتواند اصل Encapsulation اين جزييات را بشكند. در راه حل تحصين برانگيز ارايه شده توسط [jop](https://stackoverflow.com/users/11830/jop) ، به اين نكته اشاره مي‌شود كه در صورتي كه تست هاي توابع خصوصي ما به اندازه اي مهم و بزرگ باشند كه نياز حتمي به تست كردن دارند ميتوان آن را به كلاس تبديل كرده و در آنجا به صورت Public براي متد خواسته شده ‌ی قبلی تست نوشت. نمونه انجام این کار در [Method Object](https://refactoring.com/catalog/replaceFunctionWithCommand.html) موجود است. ( البته در ایراد همین روش میتوان به ایجاد Component های بدون استفاده در برنامه در صورت کاربرد بیش از حد اشاره نمود.)
* همچنین نقطه نظر‌های بیشتری در زمینه تست متد‌های Private میتوان در نظر گرفت. در مرحله Refactor کد‌ها معمولا متدهای Private بیشترین تغییر را می‌کنند.وهمین امر باعث شکننده بودن تست های نوشته شده و اضافه کردن Overhead تست می‌شود.
* برخی از زبان ها مثل جاوا به راحتی امکان تست مستقیم متدهای Private را نمی‌دهند به همین منظور در صورت استفاده از Reflection و ابزارهای مشابه برای گرفتن دسترسی عملا Bad practice انجام میدهیم.

1. سوال دوم:
2. سوال سوم:

در تست داده شده خروجي گرفته شده از متد چاپ مي‌شود در اين حالت هيچ assertion وجود ندارد كه درستي اين برابري را چك بكند. به جاي اين كار ميتوانستيم مقدار خواسته شده مورد انتظار 10 را در متغير expected نگه داري كنيم و برابري را با عمل Assertions.assertEquals(expected, result) بررسي كنيم.



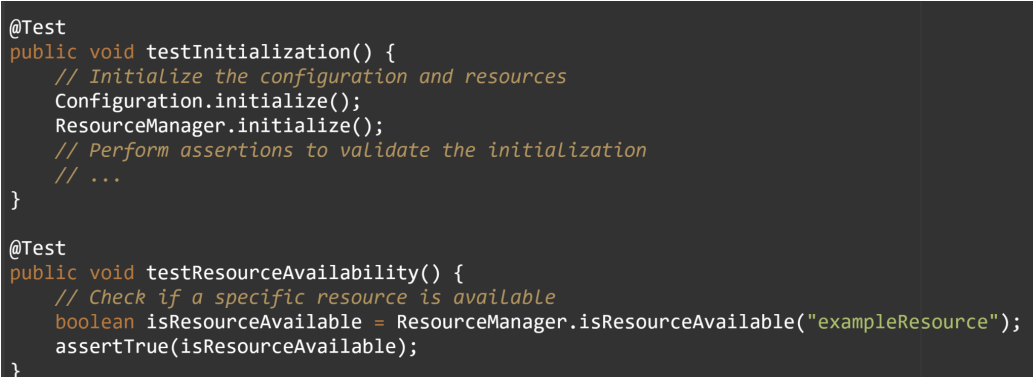
تست داده شده در اين حالت assertion خاصي ندارد براي رفع اين مشكل كه در صورتي كه ورودي بدي بدهد خروجي يك Exception درست Throw بكند بايد از assertThrows استفاده كنيم. براي اين كار به جاي خط new ميتوانيم از

Assertions.assertThrows(Exception.class, () -> {

new AnotherClass().process(badInput);

});

استفاده بكنيم.



در این مورد testInitialization اين طور به نظر ميرسد كه محيطي را setup مي‌كند كه تست resourseAvailability از آن استفاده مي‌کند که این امر مصداق chained بودن می‌باشد. به جاي اين كار ميتوان از beforeEach استفاده كرد.