تمرین دوم آزمون نرم‌افزار[1]

قسمت اول

آشنایی با چهارچوب تزریق وابستگی (Dependency Injection)

یکی از روش‌های توسعه کد قابل تست (testable) استفاده از تزریق وابستگی است که می‌تواند با استفاده از الگوی Inversion of Control انجام گیرد.

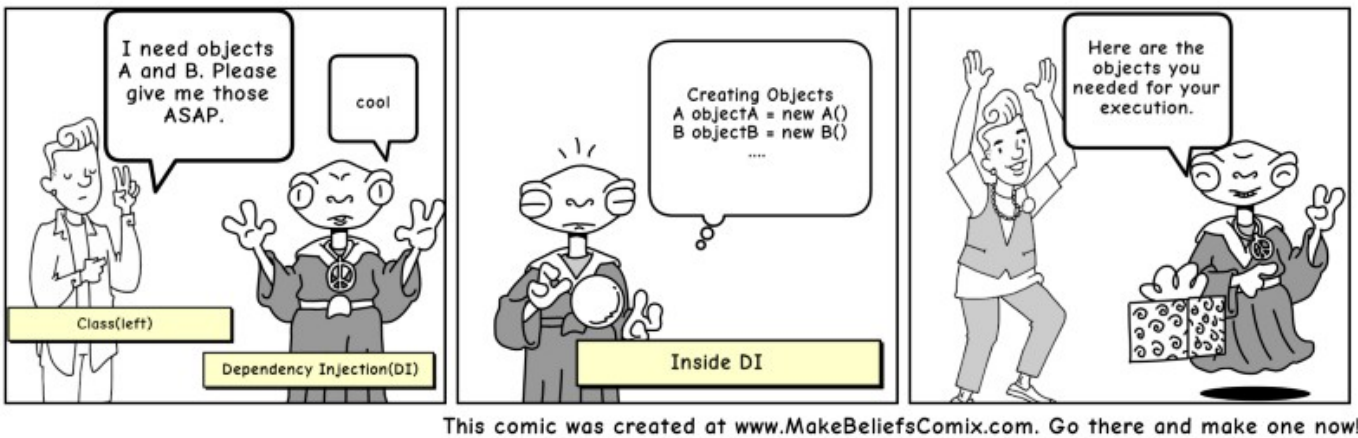
1. کلاس SimpleDI را ببینید. این کلاس به عنوان یک ابزار ساده DI مفروض است. آن را پیاده‌سازی کنید تا تست‌های آن در کلاس SimpleDITest پاس شوند (راهمایی: می‌توانید از HashMap در پیاده‌سازی خود استفاده نمایید).

توجه نمایید که با این کار ایده‌ی اساسی DI را پیاده کرده‌اید اما یک ابزار DI معمولا ویژگی‌های مهم دیگری از جمله ساختن گراف بدون دور وابستگی اشیاء و Topological Sort آن و نیز امکان Scope Management را هم، برای ساختن اشیاء بصورت خودکار شامل می‌شود.

2. الگوهای مختلفی برای تزریق وابستگی وجود دارد. برخی معتقدند نوع اول بهتر است. راجع به مقایسه آن‌ها تحقیق نمایید (نیازی به نوشتن نیست).

    1. Dependency Injection by Constructor
    2. Dependency Injection by Setter
    3. Dependency Injection by Field

یکی از چهارچوب‌های تزریق وابستگی Spring IoC Container است که از الگوی Proxy برای تزریق وابستگی‌ها استفاده می‌کند. در قسمت‌های بعدی تمرین استفاده از ابزار Spring برای DI **امتیاز** دارد هرچند می‌توانید از ابزار SimpleDI که خودتان پیاده‌سازی کرده‌اید نیز برای نوشتن تست‌ها استفاده نمایید.



1 طراحان: کیارش آذرنیا و محمدرضا یزدانی‌فر

قسمت دوم

آشنایی با بدیل‌های آزمون (Test Doubles)

در این قسمت می‌خواهیم با استفاده از کتابخانه‌ی Mockito با انواع مختلف بدیل‌های آزمون و رویکردهای مختلف صحت‌سنجی (Verification) آشنا شویم.

می‌دانیم الگوهای مختلفی برای **صحت‌سنجی** وجود دارد از جمله: State Verification و Behavior Verificaition. برای مطالعه بیشتر می‌توانید به فصل مربوطه در کتاب مراجعه نمایید.

از طرف دیگر دو رویکرد برای طراحی و پیاده‌سازی آزمون وجود دارد: Mockisty و Classical. برای مطالعه بیشتر می‌توانید به مقاله فاولر در این زمینه مراجعه نمایید.

در پیاده‌سازی تست‌های این قسمت، موارد زیر را در کدتان کامنت کنید؛

1. با توجه به تعاریف ارایه‌شده در درس، هر کدام از Double های مورد استفاده از چه نوع هستند.
   Dummy Object, Stub, Spy, Mock, Fake Object
2. هر Test Case چه نوع صحت‌سنجی انجام می‌دهد.
3. هر Test Case با چه رویکردی پیاده‌سازی شده است.

1. کلاس PetService را تست نمایید. فرض کنید فراخوانی Logger در این کلاس حیاتی است و باید از آن اطمینان حاصل کنیم. توجه نمایید که ملاک پوشش خاصی مدنظر نیست و تست کردن کارکردهای کلاس کفایت می‌کند.
2. در مورد تست کلاس PetTimedCache توضیح دهید که با چه رویکردی باید تست شود؟ چرا؟ پیاده‌سازی تست‌های این کلاس **امتیازی** است.

نکات

- حتما پروژه را از ریپازیتوری آن دریافت نمایید و قبل از انجام از بروز بودن آن نسبت به شاخه‌ی مربوط به تمرین در مخزن اطمینان حاصل کنید.
- هش آخرین کامیت ریپازیتوری و فایل گزارش خود (در صورت نیاز) را در محل مربوطه ثبت نمایید. کاربر uttest را 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 خود انجام دهید.
- پیدا کردن باگ در کلاس مربوطه حائز نمره امتیازی خواهد بود.
- در این تمرین ۱۰٪ امتیاز هست و نمره‌ی این تمرین 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.