دانشکده مهندسی کامپیوتر

برنامه‌سازی پیشرفته
تمرین های سری پنجم

مدرس: سید صالح اعتمادی
مستند : امید میرزاجانی *

مهلت ارسال:
شنبه ۳۰ فروردین ۹۹

فهرست مطالب

*با تشکر از آقای علی حیدری که در نیم سال دوم سال تحصیلی ۹۷-۹۸ این تمرین را تهیه فرمودند.

۱ مقدمه

۱.۱ موارد مورد توجه

- توجه داشته باشید که برای کسب نمره‌ی قبولی درس کسب حداقل نصف نمره‌ی هر سری تمرین الزامی می‌باشد.
- مهلت ارسال پاسخ تمرین تا ساعت ۲۳:۵۹ روز اعلام‌شده است. توصیه می‌شود نوشتن تمرین را به روزهای نهایی موکول نکنید.
- هم‌کاری و هم‌فکری شما در حل تمرین مانعی ندارد، اما پاسخ ارسالی هر کس حتما باید توسط خود او نوشته شده باشد.
- مبنای درس، اعتماد بر پاسخ ارسالی از سوی شماست؛ بنابراین ارسال پاسخ در ریپازیتوری گیت شما به این معناست که پاسخ آن تمرین، توسط شما نوشته شده است. در صورت تقلب یا اثبات عدم نوشتار پاسخ حتی یک سوال از تمرین، برای هر دو طرف تقلب‌گیرنده و تقلب‌دهنده نمره‌ی **مردود** برای درس در نظر گرفته خواهد شد.
- توجه داشته باشید که پاسخ‌ها و کدهای مربوط به هر مرحله را بایستی تا قبل از پایان زمان مربوط به آن مرحله، در سایت Azure DevOps (طبق توضیحات کارگاه‌ها و کلاس‌ها) بفرستید. درست کردن `Pull request` و `Complete` کردن `Pull request` و انتقال به شاخه‌ی `master` پس از تکمیل تمرین فراموش نشود!
- پس از پایان مهلت ارسال تا ۲ روز به ازای هر روز تاخیر ۱۰ درصد از نمره مربوط به تمرین کسر خواهد شد و پس از ۲ روز نمره‌ای به تمرین تعلق نخواهد گرفت.
- بعضی از قسمت های تمرین نیاز به پیاده سازی بر روی هر چهار زبان `C#` ، `Python` ، `C++` و `Java` را دارند بعضی هم خیر. بنابراین روبروی هر سوال زبان های مورد نیاز برای پیاده سازی مشخص شده است.

۲ آماده سازی های اولیه

۱.۲ ساخت پروژه ی C#

برای ایجاد پروژه C# کافی است کد زیر را در ترمینال خود اجرا کنید:

```
mkdir A5_cs
cd A5_cs
dotnet new sln
mkdir A5_cs
cd A5_cs
dotnet new console
cd ..
dotnet sln add A5_cs\A5_cs.csproj
mkdir A5_cs.Tests
cd A5_cs.Tests
dotnet new mstest
dotnet add reference ..\A5_cs\A5_cs.csproj
cd ..
dotnet sln add A5_cs.Tests\A5_cs.Tests.csproj
```

۲.۲ قواعد نام گذاری

قواعد نام‌گذاری تمرین را از جدول ۱ مطالعه کنید.

* در کل یک دیرکتوری داخل Assignments به نام A۵ بسازید و داخل آن،یک دیرکتوری به نام A۵_cs داشته باشید و فایل های مربوطه را داخل دیرکتوری مربوطه بگذارید.

جدول ۱: قراردادهای نام‌گذاری تمرین

Naming conventions		
Branch	Directory	Pull Request
`fb_A5`	`A5`	`A5`

۳ پیاده‌سازی تمرین

انجام این تمرین علاوه بر درک مفهوم Exception نیاز به مقدار قابل توجهی دیباگ کردن و آزمون و خطا دارد.

۱.۳ `ThrowIfOdd`

متد `ThrowIfOdd` را به گونه‌ای پیاده‌سازی کنید که در صورتی که عدد n ورودی فرد باشد یک استثنا [۱] از نوع `InvalidDataException` پرتاب [۲] کند. ۲۳۴

۲.۳ `ExceptionHandler.ctor`

سازنده‌ی این کلاس را به گونه‌ای پیاده‌سازی کنید که در صورتی که متغیر ورودی `causeExceptionInConstructor` آن true باشد استثنائی از نوع `NullReferenceException` رخ دهد. دقت کنید که شما مجاز به ساخت استثنا جدید و پرتاب آن نیستید بلکه باید استثنا در زمان اجرا رخ دهد. ۵۳۱

۳.۳ `ExceptionHandler.Input.Get`

`getter` کلاس `ExceptionHandler` را به گونه‌ای پیاده‌سازی کنید که در صورت null بودن `Input` منجر به رخ دادن استثنائی از نوع `NullReferenceException` شود. ۸۲۸

۴.۳ `ExceptionHandler.Input.Set`

`setter` کلاس `ExceptionHandler` را به گونه‌ای پیاده‌سازی کنید که در صورت null بودن value منجر به رخ دادن استثنائی از نوع `NullReferenceException` شود.

۵.۳ `IndexOutOfRangeExceptionMethod()`

متد `IndexOutOfRangeExceptionMethod()` را به گونه‌ای پیاده‌سازی کنید که منجر به رخ دادن استثنائی از نوع `IndexOutOfRangeException` در آن شود.

در صورتی که مقدار `DoNotThrow` برابر false باشد باید استثنا پس از catch شدن مجددا پرتاب شود. در غیر این صورت باید مقدار `ErrorMsg` برابر با رشته‌ی exception Caught به علاوه‌ی نوع استثنا رخ داده شود. مثلا در این جا: exception Caught IndexOutOfRangeException

۶.۳ `FormatExceptionMethod()`

متد `FormatExceptionMethod` برای مثال پیاده‌سازی شده و شما می‌توانید از این متد به عنوان راهنمایی برای پیاده‌سازی سایر متدها استفاده کنید

۷.۳ `FileNotFoundExeptionMethod()`

متد `FileNotFoundExeptionMethod` را به گونه‌ای پیاده‌سازی کنید که منجر به رخ دادن استثنائی از نوع `FileNotFoundExeption` در آن شود.

در صورتی که مقدار `DoNotThrow` برابر false باشد باید استثنا پس از catch شدن مجددا پرتاب شود. در غیر این صورت باید مقدار `ErrorMsg` برابر با رشته‌ی exception Caught به علاوه‌ی نوع استثنا رخ داده شود.

[1] Exception
[2] throw

۸.۳ `OutOfMemoryExceptionMethod()`

متد `OutOfMemoryExceptionMethod` را به گونه‌ای پیاده‌سازی کنید که منجر به رخ دادن استثنائی از نوع `OutOfMemoryException` در آن شود.
در صورتی که مقدار `DoNotThrow` برابر false باشد باید استثنا پس از catch شدن مجددا پرتاب شود. در غیر این صورت باید مقدار `ErrorMsg` برابر با رشته‌ی exception Caught به علاوه‌ی نوع استثنا رخ داده شود.

۹.۳ `OverflowExceptionMethod()`

متد `OverflowExceptionMethod` را به گونه‌ای پیاده‌سازی کنید که منجر به رخ دادن استثنائی از نوع `OverflowException` در آن شود.
در صورتی که مقدار `DoNotThrow` برابر false باشد باید استثنا پس از catch شدن مجددا پرتاب شود. در غیر این صورت باید مقدار `ErrorMsg` برابر با رشته‌ی exception Caught به علاوه‌ی نوع استثنا رخ داده شود.

۱۰.۳ `MultipleExeptionMethod()`

متد `MultipleExeptionMethod()` را به گونه‌ای پیاده‌سازی کنید که منجر به رخ دادن استثناهائی از نوع `IndexOutOfRangeException` و `OutOfMemoryException` در آن شود.
در صورتی که مقدار `DoNotThrow` برابر false باشد باید استثنا پس از catch شدن مجددا پرتاب شود. در غیر این صورت باید مقدار `ErrorMsg` برابر با رشته‌ی exception Caught به علاوه‌ی نوع استثنا رخ داده شود.

۱۱.۳ `FinallyBlockMethod()`

برای پیاده‌سازی این متد علاوه بر تسلط به مفهوم و چگونگی رفتار `try-catch-finally` ، لازم است تست‌های زیر را با دقت مطالعه و دیباگ کنید.

- `TestFinallyBlockException`
- `TestFinallyBlockNoExceptionNoReturn`
- `TestFinallyBlockExceptionNoCatch`
- `TestFinallyBlockExceptionNoCatch`

توجه کنید که همانند قسمت‌های قبلی تست‌ها به هیچ وجه نباید هیچ تغییر کنند. هدف از دیباگ کردن تست‌ها فهم رفتار متدهای مربوطه می‌باشد. با توجه به پارامترهای سازنده کلاس `ExceptionHandler` و پارامتر ورودی متد `FinallyBlockMethod` رفتار این متد متفاوت است. با مطالعه این تست‌ها متوجه پارامترهایی که رفتار این متد را تغییر می‌دهند می‌شوید. سپس بدنه متد را به گونه‌ای پیاده‌سازی کنید که تست‌های یکی پس از دیگری پاس شوند. برای کمک به شما مقداری از این متد پیاده‌سازی شده است. چنانچه علاقمند به آزمون سخت‌تری از توانایی خود دارید، بدنه موجود را پاک کرده و از ابتدا پیاده‌سازی کنید. به فیلد `FinallyBlockStringOut` و چگونگی استفاده از آن‌ها در خود متد و تست‌ها نیز دقت کنید. یکی از روش‌های راستی‌آزمایی رفتار این متد استفاده از این فیلد است.

۱۲.۳ `NestedMethods`

هدف از این تست (معما) علاوه بر تمرین و تسلط به مطالعه دقیق کد و دیباگ کردن، راست‌آزمایی تسلط شما به رفتار استثناء و `try-cath` می‌باشد. وقتی یک استثناء پرت می‌شود در درون خود اطلاعات مسیر پرتاب یا افتادن را در فیلد `StackTrace` ذخیره می‌کند. با توجه به محتوای تست `NestedExceptionTest` معلوم می‌شود که متدهایی با نام‌های `MethodA` ، `MethodB` ، `MethodC` ، و `MethodD`
باید درست شوند و استثنایی با نوع «مناسب» و از محل «مناسب» به‌گونه‌ای پرتاب شود که متدهای بالا در مسیر آن قرار گیرند.

موفق و سلامت باشید.