دانشکده‌ی مهندسی کامپیوتر

برنامه‌سازی پیشرفته (سی‌شارپ)
تمرین‌های سری پانزدهم

علی حیدری
استاد: سید صالح اعتمادی

مهلت ارسال: ۲۸ تیر ۱۳۹۸

فهرست مطالب

[1] Repository

۱ مقدمه و آماده‌سازی

۱.۱ نکات مورد توجه

- انجام این تمرین امتیازی است اما کدر صورت حل مفاهیم زیادی را از آن خواهید آموخت.
- 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 پایانی 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 ازپایان 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.
- برای طرح سوال و پرسش و پاسخ از صفحه درس در Quera استفاده کنید.

۲.۱ آماده‌سازی‌های اولیه

قواعد نام‌گذاری تمرین را از جدول ۱ مطالعه کنید.

جدول ۱: قراردادهای نام‌گذاری تمرین

Naming conventions					
Branch	Directory	Solution	Project	Test Project	Pull Request
`fb_A15`	`A15`	`A15`	`A15`	`A15Tests`	`HW15`

۱.۲.۱ آماده‌سازی‌های مربوط به git

✓ ابتدا به شاخه‌ی `master` بروید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A15)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

✓ تغییرات انجام‌شده در مخزن `Repository` را دریافت کنید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
$ git pull
remote: Azure Repos
remote: Found 8 objects to send. (90 ms)
Unpacking objects: 100% (8/8), done.
From https://9752XXXX.visualstudio.com/AP97982/_git/AP97982
   e7fd3b5..2cc74de  master               -> origin/master
Checking out files: 100% (266/266), done.
Updating e7fd3b5..2cc74de
Fast-forward
 .gitattributes                          |    63 +
 A15/A15.sln                             |    37 +
 A15/A15/A15.csproj                      |    61 +
 A15/A15/App.config                      |     6 +
 A15/A15/Program.cs                      |    15 +
 A15/A15/Properties/AssemblyInfo.cs      |    36 +
```

```
.
.
.
```

✓ یک شاخه‌ی جدید با نام `fb_A15` بسازید و تغییر شاخه دهید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
$ git checkout -b fb_A15
Switched to a new branch 'fb_A15'
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A15)
$
```

توصیه می‌شود پس از پیاده‌سازی هر کلاس تغییرات انجام شده را `commit` و `push` کنید.

۲.۲.۱ آماده‌سازی‌های مربوط به visual studio

یک پروژه‌ی جدید طبق قراردادهای نام‌گذاری موجود در جدول ۱ در ریشه‌ی ریپازیتوری git خود بسازید.
ساختار فایل پایه‌ای که در اختیار شما قرار می‌گیرد به صورت زیر است:

```
A15
    +---A15
    |   |   LogEntry.cs
    |   |   Logger.cs
    |   |
    |   +---FileNamePolicy
    |   |       FileNamePolicy.cs
    |   |       ILogFileNamePolicy.cs
    |   |       IncrementalLogFileName.cs
    |   |       TimeBasedLogFileName.cs
    |   |       WeekdayLogFileName.cs
    |   |
    |   +---Formatters
    |   |       ConsoleLogFormatter.cs
    |   |       CsvFormatter.cs
    |   |       CsvFormatterNew.cs
    |   |       ILogFormatter.cs
    |   |       XmlLogFormatter.cs
    |   |       XsvFormatter.cs
    |   |
    |   +---Loggers
    |   |       ConsoleLogger.cs
    |   |       DBLogger.cs
    |   |       FileLogger.cs
    |   |       FileLoggerFactory.cs
    |   |       ILogger.cs
    |   |
    |   +---LogWriters
    |   |       ConcurrentLogWriter.cs
    |   |       GuardedLogWriter.cs
    |   |       LockedLogWriter.cs
    |   |       LockedQueueLogWriter.cs
    |   |       NoLockLogWriter.cs
    |   |
    |   \---Scrubbers
    |           AbstractScrubber.cs
    |           CCScrubber.cs
    |           EmailScrubber.cs
    |           FullNameScrubber.cs
    |           IDScrubber.cs
    |           IPrivacyScrubber.cs
    |           PhoneNumberScrubber.cs
    |           PrivacyScrubber.cs
```

```
|           PrivacyScrubberFactory.cs
|
+---A15App
|   |   Program.cs
|
\---A15Test
        |   CsvLogFormatterTests.cs
        |   LogWriterPerfTests.cs
        |   PrivacyScrubberTest.cs
        \   XmlLogFormatterTests.cs
```

در فایل پایه دو پوشه وجود دارد شما باید فایل(های) موجود در پوشه‌ی Project را به پروژه‌ی اصلی (A15) و فایل(های) موجود در پوشه‌ی ProjectTests را به پروژه‌ی تست (A15Tests) اضافه کنید.

۲ هدف تمرین

هدف از این تمرین آشنایی شما با الگوهای طراحی شیء گرای زیر است:

- Factory Pattern
- Composite Pattern
- Strategy Pattern
- Singleton Pattern
- Observer Pattern

علاوه بر این مفاهیم شیء گرا به صورت کلی و مفاهیم زیر به صورت خاص مرور می‌شوند:

- Multi-Threading
- Concurrent Collections
- Event Handling

همچنین یک اصطلاحی در طراحی نرم‌افزار هست به نام Open/Close به این معنی که طراحی شما باید برای تغییرات بسته[۲] باشد و برای تعمیم باز[۳] باشد. به عبارت دیگر برای ایجاد یا اضافه کردن قابلیت‌های بیش‌تر نیاز به تغییر کد موجود نباشد. قابلیت جدید را شما در کد جدید پیاده‌سازی می‌کنید و به کد موجود طوری اضافه می‌کنید که تمام قابلیت‌های موجود دست نخورده می‌مانند. لذا برای اضافه‌کردن این قابلیت جدید در کدهای موجود هیچ اشکال[۴] ایجاد نمی‌شود. البته ممکن است در کد جدید اشکالی باشد، ولی کدهای قبلی مثل قبل درست کار می‌کنند. در این تمرین سعی شده که این مفهوم را هم به خوبی متوجه بشوید.

۳ موضوع تمرین

موضوع این تمرین نوشتن کتابخانه ثبت وقایع[۵] انعطاف‌پذیر و قابل توسعه است ثبت وقایع بخش مهمی از هر نرم‌افزار جدی است. چه در نرم افزارهایی مثل اپ‌های موبایل و چه در نرم‌افزارهای سمت سرور.
ثبت وقایع چیز پیچیده‌ای نیست. در ساده‌ترین حالت می‌شود از همان `Console.WriteLine` استفاده کرد و وقایع را در کنسول ثبت کرد. منتها اگر برنامه کنسول نداشت، یا کنسول بسته شد، دیگر تمام چیزهایی که توی کنسول نوشته شده بود غیرقابل بازیابی است. پس بهتر است که یک گزینه‌ای هم داشته باشیم که اگر لازم بود توی یک فایل بنویسد. حال که توی فایل می‌نویسد اسم فایل چگونه باشد؟ مثلا برای این که وقایع ثبت شده دفعه قبل، پس از هر بار اجرای برنامه پاک نشود می‌توان هر بار به اسم جدید برای فایل انتخاب کنیم. مثلا تاریخ/زمان را به اسم فایل اضافه کنیم. یا این که عدد ۱، ۲، ... به آخر فایل اضافه کنیم.
اگر کامپیوتری که برنامه روی آن اجرا می‌شود خراب شود چه اتفاقی برای اطلاعات ثبت شده می‌افتد؟ سابقه ثبت وقایع را از کجا پیدا کنیم؟ مثلا می‌توان که وقایع را در یک کامپیوتر دیگر هم ذخیره کنیم؟
خب حالا توی خود فایل به چه صورتی بنویسیم؟ توی یک خط بجز یه پیام، می‌توان که یک سری اطلاعات دیگر هم بنویسیم. مثلا اگر یک دانشجو اضافه شد می‌توان پیام زیر را بنویسیم:

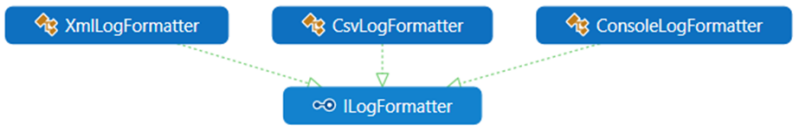
[2] close
[3] open
[4] bug
[5] logging

```
Student Added, Name: Ali Hesabi, Id: 965211212
```

خب این اطلاعات اضافه را می‌شود با ویرگول از هم جدا کرد (مثل بالا). یا به‌صورت XML:

```
<Msg>Student Added</Msg><Name>Ali Hesabi></Name><Id>965211212</Id>
```

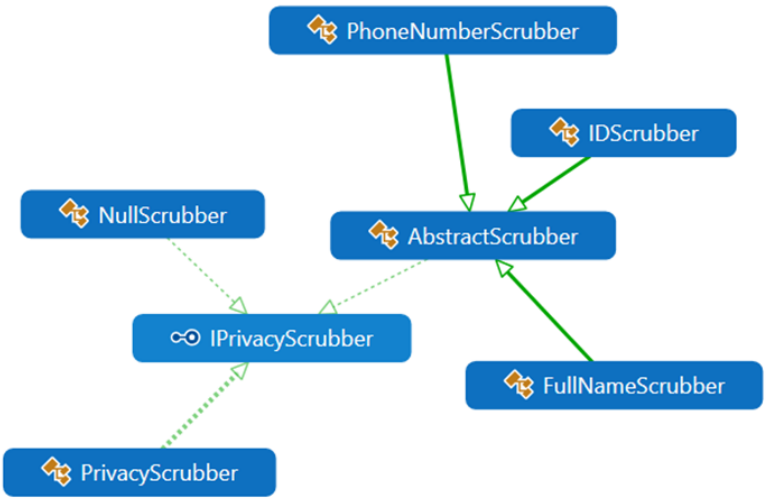
خوب، حالا آیا همه پیام‌ها مثل هم هستند؟ یا این که بعضی پیام‌ها، پیام خطا هستند، بعضی پیام هشدار، بعضی برای عیب‌یابی کردن[۶]؟ مثلا اگر برنامه را می‌خواهیم عیب‌یابی[۷] کنیم، پیام‌های عیب‌یابی را هم لازم داریم. وگرنه می‌توانیم این پیام‌ها را دیگر ضبط نکنیم. خوب است که برنامه ما این انعطاف را داشته باشد که فقط پیام‌هایی را که لازم داریم ضبط کند. یا این که پیام‌های خطا را به صورت جداگانه ذخیره کنیم.
از طرف دیگر ممکن است که نرم‌افزار ما قسمت‌های مختلف داشته باشد. مثلا بخش رابط کاربری[۸]، بخش مشتری[۹]، خدمتگذار[۱۰]، ... . خوب است که بتوانیم فقط وقایع بخش مورد نیاز را ضبط کنیم. یا وقایع را به تفکیک بخش‌ها از هم جدا کنیم.
مساله دیگر حریم خصوصی افراد است. مثلا اگر یک نرم‌افزار چت داریم، و کاربر در نرم‌افزار شماره تلفنش را به یه نفر دیگر می‌دهد، آیا جایز است که ما این پیام را به طور کامل ضبط کنیم؟ بسته به قوانین، ممکن است که این کار جایز نباشد. برای مثال GDPR[۱۱] را جست‌وجو کنید و ببینید که تعریف PII[۱۲] در کشورهای مختلف چه تفاوتی می‌کند. بنابراین بسته به این که این نرم‌افزار ثبت وقایع شما در چه کشور(هایی) سرویس می‌دهد لازم است که اطلاعات خصوصی افراد را از ثبت وقایع حذف کند.
یک مساله دیگر هم چند نخی[۱۳] است. آیا نرم افزار ما فقط یک نخ[۱۴] دارد. و اگر بیش از یک نخ دارد، اگر بیش از یک نخ در آن واحد بخواهند توی فایل ثبت وقایع بنویسند، چطوری این‌ها را با هم هماهنگ کنیم. آیا هنگام نوشتن توی فایل از Lock استفاده کنیم، که ممکن است سرعت برنامه را کم کند؟ یا این که اطلاعات ثبت وقایع را در یک لیست اضافه کنیم و در یک نخ دیگر این‌ها را توی فایل بنویسیم.
هدف ما این است که یک کتابخانه قابل انعطاف بنویسیم که تمام موارد بالا را پوشش بدهد و در عین حال قابلیت تعمیم هم داشته باشد. هدف دیگری هم که داریم این است که استفاده از این کتابخانه ساده باشد. یعنی لازم نباشد برای درست کردن یک شیء Logger تعداد زیادی پارامتر استفاده کنیم. در عین حال اگر لازم شده که استفاده کنیم بتوانیم.
مثل تمرین‌های قبل ما برنامه را طراحی و پیاده‌سازی ابتدایی شده است. برای تمرین قسمت‌هایی از پیاده‌سازی را حذف کردیم. کار شما این است که کلاس‌ها و روابط آن‌ها را مطالعه کنید و قسمت‌های پیاده‌سازی نشده را پیاده‌سازی کنید برای فهم بهتر کلاس دیاگرام را آورده شده است.



شکل ۱

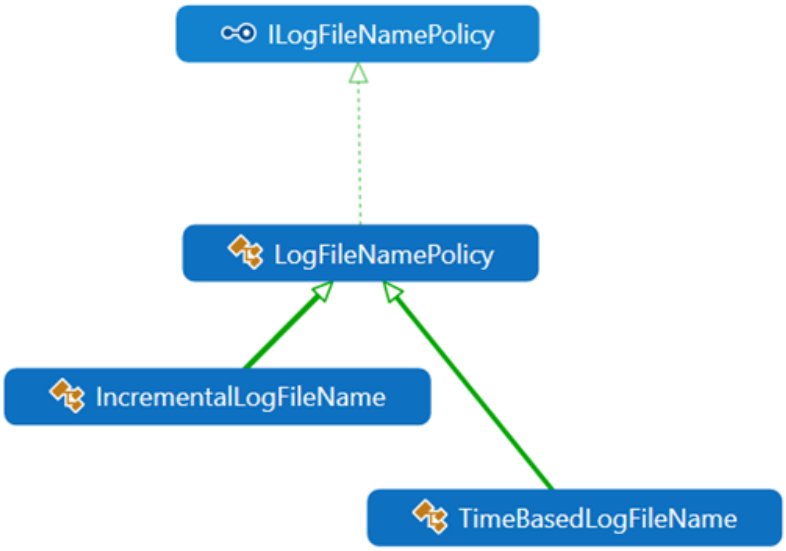
دیاگرام بالا مربوط به LogFormatter ها است. هدف از این طراحی پشتیبانی فرمت‌های مختلف فایل است. و این که همان طور که قبلا اشاره شد، شما بتوانید بدون تغییر کد، آن را تعمیم دهید. بعد انعطاف‌پذیر بعدی این طراحی در رابطه با حذف اطلاعات حریم خصوصی افراد است.

[6] debuging
[7] debug
[8] UI
[9] client
[10] server
[11] General Data Protection Regulation
[12] Personally Identifiable Information
[13] MultiThreading
[14] Thread



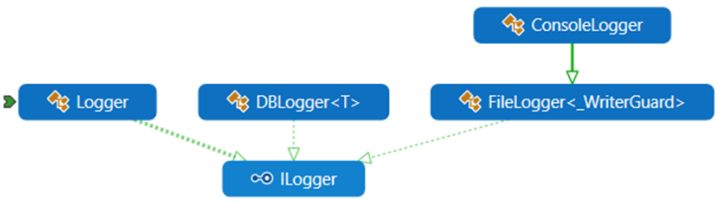
شکل ۲

دیاگرام زیر مربوط به نحوه نام‌گذاری فایل های ثبت وقایع است.



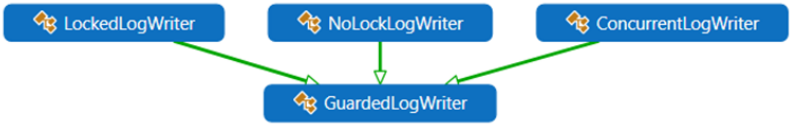
شکل ۳

کلاس‌های زیر با هدف انعطاف‌پذیری در محل ثبت وقایع طراحی شده‌اند (مثلا کنسول، فایل، پایگاه داده، ...). کلاس پایگاه داده ثبت وقایع (DBLogger) فقط برای کامل بودن و کمک کردن به فهم مطلب این‌جا گذاشته شده ولی در این تمرین پیاده سازی نمی‌شود.



شکل ۴

نهایتا بسته به محیط استفاده از این کتابخانه روش‌های مختلفی برای جلوگیری از بروز اشکال در محیط چند نخی طراحی کردیم.



شکل ۵

۴ پیاده‌سازی تمرین

۱. پروژه‌ها و شاخه‌های لازم را همان طور که در بالا آمده ایجاد/اضافه کنید.

۲. برای فهم بهتر این کتابخانه، برنامه `Console` را اجرا کنید. موارد زیر فقط برای کمک کردن به فهم این کتابخانه است. لازم نیست کدی برای این‌ها کامیت شود.

(آ) خطوط چاپ شده در کنسول را با فایل‌های ایجاد شده در `c:\log` مقایسه کنید. شباهت و فرق فایل ها و خطوط چاپ شده چیه؟

(ب) چگونه است که ما فقط یک بار دستور `Log` را زدیم ولی این خطوط هم توی فایل ذخیره شدند هم توی کنسول؟

(ج) برنامه را دیباگ کنید ببینید که چرا مثلا شماره تلفن توی صفحه چاپ شده ولی توی فایل نیست؟

(د) برنامه را دیباگ کنید ببیند چرا بعضی از خطوط در صفحه نمایش هست ولی در فایل `a15_error_0000.log` یا `a15_ui_0000.log` نیستند.

(ه) خوب یک `logger` جدید درست کنید مثل `allLogger` که اسم و شماره ملی را حذف کند، ولی شماره تلفن رو حذف نکند. این را به `Logger.Loggers` اضافه کنید و برنامه را اجرا کنید.

(و) خوب حالا یک `logger` دیگه شبیه `allLogger` درست کنید که فقط خطوطی که مربوط به مشتری هستند را ثبت کند. مجدد به `Logger.Loggers` اضافه کنید و برنامه را اجرا کنید.

۳. تست‌های موجود را اجرا کنید. همگی بجز `NoLockPerfTest` باید با موفقیت اجرا بشوند.

(آ) علت خطای این تست را پیدا کنید. یک کامنت بگذارید و علت را توضیح بدهید و بعد کل این تست را کامنت کنید که دیگر اجرا نشود.

(ب) تست‌های `ConcurrentLogWriterPerfTest` و `LockedLogWriterPerfTest` را پیدا کنید و مطالعه کنید ببینید چکار می‌کنند. این دو تست را با تعداد نخ ۱، ۲، ۵، ۱۰، ۲۰، ۵۰، ۱۰۰ اجرا کنید (تعداد پیام ها همان هزار تا باشد). زمانی که هر تست طول می‌کشد برای تعداد نخ مساوی را در یک جدول در قسمت کامنت‌ها اضافه کنید. جدول باید هفت تا ستون و دو تا ردیف داشته باشد.

(ج) کلاس‌های `ConcurrentLogWriter` و `LockedLogWriter` و نحوه استفاده از آن‌ها مطالعه کنید و علت تفاوت در زمان را توضیح دهید.

(د) کلاس جدید به نام `LockedQueueLogWriter` پیاده‌سازی کنید. این کلاس شبیه `ConcurrentLogWriter` می‌باشد ولی از `ConcurrentQueue` استفاده نمی‌کند. از `Queue` معمولی استفاده می‌کند و باید برای `Enqueue` ، `Dequeue` از lock استفاده کنید.

(ه) یک تست جدید برای `LockedQueueLogWriter` اضافه کنید به نام `LockedQueueLogWriterPerfTest` و کارایی این کلاس را در مقایسه با دو کلاس دیگر که در بالا تست کردید ارزیابی کنید و در ردیف سوم جدولی که قبلا درست کردید اضافه کنید.

۴. تست‌های موجود در `PrivacyScrubberTest` را مطالعه کنید و با نحوه اجرای آن‌ها آشنا شوید. انجام این قسمت با استفاده از Regular Expression ها راحت‌تر است. برای آشنایی می‌توانید به لینک‌های زیر مراجعه کنید:

(a) https://www.codeproject.com/Articles/9099/The-Minute-Regex-Tutorial
(b) http://regexstorm.net/tester

اگر به اشکال برخوردید در گروه سوال کنید، راهنمایی می‌کنیم. البته بدون `Regular Expression` هم می‌توان انجام داد، ولی این‌گونه آسان‌تر است.

(آ) کلاس `IDScrubber` در حال حاضر فقط شماره ملی های با فرمت مثلا `1212-23-234` را حذف می‌کند. این کلاس را تغییر بدهید به طوری که شماره ملی‌های ایرانی را هم تشخیص بدهد و پاک کند `UnitTest` اضافه کنید که نشان بدهد این کار را درست انجام داده‌اید.

(ب) یک کلاس جدید به نام `EmailScrubber` شبیه `IDScrubber` پیاده‌سازی کنید که ایمیل‌ها را حذف کند . `UnitTest` اضافه کنید که نشان بدهد درست کار می‌کند.

(ج) یک کلاس جدید به نام `CCScrubber` درست کنید که شماره کارت اعتباری شانزده رقمی را حذف کند. مجددا `UnitTest` به تعداد کافی اضافه کنید که نشان بدهد درست پیاده‌سازی کرده‌اید.

(د) الگوی `Singleton` را برای این کلاس‌های جدید هم پیاده‌سازی کنید.

(ه) متد `PrivacyScrubberFactory.ScrubAll` را بروز کنید که این دو نظیف‌کننده[۱۵] جدید را هم داشته باشد.

(و) دو تا `FactoryMethod` جدید به نام `ScrubNumbers` و `ScrubLetters` پیاده‌سازی کنید که `scrubber` های متناسب را داشته باشند (اسم و ایمیل از حروف تشکیل شده‌اند و بقیه از عدد).

۵. یک `ILogFormatter` جدید به نام `XsvFormatter` درست کنید شبیه `CsvFormatter` با این تفاوت که به جای استفاده از ویرگول بین فیلدها از یک کاراکتر دل‌خواه که در سازنده داده می‌شود استفاده کند. `CsvFormatter` را تغییر بدهید که از `XsvFormatter` به ارث ببرد و کارکتر ویرگول را استفاده کند. طراحی این دو کلاس باید به شکلی باشد که کد تکراری نداشته باشید. الگوی `Singleton` را برای `XsvFormatter` پیاده‌سازی نکنید.

۶. خب تا حالا این کتابخانه را از ابعاد مختلفی گسترش دادید. حال نوبت اسم فایل ها است. یک کلاس جدید به نام `WeekdayLogFileName` پیاده‌سازی کنید شبیه `IncrementalLogFileName` با این تفاوت که بجای شماره از روز هفته (Sat, Mon, Tue, Wed, Thu, Fri, Sun) استفاده می‌کند. و اگر در یک روز دو بار اجرا شد به انتهای فایل قبلی اضافه می‌کند.

۷. خوب برگردید توی تابع `Main` . یک `callback` برای `event` `Logger.Instance.OnLog` تعریف کنید که تعداد کاراکترهای Debug، Error، Info، Warn را در انتهای اجرای برنامه چاپ کند.

۸. خوب حالا یه نگاهی به این تابع `Main` بکنید. کمی پیچیده است. به طور خاص به سازنده کلاس `FileLogger` . می‌خواهیم آن‌را یک مقداری ساده کنیم.

(آ) یک کلاس ایستا[۱۶] `FileLoggerFactory` تعریف کنید.

(ب) اول حساب کنید که کلاس `FileLogger` چند جور می‌تواند درست باشد (۱۵۳۶ جور)؟

آ) ۴ تا `FileFormatter`
ب) ۳۲ تا `Scrubber`
پ) ۳ تا `FileNamePolicy`
ت) ۴ تا `LogWriter`

(ج) برای راحتی و قابل انجام شدن کار از نظیف‌کننده ها فقط آن سه‌تایی که برای آن‌ها `PrivacyScrubberFactory` داریم را در نظر می‌گیریم. بازهم می‌شود ۱۴۴ تا. با در نظر گرفتن موارد زیر می‌شود در کل ۳۲ تا. برای هر یک از این ۳۲ مورد در کلاس `FileLoggerFactory` یک factory method درست کنید.

آ) از `LogWriter` ها فقط `ConcurrentLogWriter` ، `NoLockWriter` را در نظر بگیرید.
ب) از `Scrubber` ها `NullScrubber` و `AllScrubber` .
پ) از `FileNamePolicy` ها `IncrementalLogFileName` و `TimeBasedLogFileName` .
ت) و همه `FileFormat` ها.

برای نمره کامل لازم است:

۱. کد شما کامل و درست باشد.

۲. بعد از تحویل تمرین UnitTest های جدید توسط حل تمرین‌ها به تمرین اضافه می‌شود و اگر آنها خطا بدهند از نمره تمرین کم می‌شود.

[15] scrubber
[16] Static

۳. بیلد مربوط به Pull Request شما موفقیت‌آمیز بوده باشد.

۴. تست‌ها همگی موفقیت‌آمیز باشند.

۵. کامنت‌های XML کامل باشد و کامنت‌ها نشانگر فهم شما از مفاهیم شیء گرا باشد.

۶. جداول و کامنت‌هایی که از شما خواستم موجود و کامل باشند.

۵ ارسال تمرین

در اینجا یک‌بار دیگر ارسال تمرینات را با هم مرور می‌کنیم:

۱.۵ مشاهده‌ی وضعیت اولیه‌ی فایل‌ها

ابتدا وضعیت فعلی فایل‌ها را مشاهده کنید:

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A15)
$ git status
On branch fb_A15
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    A15/

nothing added to commit but untracked files present (use "git add" to track)
```

همان‌طور که مشاهده می‌کنید فولدر `A15` و تمام فایل‌ها و فولدرهای درون آن در وضعیت `Untracked` قرار دارند و همان‌طور که در خط آخر خروجی توضیح داده شده برای `commit` کردن آن‌ها ابتدا باید آن‌ها را با دستور `git add` وارد `stage` کنیم.

۲.۵ اضافه کردن فایل‌های تغییر یافته به stage

حال باید فایل‌ها و فولدرهایی را که در `stage` قرار ندارند را وارد `stage` کنیم. برای این کار از دستور `git add` استفاده می‌کنیم.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A15)
$ git add A15/*
```

حال دوباره وضعیت فایل‌ها و فولدرها را مشاهده می‌کنیم:

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A15)
$ git status
On branch fb_A15
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   A15/A15.sln
    new file:   A15/A15/A15.csproj
    new file:   A15/A15/App.config
    new file:   A15/A15/Program.cs
    new file:   A15/A15/Properties/AssemblyInfo.cs
    new file:   A15/A15Tests/A15Tests.csproj
    new file:   A15/A15Tests/Properties/AssemblyInfo.cs
    new file:   A15/A15Tests/packages.config
    .
    .
    .
```

همانطور که مشاهده می‌کنید فولدر `A15` و تمام فولدرها و فایل‌های درون آن (به جز فایل‌هایی که در `gitignore` معین کرده‌ایم) وارد `stage` شده‌اند.

۳.۵ commit کردن تغییرات انجام شده

در گام بعدی باید تغییرات انجام شده را `commit` کنیم. فراموش نکنید که فقط فایل‌هایی را می‌توان `commit` کرد که در `stage` قرار داشته باشند. با انتخاب یک پیام مناسب تغییرات صورت گرفته را `commit` می‌کنیم:

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A15)
$ git commit -m "Implement HW15"
[fb_A15 c1f21df] Implement HW15
 15 files changed, 595 insertions(+)
 create mode 100644 A15/A15.sln
 create mode 100644 A15/A15/A15.csproj
 create mode 100644 A15/A15/App.config
 create mode 100644 A15/A15/Program.cs
 create mode 100644 A15/A15/Properties/AssemblyInfo.cs
 create mode 100644 A15/A15Tests/A15Tests.csproj
 create mode 100644 A15/A15Tests/Properties/AssemblyInfo.cs
 create mode 100644 A15/A15Tests/packages.config
.
.
.
```

۴.۵ ارسال تغییرات انجام شده به مخزن Remote

گام بعدی ارسال تغییرات انجام شده به مخزن `Remote` است.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A15)
$ git push origin fb_A15
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 8 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (25/25), 9.56 KiB | 890.00 KiB/s, done.
Total 25 (delta 4), reused 0 (delta 0)
remote: Analyzing objects... (25/25) (5 ms)
remote: Storing packfile... done (197 ms)
remote: Storing index... done (84 ms)
To https://9752XXXX.visualstudio.com/AP97982/_git/AP97982
 * [new branch]      fb_A15 -> fb_A15
```

۵.۵ ساخت Pull Request

با مراجعه به سایت Azure DevOps یک Pull Request جدید با نام `HW15` بسازید به طوری که امکان `merge` کردن شاخه‌ی `fb_A15` را بر روی شاخه‌ی `master` را بررسی کند. (این کار درصورتی انجام می‌شود که کد شما کامپایل شود و همچنین تست‌های آن پاس شوند) در نهایت با انتخاب گزینه‌ی `set auto complete` در صفحه‌ی Pull Request مربوطه تعیین کنید که در صورت وجود شرایط `merge` این کار انجام شود. دقت کنید که گزینه‌ی `Delete source branch` **نباید** انتخاب شود.

۶.۵ ارسال Pull Request به بازبیننده

در نهایت Pull Request ساخته شده را برای بازبینی، با بازبیننده‌ی خود به اشتراک بگذارید.