دانشکده‌ی مهندسی کامپیوتر

برنامه‌سازی پیشرفته (سی‌شارپ)
تمرین‌های سری دوازدهم (LINQ)

علی حیدری، امیرمحمد احسان‌دار، یگانه مرشدزاده
استاد: سید صالح اعتمادی

مهلت ارسال: ۱۱ خرداد ۱۳۹۸

فهرست مطالب

۱ مقدمه و آماده‌سازی ۳
۱.۱ نکات مورد توجه ۳
۲.۱ آماده‌سازی‌های اولیه ۳
۱.۲.۱ آماده‌سازی‌های مربوط به git ۳
۲.۲.۱ آماده‌سازی‌های مربوط به visual studio ۴

۲ پیاده‌سازی تمرین ۴
۱.۲ مقدمه ۴
۲.۲ پیاده‌سازی اولیه ۴
۳.۲ نمودار UML[1] ۴
۴.۲ پیاده‌سازی اولیه ۴
۵.۲ ۵
۶.۲ ۵
۷.۲ ۶
۸.۲ ۶
۹.۲ ۶
۱۰.۲ ۶
۱۱.۲ ۶
۱۲.۲ ۶
۱۳.۲ ۶
۱۴.۲ ۷

۳ ارسال تمرین ۷
۱.۳ مشاهده‌ی وضعیت اولیه‌ی فایل‌ها ۷
۲.۳ اضافه کردن فایل‌های تغییر یافته به stage ۷
۳.۳ commit کردن تغییرات انجام شده ۷
۴.۳ ارسال تغییرات انجام شده به Remote repository ۸
۵.۳ ساخت Pull Request ۸
۶.۳ ارسال Pull Request به بازبیننده ۸

واژه نامه انگلیسی به فارسی ۹

واژه نامه فارسی به انگلیسی ۹

فهرست اختصارات ۱۰

[1] Unified Modeling Language

۱ مقدمه و آماده‌سازی

۱.۱ نکات 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 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_A12`	`A12`	`A12`	`A12`	`A12Tests`	`HW12`

۱.۲.۱ آماده‌سازی‌های مربوط به git

✓ ابتدا به شاخه‌ی `master` بروید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A11)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

✓ تغییرات انجام‌شده در `Remote 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 +
 A12/A12.sln                                   |    37 +
 A12/A12/A12.csproj                            |    61 +
 A12/A12/App.config                            |     6 +
 A12/A12/Program.cs                            |    15 +
 A12/A12/Properties/AssemblyInfo.cs            |    36 +
 .
 .
```

```
.
```

✓ یک شاخه‌ی جدید با نام `fb_A12` بسازید و تغییر شاخه دهید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
$ git checkout -b fb_A12
Switched to a new branch 'fb_A12'
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
$
```

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:

```
A12
+---Project
\---ProjectTests
        GradedTests.cs
        googleplaystore.csv

2 directories, 2 files
```

در فایل پایه دو پوشه وجود دارد شما باید فایل(های) موجود در پوشه‌ی Project را به پروژه‌ی اصلی (A12) و فایل(های) موجود در پوشه‌ی ProjectTests را به پروژه‌ی تست (A12Tests) اضافه کنید.

۲ پیاده‌سازی تمرین

۱.۲ مقدمه

یک dataset از سایت www.kaggle.com در اختیار شماست که حاوی اطلاعات اولیه برنامه‌ها[۲] ی موجود در سایت play.google.com است. قرار است شما طی مراحلی اولا این فایل را به لیستی از اشیای حاوی این اطلاعات تبدیل کنید. پس از آن با انجام پرسش‌ها[۳]یی اطلاعاتی از آن استخراج کنید.

۲.۲ پیاده‌سازی اولیه

از روی نمودار ؟؟ کلاس `AppData` و `AppAnalysis` را با متدهای آن‌ها (به صورت mock) پیاده‌سازی کنید تا کد کامپایل شود و هیچ تستی پاس نشود.

۳.۲ نمودار UML

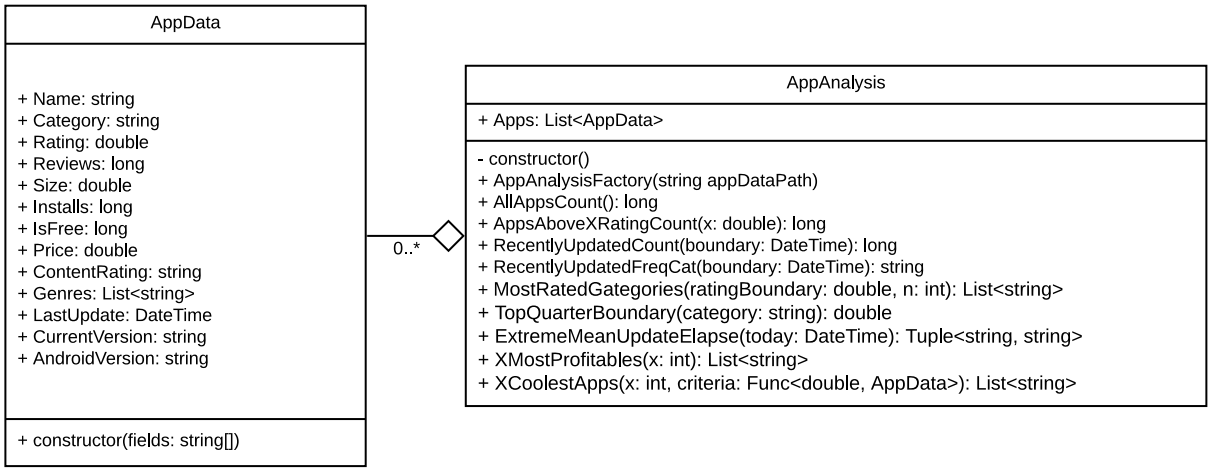
کلاس `AppData` قرار است حاوی اطلاعات یک برنامه باشد که از فایل داده‌ها جداسازی[۴] شده است. در این مرحله فقط کلاس ساخته شود.
برای آن که بدانید چه `attribute` هایی با چه نوعی باید تعریف کنید فایل `CSV`[۵] و کلاس دایاگرام را بازبینی کنید.
راهنمایی: سوالات بعدی را بخوانید! شاید لازم باشد بعضی فیلدها (مثل حجم و ژانر) با نوع دیگری ذخیره شوند.

۴.۲ پیاده‌سازی اولیه

در این قسمت قرار است از فایل ورودی یک لیست از `AppData` استخراج شود. درکلاس `AppAnalysis` متد ایستا[۶] `AppAnalysisFactory` را به گونه‌ای پیاده‌سازی کنید که ورودی آن آدرس فایل `csv` و خروجی آن یک شی از نوع `AppAnalysis` باشد. همان‌طور که در دایاگرام مشخص است در `AppAnalysis` یک `attribute` (غیر ایستا) از نوع `List<AppData>` وجود دارد که برای شی خروجی این

[2] Application
[3] Query
[4] parse
[6] Static

شکل ۱: نمودار UML



متغیر باید مقداردهی شده باشد. در این مرحله باید تست `AppAnalysisFactoryTest` پاس شود. برای جداسازی فایل `csv` از این پیوند[۷] کمک بگیرید.
لازم به ذکر است بعضی از ردیف‌ها دارای مقادیر ویژه‌ای هستند. این مقادیر را با یک مقدار پیش فرض که فکر می‌کنید به جنبه‌های آماری کار ضربه نمی‌زند جای‌گزین کنید (مثلا صفر!). تغییر دادن فایل اصلی قابل قبول نیست.
نمونه پیاده سازی متد `AppAnalysisFactory` :

```
public static AppAnalysis AppAnalysisFactory(string csvAddress)
{
  var appAnalysis = new AppAnalysis();
  using (TextFieldParser parser = new TextFieldParser(csvAddress))
  {
    parser.TextFieldType = FieldType.Delimited;
    parser.SetDelimiters(",");
    var fields = parser.ReadFields();
     while (!parser.EndOfData)
    {
      fields = parser.ReadFields();
      appAnalysis.AppendApp(fields);
    }
  }
  return appAnalysis;
}
```

راهنمایی ۱: برای کامپایل شدن کدتان باید یک Assembly Reference به نام `VisualBasic` به کدتان اضافه کنید.
راهنمایی ۲: اگر در جداسازی عبارت‌هایی مثل `10,000` به `long` مشکل دارید راه‌حل در این پیوند وجود دارد. مشکل دیگر این است که گاهی شما یک چیز غیر `double` مثل NaN[۸] را می‌خواهید جداسازی کنید! در این‌جا استفاده از `try` / `catch` یا استفاده از متد `TryParse` توصیه می‌شود.

۵.۲

تا این مرحله داده‌ها آماده‌سازی شده است. حالا قرار است با استفاده از LINQ[۹] اطلاعات آماری از این داده‌ها استخراج کنیم. بعد از حل این تمرین شما می‌دونید برنامه‌های پول‌ساز و پرطرفدار در سطح جهانی (با دیتای چند ماه قبل) کدوم هاست.
نکته: لازم نیست حل همه سوال‌ها فقط با LINQ انجام بشوند. حل سوال‌های علامت دار فقط با LINQ نمره امتیازی دارد

۶.۲

متد `AllAppsCount` را به گونه‌ای پیاده‌سازی کنید که تعداد کل برنامه‌های موجود در دیتابیس را برگرداند.

[7] Link
[8] Not a Number
[9] Language Integrated Query

پس از حل این بخش تست `AllAppsCountTest` پاس می‌شود.

۷.۲

✓متد `AppsAboveXRatingCount` را به گونه‌ای پیاده‌سازی کنید که تعداد برنامه‌ها با امتیاز[10] بالای `x` (به عنوان پارامتر ورودی) موجود در دیتابیس را برگرداند.
پس از حل این بخش تست `AppsAboveXRatingCountTest` پاس می‌شود.

۸.۲

متد `RecentlyUpdatedCount` را به گونه‌ای پیاده‌سازی کنید که تعداد برنامه‌هایی در دیتابیس را که از تاریخ `boundary` به بعد آپدیت شده‌اند برگرداند.
پس از حل این بخش تست `RecentlyUpdatedCountTest` پاس می‌شود.
رفع ابهام: برنامه‌های آپدیت شده در خود این روز نیز شمرده شود.

۹.۲

متد `RecentlyUpdatedFreqCat` را به گونه‌ای پیاده‌سازی کنید که نام پرتکرارترین دسته‌بندی[11] در برنامه‌های قسمت قبل را برگرداند.
پس از حل این بخش تست `RecentlyUpdatedFreqCatTest` پاس می‌شود.

۱۰.۲

متد `MostRatedCategories` را به گونه‌ای پیاده‌سازی کنید که بین برنامه‌های با امتیاز `x` و بالاتر `n` دسته‌بندی با بیش‌ترین تعداد برنامه را برگرداند.
پس از حل این بخش تست `MostRatedCategoriesTest` پاس می‌شود.

۱۱.۲

متد `TopQuarterBoundary` را به گونه‌ای پیاده‌سازی کنید که مرز چارک بالای (از نظر امتیاز) همه برنامه‌های دسته‌بندی `"PHOTOGRAPHY"` را برگرداند.
محاسبه با دقت خیلی بالا مد نظر نیست. همین که عدد خروجی درست باشد کفایت می‌کند.
پس از حل این بخش تست `TopQuarterBoundaryTest` پاس می‌شود.

۱۲.۲

متد `ExtremeMeanUpdateElapse` را به گونه‌ای پیاده‌سازی کنید که دسته‌بندی بیش‌ترین و کم‌ترین میانگین مدت زمان گذشتن از آخرین به‌روزرسانی[12] را برگرداند.
راهنمایی: اول بدانیم مدت زمان گذشتن از آخرین به‌روزرسانی (نسبت به روز ۲۵ می ۲۰۱۹) چه عددهایی است. بعد بر اساس دسته‌بندی، میانگین بگیریم و بعد مقادیر استخراج می‌شوند.
پس از حل این بخش تست `ExtremeMeanUpdateElapseTest` پاس می‌شود.

۱۳.۲

متد `XMostProfitables` را به گونه‌ای پیاده‌سازی کنید که لیست نام‌های `x` برنامه‌های پولی‌ای که بیش‌تر از همه پول‌ساز بوده‌اند را برگرداند. (مرتب‌سازی دوم بر اساس امتیاز)
رفع ابهام: پول‌ساز بودن از تعداد فروش و قیمت بدست می‌آید.
یکتا بودن نام‌های خروجی الزامی نیست. اما پیاده‌سازی به گونه‌ای که نام همه‌ی برنامه‌های خروجی یکتا باشد (فقط با LINQ و نه متد دارای بدنه) نمره امتیازی دارد. در این صورت تست را به صورت غیر درهم‌نهی[13] بازنویسی کنید.
پس از حل این بخش تست `XMostProfitablesTest` پاس می‌شود.

[10] Rate
[11] Category
[12] Update
[13] Hash

۱۴.۲

با `delegate` از طریق پارامتر ورودی، `x` برنامه خفن‌تر رو بیابید.
پس از حل این بخش تست `XCoolestAppsTest` پاس می‌شود.

۳ ارسال تمرین

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

۱.۳ مشاهده‌ی وضعیت اولیه‌ی فایل‌ها

ابتدا وضعیت فعلی فایل‌ها را مشاهده کنید:

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
$ git status
On branch fb_A12
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    A12/

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

همان‌طور که مشاهده می‌کنید فولدر `A12` 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.

۲.۳ 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_A12)
$ git add A12/*
```

حال دوباره وضعیت فایل‌ها و فولدرها را مشاهده می‌کنیم:

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
On branch fb_A12
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   A12/A12.sln
    new file:   A12/A12/A12.csproj
    new file:   A12/A12/App.config
    new file:   A12/A12/Program.cs
    new file:   A12/A12/Properties/AssemblyInfo.cs
    new file:   A12/A12Tests/A12Tests.csproj
    new file:   A12/A12Tests/Properties/AssemblyInfo.cs
    new file:   A12/A12Tests/packages.config
    .
    .
    .
```

همانطور که مشاهده می‌کنید فولدر `A12` و تمام فولدرها و فایل‌های درون آن (به جز فایل‌هایی که در `gitignore` معین کرده‌ایم) وارد `stage` شده‌اند.

۳.۳ 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_A12)
$ git commit -m "Implement HW12"
[fb_A12 c1f21df] Implement HW12
 15 files changed, 595 insertions(+)
 create mode 100644 A12/A12.sln
 create mode 100644 A12/A12/A12.csproj
 create mode 100644 A12/A12/App.config
 create mode 100644 A12/A12/Program.cs
 create mode 100644 A12/A12/Properties/AssemblyInfo.cs
 create mode 100644 A12/A12Tests/A12Tests.csproj
 create mode 100644 A12/A12Tests/Properties/AssemblyInfo.cs
 create mode 100644 A12/A12Tests/packages.config
.
.
.
```

۴.۳ ارسال تغییرات انجام شده به Remote repository

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

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
$ git push origin fb_A12
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_A12 -> fb_A12
```

۵.۳ ساخت Pull Request

با مراجعه به سایت Azure DevOps یک Pull Request جدید با نام `HW12` بسازید به طوری که امکان `merge` کردن شاخه‌ی `fb_A12` 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.

۶.۳ ارسال Pull Request به بازبیننده

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

واژه‌نامه انگلیسی به فارسی

A

برنامه ..... Application

C

دسته‌بندی ..... Category

H

درهم‌نهی ..... Hash

L

پیوند ..... Link

P

جداسازی ..... parse

Q

پرسش ..... Query

R

امتیاز ..... Rate

S

ایستا ..... Static

U

به‌روزرسانی ..... Update

واژه‌نامه فارسی به انگلیسی

ا

امتیاز ..... Rate
ایستا ..... Static

ب

برنامه ..... Application
به‌روزرسانی ..... Update

پ

پیوند ..... Link

ج

جداسازی ..... parse

د

درهم‌نهی ..... Hash
دسته‌بندی ..... Category

فهرست اختصارات

C

CSV ..... Comma-separated values

L

LINQ ..... Language Integrated Query

N

NaN ..... Not a Number

U

UML ..... Unified Modeling Language