



دانشکده‌ی مهندسی کامپیوتر

برنامه‌سازی پیشرفته (سی شارپ)  
تمرین‌های سری دوازدهم (LINQ)

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

مهلت ارسال: ۱۱ خرداد ۱۳۹۸

## فهرست مطالب

۳	۱	مقدمه و آماده‌سازی
۳	۱.۱	نکات مورد توجه
۳	۲.۱	آماده‌سازی‌های اولیه
۳	۱.۲.۱	آماده‌سازی‌های مربوط به git
۴	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 <sup>1</sup>
۴	۴.۲	پیاده‌سازی اولیه
۵	۵.۲	
۵	۶.۲	
۶	۷.۲	
۶	۸.۲	
۶	۹.۲	
۶	۱۰.۲	
۶	۱۱.۲	
۶	۱۲.۲	
۶	۱۳.۲	
۷	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
۸	۴.۳	ارسال تغییرات انجام شده به Remote repository
۸	۵.۳	ساخت Pull Request
۸	۶.۳	ارسال Pull Request به بازبیننده
۹		واژه نامه انگلیسی به فارسی
۹		واژه نامه فارسی به انگلیسی
۱۰		فهرست اختصارات

<sup>1</sup>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 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.
- 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.

```

1 Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A11)
2 $ git checkout master
3 Switched to branch 'master'
4 Your branch is up to date with 'origin/master'.

```

✓ تغییرات انجام‌شده در Remote Repository را دریافت کنید.

```

1 Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
2 $ git pull
3 remote: Azure Repos
4 remote: Found 8 objects to send. (90 ms)
5 Unpacking objects: 100% (8/8), done.
6 From https://9752XXXX.visualstudio.com/AP97982/_git/AP97982
7 e7fd3b5..2cc74de master -> origin/master
8 Checking out files: 100% (266/266), done.
9 Updating e7fd3b5..2cc74de
10 Fast-forward
11 .gitattributes | 63 +
12 A12/A12.sln | 37 +
13 A12/A12/A12.csproj | 61 +
14 A12/A12/App.config | 6 +
15 A12/A12/Program.cs | 15 +
16 A12/A12/Properties/AssemblyInfo.cs | 36 +
17 .
18 .

```

19

✓ یک شاخه‌ی جدید با نام fb\_A12 بسازید و تغییر شاخه دهید.

```
1 Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
2 $ git checkout -b fb_A12
3 Switched to a new branch 'fb_A12'
4 Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
5 $
```

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:

```
1 A12
2 +---Project
3 \---ProjectTests
4     GradedTests.cs
5     googleplaystore.csv
6
7 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](http://www.kaggle.com) در اختیار شماست که حاوی اطلاعات اولیه برنامه‌ها<sup>۲</sup> می‌باشد. موجود در سایت [play.google.com](http://play.google.com) است. قرار است شما طی مراحل اولی این فایل را به لیستی از اشیای حاوی این اطلاعات تبدیل کنید. پس از آن با انجام پرسش‌هایی 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<sup>۴</sup> شده است. در این مرحله فقط کلاس ساخته شود. برای آن که بدانید چه attribute هایی با چه نوعی باید تعریف کنید فایل CSV<sup>۵</sup> 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. در کلاس AppAnalysis متد ایستا<sup>۶</sup> AppAnalysisFactory را به گونه‌ای پیاده‌سازی کنید که ورودی آن آدرس فایل csv و خروجی آن 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

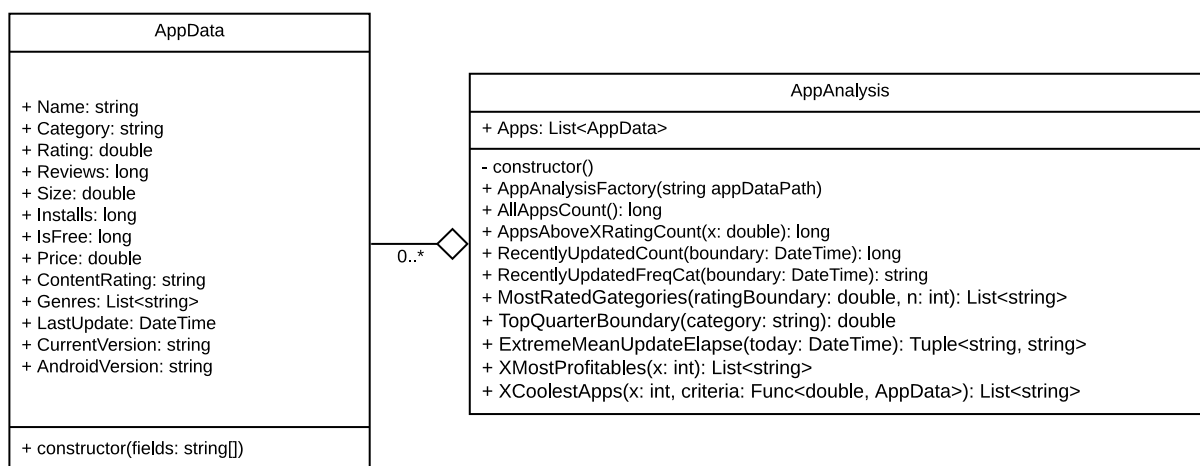
<sup>۲</sup>Application

<sup>۳</sup>Query

<sup>۴</sup>parse

<sup>۶</sup>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<sup>۷</sup> 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 (مثلاً صفر!). تغییر دادن فایل اصلی قابل قبول نیست.

نمونه پیاده‌سازی متد `AppAnalysisFactory`:

```

1 public static AppAnalysis AppAnalysisFactory(string csvAddress)
2 {
3     var appAnalysis = new AppAnalysis();
4     using (TextFieldParser parser = new TextFieldParser(csvAddress))
5     {
6         parser.TextFieldType = FieldType.Delimited;
7         parser.SetDelimiters(",");
8         var fields = parser.ReadFields();
9         while (!parser.EndOfData)
10        {
11            fields = parser.ReadFields();
12            appAnalysis.AppendApp(fields);
13        }
14    }
15    return appAnalysis;
16 }
  
```

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`<sup>۸</sup> را می‌خواهید جداسازی کنید! در این جا 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`<sup>۹</sup> 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.

<sup>۷</sup>Link

<sup>۸</sup>Not a Number

<sup>۹</sup>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<sup>۱۰</sup> 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<sup>۱۱</sup> 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<sup>۱۲</sup> را برگرداند.  
راهنمایی: اول بدانیم مدت زمان گذشتن از آخرین به‌روزرسانی (نسبت به روز ۲۵ می ۲۰۱۹) چه عددی 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<sup>۱۳</sup> بازنویسی کنید.  
پس از حل این بخش تست `XMostProfitablesTest` پاس می‌شود.

<sup>10</sup>Rate

<sup>11</sup>Category

<sup>12</sup>Update

<sup>13</sup>Hash

## ۱۴.۲

با `delegate` از طریق پارامتر ورودی، `x` برنامه خفن‌تر رو بیابید.  
پس از حل این بخش تست `XCoollestAppsTest` 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:

```
1 Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
2 $ git status
3 On branch fb_A12
4 Untracked files:
5   (use "git add <file>..." to include in what will be committed)
6
7   A12/
8
9 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.

```
1 Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
2 $ git add A12/*
```

حال دوباره وضعیت فایل‌ها و فولدرها را مشاهده می‌کنیم:

```
1 Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
2 $ git status
3 On branch fb_A12
4 Changes to be committed:
5   (use "git reset HEAD <file>..." to unstage)
6
7   new file:   A12/A12.sln
8   new file:   A12/A12/A12.csproj
9   new file:   A12/A12/App.config
10  new file:   A12/A12/Program.cs
11  new file:   A12/A12/Properties/AssemblyInfo.cs
12  new file:   A12/A12Tests/A12Tests.csproj
13  new file:   A12/A12Tests/Properties/AssemblyInfo.cs
14  new file:   A12/A12Tests/packages.config
15  .
16  .
17  .
```

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:

```

1 Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
2 $ git commit -m "Implement HW12"
3 [fb_A12 c1f21df] Implement HW12
4 15 files changed, 595 insertions(+)
5 create mode 100644 A12/A12.sln
6 create mode 100644 A12/A12/A12.csproj
7 create mode 100644 A12/A12/App.config
8 create mode 100644 A12/A12/Program.cs
9 create mode 100644 A12/A12/Properties/AssemblyInfo.cs
10 create mode 100644 A12/A12Tests/A12Tests.csproj
11 create mode 100644 A12/A12Tests/Properties/AssemblyInfo.cs
12 create mode 100644 A12/A12Tests/packages.config
13 .
14 .
15 .

```

### ۴.۳ ارسال تغییرات انجام شده به Remote repository

گام بعدی ارسال تغییرات انجام شده به Remote Repository است.

```

1 Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A12)
2 $ git push origin fb_A12
3 Enumerating objects: 25, done.
4 Counting objects: 100% (25/25), done.
5 Delta compression using up to 8 threads
6 Compressing objects: 100% (22/22), done.
7 Writing objects: 100% (25/25), 9.56 KiB | 890.00 KiB/s, done.
8 Total 25 (delta 4), reused 0 (delta 0)
9 remote: Analyzing objects... (25/25) (5 ms)
10 remote: Storing packfile... done (197 ms)
11 remote: Storing index... done (84 ms)
12 To https://9752XXXX.visualstudio.com/AP97982/_git/AP97982
13 * [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 در صورتی 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**

Application ..... برنامه

**C**

Category ..... دسته‌بندی

**H**

Hash ..... درهم‌نهی

**L**

Link ..... پیوند

**P**

parse ..... جداسازی

**Q**

Query ..... پرسش

**R**

Rate ..... امتیاز

**S**

Static ..... ایستا

**U**

Update ..... به‌روزرسانی

## واژه‌نامه فارسی به انگلیسی

**ا**

Rate ..... امتیاز

Static ..... ایستا

**ب**

Application ..... برنامه

Update ..... به‌روزرسانی

**پ**

Link ..... پیوند

**ج**

parse ..... جداسازی

د

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

## فهرست اختصارات

### C

CSV ..... Comma-separated values

### L

LINQ ..... Language Integrated Query

### N

NaN ..... Not a Number

### U

UML ..... Unified Modeling Language