دانشکده‌ی مهندسی کامپیوتر

برنامه‌سازی پیشرفته (سی‌شارپ)
تمرین‌های سری دوم (انواع داده‌ی ارجاعی و مقداری)

سید صالح اعتمادی *

مهلت ارسال: ۳۰ اسفند ۱۳۹۹

فهرست مطالب

*تشکر ویژه از آقای علی حیدری که نسخه اول این مستند را در بهار ۹۸ تهیه کردند.

۱ مقدمه و آماده‌سازی

۱.۱ نکات مورد توجه

- 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 `main` 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.
-

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_A2`	`A2`	`A2`	`A2`	`A2.Tests`	`HW2`

۱.۲.۱ آماده‌سازی‌های مربوط به git

اگر چه در گارگاه git مفاهیم و روش کار با آن آموزش داده شد اما بار دیگر در اینجا کارهایی را که باید در ابتدای تمرین انجام دهید را مرور می‌کنیم.

✓ ابتدا به شاخه‌ی `main` بروید.

```
c:\git\AP99002 (fb_A2)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

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

```
c:\git\AP99002 (main)
$ git pull
remote: Azure Repos
remote: Found 8 objects to send. (90 ms)
Unpacking objects: 100% (8/8), done.
From https://9652XXXX.visualstudio.com/AP99002/_git/AP99002
   e7fd3b5..2cc74de  main                -> origin/main
Checking out files: 100% (266/266), done.
Updating e7fd3b5..2cc74de
Fast-forward
 .gitattributes                          |    63 +
 A2/A2.sln                               |    37 +
 A2/A2/A2.csproj                         |    61 +
 A2/A2/App.config                        |     6 +
 A2/A2/Program.cs                        |    15 +
 .
 .
```

```
.
```

✓ یک شاخه‌ی جدید با نام `fb_A2` بسازید و تغییر شاخه دهید.

```
c:\git\AP99002 (main)
$ git checkout -b fb_A2
Switched to a new branch 'fb_A2'
c:\git\AP99002 (fb_A2)
$
```

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

۲.۲.۱ آماده‌سازی‌های مربوط به visual studio

ساختار فایل پایه‌ای که در اختیار شما قرار می‌گیرد به صورت زیر است:

```
+---A2
    +---A2
    +---A2.Tests
            MemoryTestTests.cs
```

فایل‌های موجود در پوشه‌ی `ProjectTests` را به پروژه‌ی تست (`A2.Tests`) اضافه (`Add`) کنید.

۲ پیاده‌سازی تمرین

۱.۲ TypeOfSize

نوع‌های داده‌ای `TypeOfSize5` ، `TypeOfSize22` ، `TypeOfSize125` ، `TypeOfSize1024` و `TypeOfSize512288` را بگونه‌ای پیاده‌سازی کنید که اندازه متغیرهای از آن نوع داده‌ای، معادل اعداد انتهای نام هر نوع داده‌ای باشد و تست `VariableSizeTest` پاس شود. ۱۱/۱

۲.۲ TypeForMaxStackOfDepth

نوع‌های داده‌ای `TypeForMaxStackOfDepth10` ، `TypeForMaxStackOfDepth100` ، `TypeForMaxStackOfDepth1000` ، `TypeForMaxStackOfDepth3000` را بگونه‌ای پیاده‌سازی کنید که حداکثر عمق بازگشتی قابل اجرا برای متدی که این نوع داده‌ای را به عنوان تنها پارامتر دریافت کند معادل اعداد انتهای نام هر نوع داده‌ای باشد و تست‌های `StackDepth100Test` ۱۰/۲، `StackDepth10Test` ۹/۳، `StackDepth1000Test` ۸/۴ و `StackDepth3000Test` پاس شوند. ۷/۵

۳.۲ TypeWithMemoryOnHeap

نوع داده‌ای `TypeWithMemoryOnHeap` را به همراه متدهای `Allocate` و `DeAllocate` به گونه‌ای پیاده‌سازی کنید که تست `HeapMemoryTest` ۶/۶ پاس شود.

۴.۲ StructOrClass

نوع‌های داده‌ای `StructOrClass1` و `StructOrClass2` و `StructOrClass3` را بگونه‌ای پیاده‌سازی کنید که تست‌های `RefValueTypeCopyTest1` ۵/۷ و `RefValueTypeCopyTest2` ۴/۸ و `RefValueTypeCopyTest3` ۳/۹ و `BoxingTest` ۲/۱۰ پاس شوند.

۵.۲ GetObjectType

درکلاس `Program` متدی `static` با مقدار بازگشتی `int` و با نام `GetObjectType` پیاده‌سازی کنید که یک شی از نوع `object` می‌گیرد به گونه‌ای که تست `TypeTest` ۱/۱۱ پاس شود.

۶.۲ FutureHusbandType

در کلاس شوهر ایده‌ال باید یه چندتا عیب داشته باشه ولی دیگه خیلی هم عیب نداشته باشه. نوع داده‌ای `FutureHusbandType` را بگونه‌ای تنظیم کرده و متد `public static bool IdealHusband(FutureHusbandType fht)` رابه گونه‌ای پیاده‌سازی کنید که تست `IdealHusbandTest` ۰/۱۲ پاس شود. برای پیاده‌سازی می‌توانید از قطعه کد زیر به عنوان راهنمایی استفاده کنید.

```
public enum FutureHusbandType : int
{
    None = /*TODO*/,
    HasBigNose = /*TODO*/,
    IsBald = /*TODO*/,
    IsShort = /*TODO*/
}
```

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:

```
c:\git\AP99002 (fb_A2)
$ git status
On branch fb_A2
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    A2/

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

همان‌طور که مشاهده می‌کنید فولدر `A2` 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.

```
c:\git\AP99002 (fb_A2)
$ git add .
```

حال دوباره وضعیت فایل‌ها و فولدرها را مشاهده می‌کنیم:

```
c:\git\AP99002 (fb_A2)
On branch fb_A2
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   A2/A2.sln
    new file:   A2/A2/A2.csproj
    new file:   A2/A2/Program.cs
    new file:   A2/A2.Tests/A2.Tests.csproj
    new file:   A2/A2.Tests/MemoryTestTests.cs
```

همانطور که مشاهده می‌کنید فولدر `A2` 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:

```
c:\git\AP99002 (fb_A2)
$ git commit -m "Implement HW2"
[fb_A2 c1f21df] Implement HW2
 15 files changed, 595 insertions(+)
 create mode 100644 A2/A2.sln
 create mode 100644 A2/A2/A2.csproj
 create mode 100644 A2/A2/Program.cs
 create mode 100644 A2/A2.Tests/A2.Tests.csproj
 create mode 100644 A2/A2.Tests/MemoryTestTests.cs
```

۴.۳ ارسال تغییرات انجام شده به Remote repository

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

```
c:\git\AP99002 (fb_A2)
$ git push origin fb_A2
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/AP99002/_git/AP99002
 * [new branch]      fb_A2 -> fb_A2
```

۵.۳ ساخت Pull Request

در نهایت باید با مراجعه به سایت Azure DevOps یک `Pull Request` جدید با نام `HW2` بسازید به طوری که امکان `merge` کردن شاخه‌ی `fb_A2` را بر روی شاخه‌ی `main` 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.