دانشکده‌ی مهندسی کامپیوتر

برنامه‌سازی پیشرفته (سی‌شارپ)
تمرین انواع داده‌ی ارجاعی و مقداری

سید صالح اعتمادی *

مهلت ارسال: ۳ فروردین ۱۴۰۳

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 Github (طبق توضیحات کارگاه‌ها و کلاس‌ها) بفرستید. چک کردن درست بودن نمره نهایی در سیستم نمره‌دهی اتوماتیک و عدم وجود مشکل در فرآیند `Compile/Build` پس از تکمیل تمرین فراموش نشود!

۲.۱ آماده‌سازی‌های اولیه

قواعد نام‌گذاری تمرین را از جدول ۱ مطالعه کنید و از طریق لینک ارائه شده عضو تمرین شوید.

جدول ۱: قراردادهای نام‌گذاری تمرین

Naming conventions			
No	Solution	Project	Test Project
01	A6	[Solution]	[Solution].Tests

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.

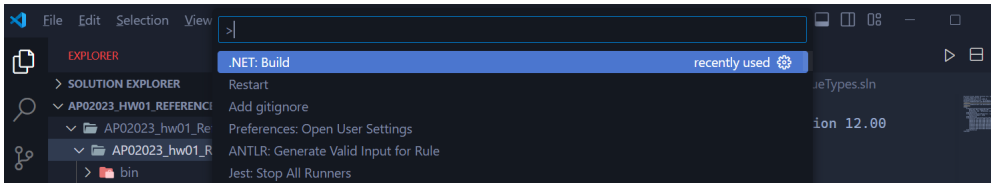
✓ بعد از عضویت در تمرین با استفاده از لینک موجود در جدول ۱ مخزن ساخته شده برای خود را `clone` کنید. نام مخزن شما به طور معمول به شکل `comp-iust/hw[No]-[ExerciseName]-[YourGithubUsername]` خواهد بود.

```
$ git clone https://github.com/comp-iust/hw[No]-[ExerciseName]-[Username].git
```

✓ افزونه C# Dev Kit را روی ویرایشگر `VSCode` نصب کنید و پروژه را باز کنید. مقداری صبر کنید تا `Solution` پروژه روی ویرایشگر بارگذاری شود. به عنوان مثال عکسی از بخش `Explorer` ویرایشگر نشان داده شده است.

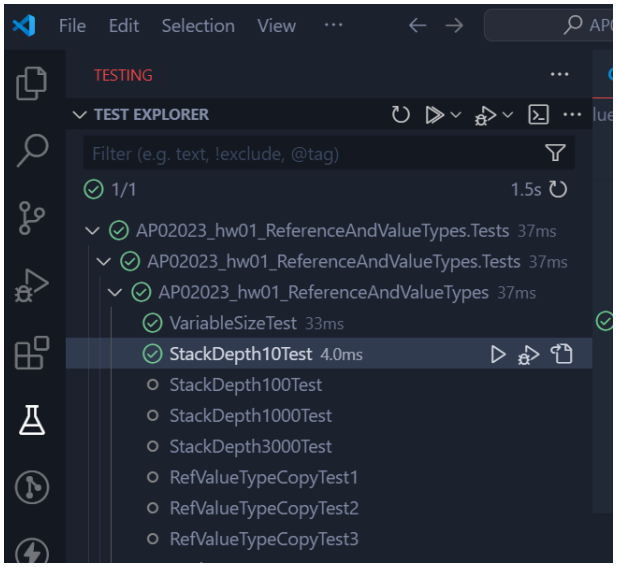
```
$ cd hw[No]-[ExerciseName]-[Username].git
$ code .
```

✓ از میانبر `CTRL + SHIFT + P` استفاده کرده و در Command Palette باز شده عبارت .Net: Build را تایپ کرده و آن را انتخاب کنید.

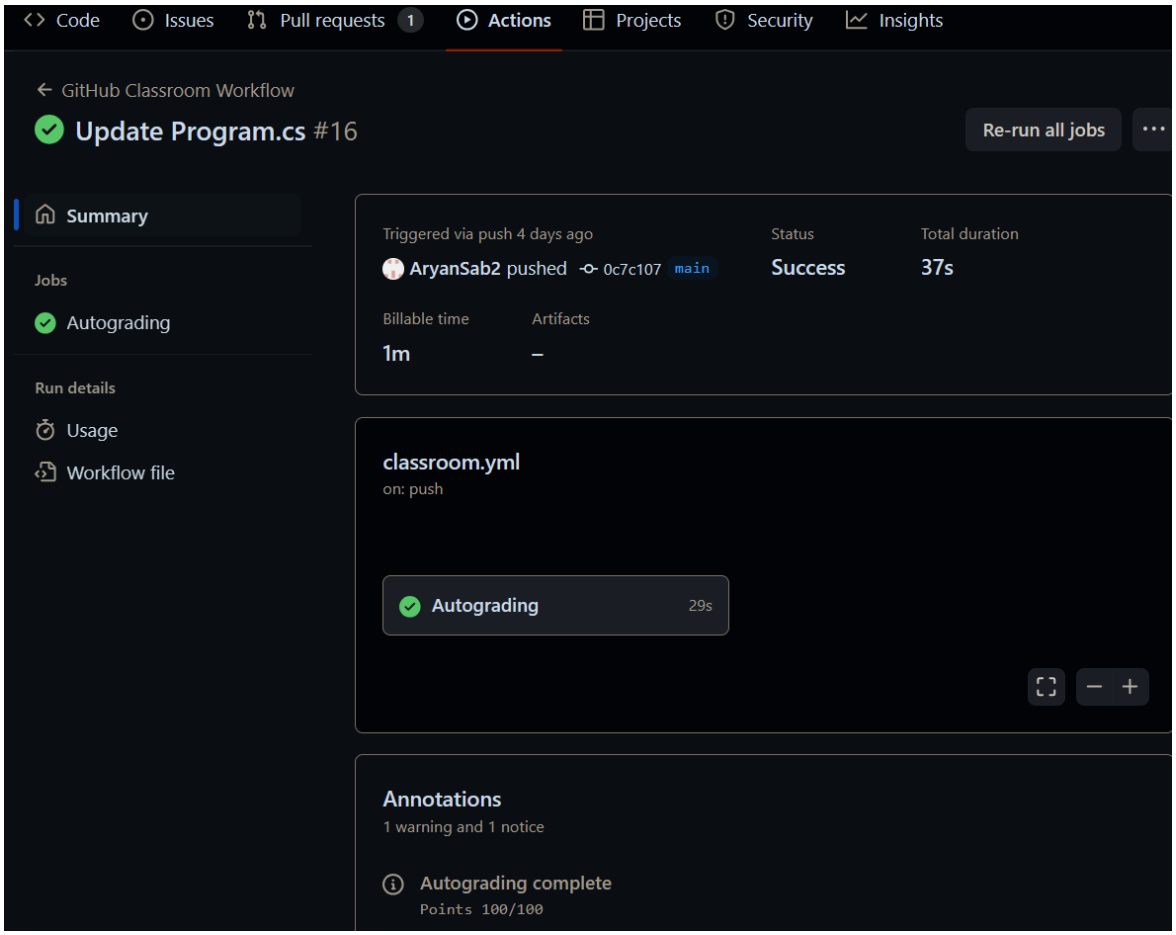


در صورت تکمیل شدن و بدون خطا بودن فرآیند `Build` خواهید توانست در بخش تست ویرایشگر، تست های تمرین را مشاهده کنید.

تست



توصیه می‌شود پس از پیاده‌سازی هر کلاس تغییرات انجام شده را `commit` و `push` کنید. سیستم نمره‌دهی اتوماتیک `Github` بعد از هر `push` کد شما را به طور اتوماتیک تصحیح خواهند کرد که در برگه‌ی `Actions` در مخزن بعد از کلیک روی `commit` مورد نظر خواهید توانست نمره خود را ببینید.



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. [11/1]

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 `StackDepth10Test` [9/3]، `StackDepth100Test` [10/2]، `StackDepth1000Test` [8/4] و `StackDepth3000Test` پاس شوند. [7/5]

۳.۲ TypeWithMemoryOnHeap

نوع داده‌ای `TypeWithMemoryOnHeap` را به همراه متدهای `Allocate` و `DeAllocate` به گونه‌ای پیاده‌سازی کنید که تست `HeapMemoryTest` [6/6] پاس شود.

۴.۲ StructOrClass

نوع‌های داده‌ای `StructOrClass1` و `StructOrClass2` و `StructOrClass3` را به‌گونه‌ای پیاده‌سازی کنید که تست‌های `RefValueTypeCopyTest1` [5/7] و `RefValueTypeCopyTest2` [4/8] و `RefValueTypeCopyTest3` [3/9] و `BoxingTest` [2/10] پاس شوند.

۵.۲ GetObjectType

در کلاس `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` [1/11] 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` [0/12] 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:

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

    ./src/Program.cs
  ./src/DefinedTypes.cs

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

همان‌طور که مشاهده می‌کنید فولدر A6 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.

```
$ git add .
```

حال دوباره وضعیت فایل‌ها و فولدرها را مشاهده می‌کنیم:

```
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/DefinedTypes.cs
        modified:   src/Program.cs
```

همانطور که مشاهده می‌کنید فولدر src و تمام فایل‌های تغییریافته یا اضافه‌شده 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:

```
$ git commit -m "Solved Successfully"
[main 8ad4c54] Solved Successfully
 2 files changed, 154 insertions(+), 3 deletions(-)
```

۴.۳ ارسال تغییرات انجام شده به Remote repository

گام بعدی ارسال تغییرات انجام شده به Remote Repository است.

```
$ git push origin main
```