دانشکده‌ی مهندسی کامپیوتر

برنامه‌سازی پیشرفته (سی‌شارپ)
پروژه پایانی

علی حیدری
استاد: سید صالح اعتمادی

مهلت ارسال: ۲۹ تیر ۱۳۹۸

فهرست مطالب

[1] Repository

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	Pull Request
fb_P1	P1	P1	P1	P1

۱.۲.۱ آماده‌سازی‌های مربوط به git

✓ ابتدا به شاخه‌ی `master` بروید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_P1)
$ 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
 .
 .
 .
```

✓ یک شاخه‌ی جدید با نام `fb_P1` بسازید و تغییر شاخه دهید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
$ git checkout -b fb_P1
Switched to a new branch 'fb_P1'
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_P1)
$
```

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. برای پروژه فایل پایه‌ای موجود نیست. لذا از ابتدا پروژه‌ای به نام (P1) از نوع `WPF Application (.NET Framework)` در ریشه گیت درست کنید.

۳.۱ توضیحات تکمیلی

دقت کنید که برای انجام پروژه نیاز به یادگیری نحوه کار و استفاده از کتابخانه‌ای مختلف از جمله WPF پیدا خواهید کرد. برای آشنایی بیشتر از منابع اینترنتی، ویدیوهای آموزشی، همکلاسی‌ها، حل‌تمرین‌ها و استاد درس استفاده کنید. کمک به همدیگر برای یادگیری شدیدا توصیه می‌شود. ولی کپی کردن کد، مطلقا ممنون است. در رابطه با تست و UnitTest انتظار می‌رود که در طول ترم به اهمیت و جایگاه آن پی برده باشید. مسلم است که داشتن UnitTest لازم است ولی اندازه و میزان آن بر عهده شماست. هنگام تحویل پروژه آمادگی توضیح در مورد علت داشتن یا نداشتن تست را داشته باشید.
در رابطه با طراحی واسط کاربر، شکل‌های ارائه شده فقط برای انتقال مفهوم می‌باشد. برای طراحی واسط کابر سعی کنید از خلاقیت خود بهره گرفته و یک طراحی منحصر به فرد ارائه کنید. برای طراحی واسط کاربر خلاقانه تا ۱۰ درصد نمره مثبت در نظر گرفته می‌شود.

۲ هدف پروژه

هدف پروژه به چند محور تقسیم می‌شود:

- در مرحله اول تمرین و کسب مهارت شروع از تصور یک برنامه کاربردی و پیاده‌سازی آن می‌باشد.
- این برنامه بدون استفاده از مفاهیم برنامه‌سازی شیئ‌گرا قابل پیاده‌سازی می‌باشد. هدف دوم از این پروژه این است که شما از مفاهیم شیء گرا به شکل مفید و صحیح استفاده کنید. مفهومی 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. هنگام طراحی و پیاده‌سازی پروژه به این نکته توجه کنید و سعی کنید حتی‌الامکان از `interface` ها در جاهای مناسب استفاده کنید.
- هدف بعدی آشنایی شما با بعضی مفاهیم و تکنولوژی‌های برنامه‌نویسی مانند طراحی واسط کاربر، `WPF` و `MultiThreading` می‌باشد.
- نهایتا با توجه به درس‌های علوم پایه ریاضی و فیزیک که اخیرا داشته‌اید، امیدواریم با پیاده‌سازی بخش تحلیل ریاضی پروژه، برخی مفاهیم ریاضی نیز برای شما جذاب‌تر و کاربردی‌تر جا بی‌افتند.

۳ ساعت عقربه‌ای شیئ گرا

بخش اول پروژه طراحی و پیاده‌سازی یک ساعت عقربه‌ای شیء گرا می‌باشد. این ساعت دارای عقربه‌های ثانیه‌شمار، دقیقه شمار و ساعت شمار می‌باشد. پیاده‌سازی یک ساعت عقربه‌ای را به دو دلیل برای شما انتخاب کردیم.

۱. با بررسی یک ساعت عقربه‌ای واقعی می‌توانید به اجزاء یا اشیاء تشکیل دهنده آن پی برده و یک طراحی شیء گرای مناسب برای آن ارائه دهید. در واقع اشیاء موجود در ساعت عقربه‌ای فیزیکی به شما ایده خوبی برای اشیاء مورد استفاده در ساعت عقربه‌ای شیئ‌گرا می‌دهد.

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

۲. برای آشنایی بیشتر و ملموس شدن مفهوم `MultiThreading` بعد از پیاده‌سازی ساعت عقربه‌ای، لازم است ساعت عقربه‌ای در کنار برنامه شما در حال کار بوده و همزمان بخش دوم برنامه که یک تحلیل‌گر ریاضی است اجرا شود.

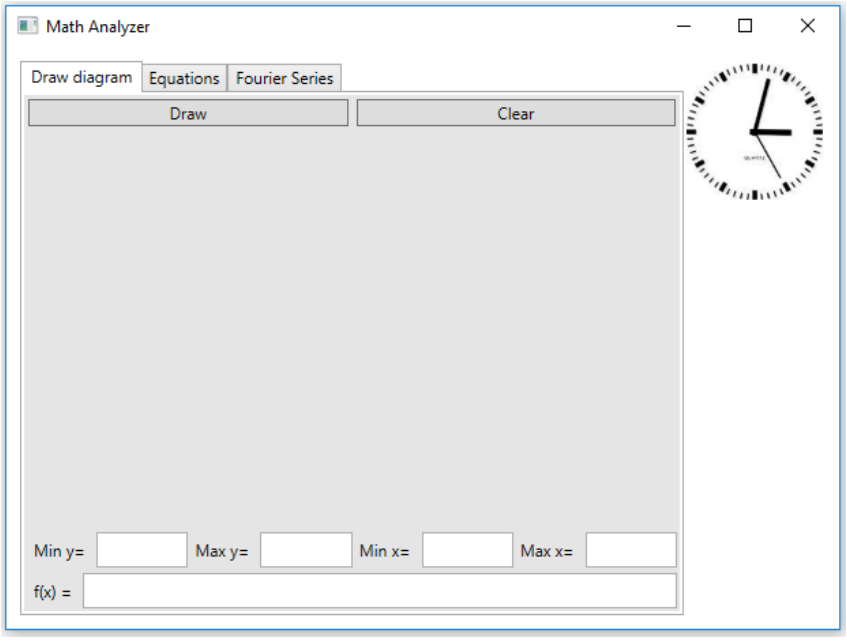
برای این پیاده‌سازی از کتابخانه‌های کشیدن خط و نقطه روی `Canvas` باید استفاده کنید. برای آشنایی بیشتر می‌توانید از کلمات کلیدی `WPF` ، `Canvas` و `Drawing` برای جستجو استفاده کنید. برای راهنمایی به موارد زیر توجه کنید.

- برای شروع با کتابخانه کشیدن خط و نقطه آشنا بشوید. مثلا یک خط بکشید. یک نقطه بگذارید یا یک دایره بکشید.
- برای حرکت عقربه ساعت لازم است عقربه را با رنگ پس‌زمینه در محل قبلی کشیده و سپس با رنگ عقربه در محل جدید بکشید. برای تمرین می‌توانید یک نقطه را از سمت چپ تا سمت راست صفحه حرکت دهید.
- برای پیدا کشیدن عقربه‌های ساعت و اعداد روی ساعت به مفاهیم تبدیل/انتقال محورهای مختصات، و استفاده از سینوس و کسینوس برای پیدا کردن x و y متناظر با انتهای عقربه، توجه کنید.
- بعد از آشنایی اولیه با شیوه کشیدن و کار کردن یک ساعت، لازم است به مفاهیم شیئ‌گرایی توجه کنید. مثلا ساعت عقربه‌ای یک شیئ است که خود دارای اشیاء دیگری مثل قاب، تعداد عدد، تعداد عقربه، ... است. در رابطه با هر کدام از این اشیاء فکر کنید که چه خصوصیاتی دارند. مثلا عقربه ساعت شمار، اندازه و ضخامت متفاوتی با عقربه دقیقه شمار و ثانیه شمار دارد. ضمن اینکه سرعت حرکت هر کدام از اینها متفاوت است. همچنین به الگوی اتفاق/event توجه کنید که چه کابردی در طراحی شما می‌تواند داشته باشد. همچنین سعی کنید که طراحی شما به گونه‌ای باشد که بتوان ساعت‌های گوناگون ساخت. مثلا ساعتی که اعداد لاتین داشته باشد. یا ساعتی که فقط در محل ساعت‌ها خط داشته باشد. یا ساعتی که فقط در محل ۳، ۶، ۹ و ۱۲ خط داشته باشد. یا ساعتی با قاب دایره‌ای، یا قاب مربعی. یا عقربه‌ها مثل پیکان باشند،... . کیفیت طراحی شیئ گرای شما نمره قابل توجهی دارد. لذا برای یک طراحی خوب تلاش کنید و فقط به اجرای صحیح برنامه بسنده نکنید.

۴ تحلیل‌گر ریاضی

با توجه به اینکه اخیرا درس‌های ریاضی و فیزیک علوم‌پایه را داشته‌اید، انتظار می‌رود که در مورد مفاهیم ریاضی حضور ذهن داشته، و از این جهت آمادگی نسبتا خوبی برای انجام پروژه پایانی با موضوع ریاضی داشته باشید.

بعد از اتمام پیاده‌سازی بخش بالا، لازم است طراحی واسط کاربر شما به گونه‌ای باشد که ساعت عقربه‌ای همیشه در گوشه راست، بالا در حال نمایش و حرکت باشد.

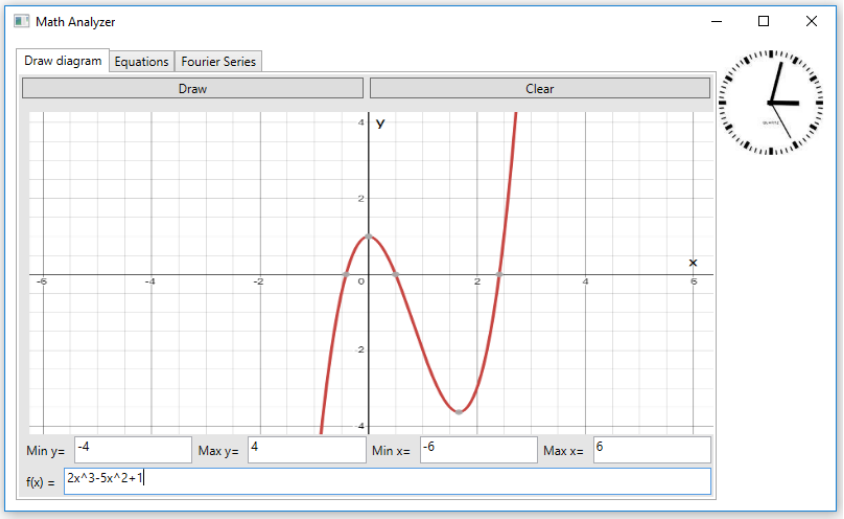


شکل ۱

لازم است تحلیل‌گر ریاضی شما دارای سه بخش زیر باشد.

۱.۴ رسم تابع

مطابق شکل ۱ این بخش برنامه امکان وارد کردن یک تابع چند جمله‌ای و حدود x و y را فراهم کرده و با فشار دادن دکمه `Draw` تابع را به محورهای مختصات با برچسب x و y و نشانه‌گذاری برای اعداد رسم می‌کند.



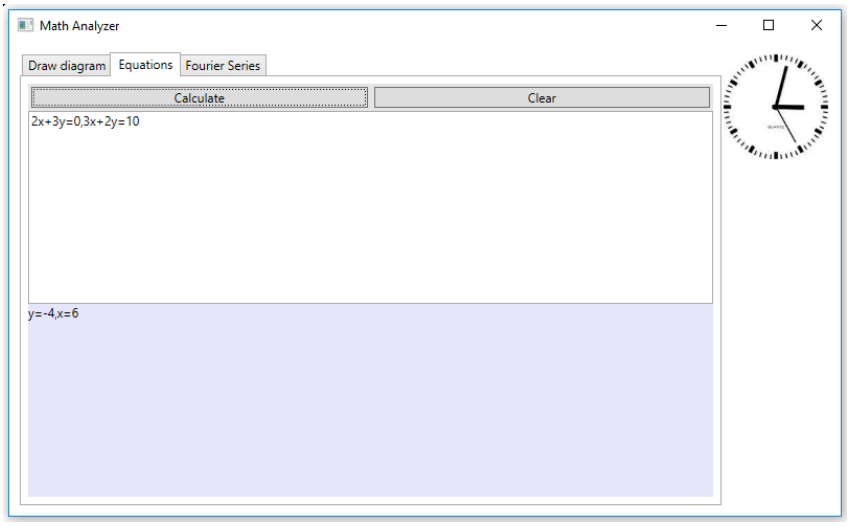
شکل ۲

برای این قسمت به موارد زیر توجه کنید.

- در درجه اول لازم است که بتوانید تابع چند جمله‌ای را `Parse` کرده و تبدیل به یک `delegate` کنید که یک عدد حقیقی به عنوان ورودی گرفته و مقدار تابع را در آن نقطه حساب کرده و برگرداند. فرمت را از روی مثال شکل ۲ ببینید. در صورت امکان استفاده از توابع بیشتر مثل سینوس/کسینوس/لاگ/... برای پروژه ۵ درصد امتیاز مثبت دریافت خواهید کرد.
- مشابه رسم ساعت در اینجا نیز لازم است برای رسم تابع، جهت و مبدا مختصات را تبدیل کنید.
- لازم است مشابه طراحی ساعت عقربه‌ای، اینجا هم به شکل مشابهی عمل کنید. به این شکل که کل صفحه رسم تابع را به عنوان یک شیئ در نظر بگیرید. این شیئ خود شامل اشیاء دیگری می‌باشد: محور مختصات x، محور مختصات y و یک یا بیشتر تابع. هر تابع از تعدادی نقطه تشکیل شده است. محور مختصات هم از یک خط، تعدادی نشانه‌گذاری و یک برچسب تشکیل شده است. حتی الامکان تمام اشیاء را بصورت جداگانه و با روابط متناسب تعریف کرده، به شکلی که اصول طراحی شیئ گرا از جمله `Single Responsibility Principle` و `Encapsulation Principle` رعایت شده باشند. با توجه به تجربه محدود شما در برنامه‌نویسی و به طور خاص برنامه‌نویسی شیءگرا لازم است بعد از اینکه برنامه شما به درستی کار می‌کند، به بازنگری و تغییر کد پرداخته تا زمانی که معماری برنامه مطابق اصول شیئگرا تغییر کند.
- توابع رسم شده قبلی فقط با فشار دادن دکمه `Clear` پاک شوند.
- لازم است با انجام عمل `drag and drop` یا نگه داشتن دکمه ماوس و حرکت آن، پس از رها کردن دکمه ماوس، تابع و محور مختصات متناسب با مقدار تغییر x و y، حرکت کرده، و محورها و تابع با توجه به این تغییر مجددا رسم شوند.
- همچنین لازم است با انجام عمل `scroll` توسط ماوس، تابع و محورها، بسته به جهت `scroll` بزرگ‌نمایی یا کوچک‌نمایی شوند.

۲.۴ حل دستگاه معادلات

در این بخش یک دستگاه معادلات از ورودی دریافت کرده و دستگاه معادلات را حل می‌کنید. برای این بخش لازم است از تمرین $A10$ با موضوع ماتریس استفاده کرده و به ماتریس امکان گرفتن معکوس را اضافه کرده و با ضرب ماتریس‌های مناسب جواب را بدست بیاورید. مطابق مثال معادلات با ویرگول یا خط جدید از هم جدا می‌شوند. تمام حروف الفبای انگلیسی از a تا z می‌توانند برای متغیر استفاده شوند. لازم است خروجی نیز مانند مثال شکل ۳ باشد که مقادیر متغیرها با ویرگول یا خط جدید از هم جدا شوند. چنانچه دستگاه معادلات جواب نداشته عبارت `No Solution` و در صورت نداشتن جواب یکتا عبارت `No Unique Solution` در خروجی چاپ شود. چنانچه قابلیت رسم خطوط و محل تقاطع آن‌ها را برای دو معادله/دو مجهول اضافه کنید، ۵ درصد امتیاز مثبت دریافت خواهید کرد.

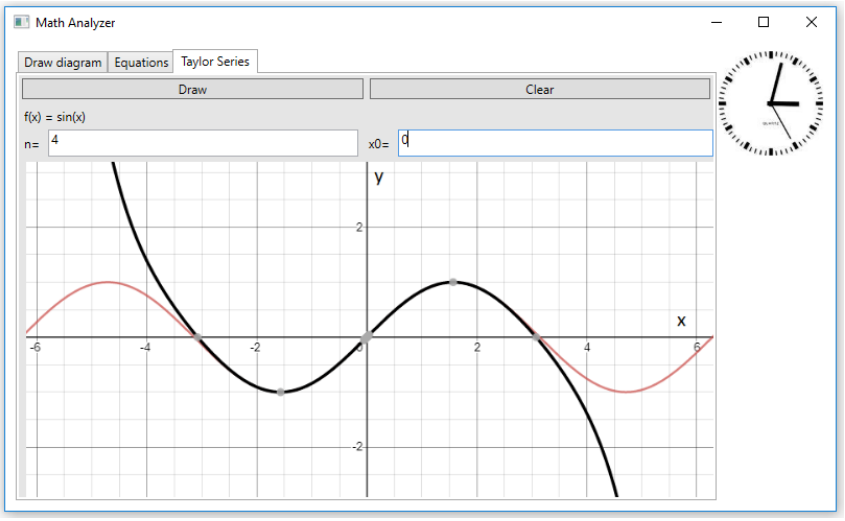


شکل ۳

۳.۴ دنباله `Taylor`

دنباله `Taylor` برای تقریب یک تابع در یک نقطه توسط یک تابع چند جمله‌ای استفاده می‌شود. در این قسمت لازم است تابع سینوس را رسم کنید. سپس در نقطه مشخص شده دنباله `Taylor` را تا جمله nام محاسبه کرده و آن را نیز رسم کنید. به عنوان مثال در شکل ۴ دنباله `Taylor` تا جمله چهارم رسم شده است.

$$x - \frac{x^{۳}}{۳!} + \frac{x^{۵}}{۵!} - \frac{x^{۷}}{۷!}$$



شکل ۴

در این بخش لازم است از اشیاء تعریف شده در قسمت اول (رسم تابع) استفاده کنید. حتی الامکان باید سعی شود که کلاس یا کد تکراری موجود نباشد. با این تفاوت که در این قسمت لازم است محدوده x و y بصورت خودکار محاسبه شود. به این شکل که محدوده x با مرکز x. و محدوده دو برابر اندازه‌ای که $\sin x$ با دنباله `Taylor` کمتر از یک درصد تفاوت دارند باشد. همچنین محدوده y دو برابر مقدار حداقل و حداکثر تابع $\sin x$ و باشد. چنانچه امکان وارد کردن تابع دلخواه بجای $\sin x$ را اضافه کنید، ۱۰ درصد نمره مثبت دریافت خواهید کرد.

۴.۴ بخش امتیازی

چنانچه وقت، حوصله و علاقه دارید هر کدام از امکانات زیر را اضافه کنید، ۱۰ درصد نمره مثبت دریافت خواهید کرد. جزئیات این بخش‌ها مشخص نشده و بر عهده شما می‌باشد.

۱. رسم تابع دلخواه به همراه شیب منحنی تابع در نقطه مشخص شده.

۲. رسم تابع دلخواه به همراه مشتق و انتگرال تابع.

۳. رسم فرکتال [۵] به صورت بازگشتی.

در مجموع امکان دریافت ۶۰ درصد (۱/۸ نمره از ۲۰ نمره) نمره مثبت برای پروژه موجود می‌باشد.

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_P1)
$ git status
On branch fb_P1
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    P1/

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

همان‌طور که مشاهده می‌کنید فولدر `P1` 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_P1)
$ git add P1/*
```

حال دوباره وضعیت فایل‌ها و فولدرها را مشاهده می‌کنیم:

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_P1)
$ git status
On branch fb_P1
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   P1/P1.sln
    new file:   P1/P1/P1.csproj
    new file:   P1/P1/App.config
    new file:   P1/P1/Program.cs
    new file:   P1/P1/Properties/AssemblyInfo.cs
    new file:   P1/P1Tests/P1Tests.csproj
    new file:   P1/P1Tests/Properties/AssemblyInfo.cs
    new file:   P1/P1Tests/packages.config
    .
    .
    .
```

همانطور که مشاهده می‌کنید فولدر `P1` و تمام فولدرها و فایل‌های درون آن (به جز فایل‌هایی که در `gitignore` معین کرده‌ایم) وارد `stage` شده‌اند.

[۵] Fractal

۳.۵ commit کردن تغییرات انجام شده

درگام 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_P1)
$ git commit -m "Implement P1"
[fb_P1 c1f21df] Implement P1
 15 files changed, 595 insertions(+)
 create mode 100644 P1/P1.sln
 create mode 100644 P1/P1/P1.csproj
 create mode 100644 P1/P1/App.config
 create mode 100644 P1/P1/Program.cs
 create mode 100644 P1/P1/Properties/AssemblyInfo.cs
 create mode 100644 P1/P1Tests/P1Tests.csproj
 create mode 100644 P1/P1Tests/Properties/AssemblyInfo.cs
 create mode 100644 P1/P1Tests/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_P1)
$ git push origin fb_P1
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_P1 -> fb_P1
```

۵.۵ ساخت Pull Request

با مراجعه به سایت Azure DevOps یک Pull Request جدید با نام `P1` بسازید به طوری که امکان `merge` کردن شاخه‌ی `fb_P1` 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.