دانشکده‌ی مهندسی کامپیوتر

برنامه‌سازی پیشرفته (سی‌شارپ)
آزمون عملی سوم

علی حیدری
استاد: سید صالح اعتمادی

۲۰ خرداد ۱۳۹۸

فهرست مطالب

۱ مقدمه و آماده‌سازی

۱.۱ نکات مورد توجه

برای برگزاری موفقیت‌آمیز و همراه با آرامش امتحان عملی لطفا نکات زیر را رعایت کنید. عدم تذکر من نشانه عدم کشف تخلف نیست.

- دقت کنید که تمامی تست‌ها در ابتدا Comment شده‌اند و شما باید برای اجرا آن‌ها را از این حالت خارج کنید.
- مشکلات یا سوالات خود را **فقط** از طریق این فرم اعلام کنید.
- ۱۵ دقیقه قبل از زمان شروع امتحان در محل امتحان حاضر باشید تا از تذکرات و روند امتحان مطلع شوید و هم‌چنین بتوانید برخی از آماده‌سازی‌های اولیه را انجام دهید.
- استفاده از کد تمرین‌های خودتان بلامانع است اما نباید از کد فرد دیگری استفاده کنید.
- ابتدای هر تست `Assert.Inconclusive();` اضافه شده است که باعث می‌شود از اجرای تست شما صرف نظر شود. برای اجرای کامل تست باید این خط را حذف کنید.
- استفاده از ویدیوهای آموزشی حین امتحان مجاز نیست.
- استفاده از تلفن همراه بعد از شروع امتحان مجاز نیست. چنان‌چه قصد استفاده از تلفن همراه برای اتصال به شبکه دارید قبل از شروع امتحان تنظیمات لازم را انجام داده، و پس از قرار دادن در وضعیت بی‌صدا درون کیف خود خارج از دید و دسترس قرار دهید.
- می‌توانید از هر گونه متن کاغذی یا دیجیتالی (کتاب، جزوه، PDF) استفاده کنید.
- پاسخ هر سوال را در مخزن[1] `Git` خودتان بارگذاری نمایید.
- توجه داشته باشید که برای کسب نمره‌ی قبولی درس کسب حداقل نصف نمره‌ی هر سری تمرین و امتحان الزامی می‌باشد.
- کپی کردن هرگونه کد از روی اینترنت یا غیراینترنت مجاز نیست. پاسخ ارسالی هر کس حتما باید توسط خود او نوشته شده باشد. کمک گرفتن از دیگران در طول مدت امتحان مجاز نیست و منجر به درج نمره‌ی **مردود** برای این درس می‌شود.
- حین امتحان تنها اجازه ارتباط با استاد درس را دارید. هر گونه ارتباط با هر فرد دیگری در جلسه امتحان یا خارج از جلسه امتحان به صورت حضوری یا مجازی مجاز نمی‌باشد.
- لطفا تمهیدات لازم برای عدم نیاز به خروج از جلسه امتحان را قبل از امتحان انجام دهید. در صورت نیاز به خروج از محل امتحان قبل از اتمام امتحان، امکان خروج بعد از هماهنگی با استاد به صورت یک نفر، یک نفر هست.
- صدا و صفحه نمایش شما باید از طریق نرم‌افزار Flashback recorder به طور کامل از ابتدا تا انتهای امتحان با تنظیمات گفته شده ضبط و ذخیره شود. لطفا قبل از شروع امتحان از درستی عملکرد این نرم‌افزار مطمئن شوید و درصورت وجود اشکال آن را رفع کنید.
- استفاده از خوردنی و نوشیدنی بدون سروصدا و بهم‌زدن نظم جلسه اشکالی ندارد.
- از هرگونه رفتار که باعث برهم‌زدن نظم و حواس‌پرتی دیگران می‌شود (مانند تایپ کردن پرسروصدا با صفحه‌کلید، بازی با قسمت فشاری خودکار، زیاد بودن صدای لپ‌تاپ هنگام خطاهای ویندوز و...) پرهیز کنید.
- استفاده از اینترنت فقط برای استفاده از `Git` و visualstudio.com لازم است. خارج از این تنها استفاده جایز از اینترنت برای پیدا کردن `syntax` سی‌شارپ یا کمک در رابطه با خطای کامپایل از سایت MSDN می‌باشد. از این جهت کلیه‌ی اتصالات مشکوک بررسی و پی‌گیری می‌شوند.
- با توجه به این‌که به هیچ سوالی پاسخ داده نمی‌شود مسئولیت رفع تمامی مشکلات پیش‌آمده با خود شماست. بنابراین توصیه می‌شود که پیش از شروع امتحان دستورات زیر را در مخزن `Git` محلی[2] خود وارد کنید تا بتوانید اشکالات احتمالی را پیش از شروع امتحان رفع کنید.

  ابتدا با دستور `git status` از وضعیت کنونی مخزن مطلع شوید. در صورت وجود فایل یا پوشه‌های `commit` نشده آن‌ها را در شاخه[3] مناسب `commit` کنید.

  مثلا در حالت زیر پوشه‌ی `A13` در وضعیت `stage` نشده قرار دارد و شما باید ابتدا آن را به `stage` اضافه و سپس آن را `commit` کنید. دقت کنید که پوشه و فایل‌های مربوط به هر تمرین در شاخه مربوطه‌اش `commit` شود.

[1] Repository
[2] local
[3] branch

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A13)
$ git status
On branch fb_A13
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    A13/

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

پس از رفع تمام اشکالات احتمالی به شاخه‌ی `master` رفته و دستور `git pull origin master` را بزنید. این دستور شاخه `master` مخزن remote را روی شاخه `master` مخزن محلی ادغام[۴] می‌کند.

سپس یک‌بار دیگر دستور `git status` را وارد کنید تا از وضعیت مخزن مطلع شوید. خروجی این دستور باید **دقیقا** به صورت زیر باشد در صورتی که این خروجی را دریافت نمی‌کنید مخزن شما هنوز آماده نیست و ممکن است در طول امتحان به مشکل بر بخورید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

۲.۱ آماده‌سازی‌های اولیه

قواعد نام‌گذاری آزمون را از جدول ۱ مطالعه کنید.

جدول ۱: قراردادهای نام‌گذاری آزمون

Naming conventions					
Branch	Directory	Solution	Project	Test Project	Pull Request
fb_E2	E2	E2	E2	E2Tests	Exam2

۱.۲.۱ آماده‌سازی‌های مربوط به git

✓ ابتدا بار دیگر از وضعیت مخزن مطلع شوید. طبق توضیحات گفته شده برای آماده‌سازی‌های مربوط به پیش از امتحان خروجی باید به صورت زیر باشد.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

✓ یک شاخه‌ی جدید با نام `fb_E2` بسازید و تغییر شاخه دهید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
$ git checkout -b fb_E2
Switched to a new branch 'fb_E2'
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_E2)
$
```

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:

```
E2
+---Project
```

[۴] merge

```
|   |   Basics.cs
|   |   DotNetInterfaces.cs
|   |   Events.cs
|   |   Inheritance.cs
|   |   Threading.cs
|   |
|   \---Linq
|           MessageAnalysis.cs
|           MessageData.cs
|
\---ProjectTests
        BasicsTests.cs
        chats.csv
        CustomStringOperatorTests.cs
        InheritanceTests.cs
        MessageAnalysisTests.cs
        ThreadsAndEvents.cs
```

در فایل پایه دو پوشه وجود دارد شما باید فایل(های) موجود در پوشه‌ی Project را به پروژه‌ی اصلی (E2) و فایل(های) موجود در پوشه‌ی ProjectTests را به پروژه‌ی تست (E2Tests) اضافه کنید.

۲ پیاده‌سازی آزمون

۱.۲ مجموعه تست‌های `Basics`

هدف این بخش سنجش آشنایی ابتدایی با پردازش رشته‌ها، پارامترهای خروجی و مدیریت خطا می‌باشد.

۱.۱.۲ تست `CalculateSum`

متد `CalculateSum` را بگونه‌ای پیاده‌سازی کنید که یک رشته شامل تعداد عدد و عملگر جمع را به عنوان پارامتر دریافت کرده و مقدار معادل عددی را برگرداند. [۲۱/۱]

پاسخ.

```
public static int CalculateSum(string expression)
  {
      var toks = expression.Split(new char [ ] { '+' },
          StringSplitOptions.RemoveEmptyEntries);
      if (toks.Length < 2)
          throw new InvalidDataException();

      int sum = 0;
      foreach (var tok in toks)
          sum += int.Parse(tok);

      return sum;
  }
```

۲.۱.۲ تست `CalculateSumInvalid`

در صورتیکه رشته ورودی یک عبارت ریاضی صحیح نباشد لازم است متد `CalculateTest` یک `Exception` از نوع `InvalidDataException` پرتاب کند. [۲۰/۲]

پاسخ.

```
public static int CalculateSum(string expression)
  {
      var toks = expression.Split(new char [ ] { '+' },
          StringSplitOptions.RemoveEmptyEntries);
      if (toks.Length < 2)
          throw new InvalidDataException();
```

```
        int sum = 0;
        foreach (var tok in toks)
            sum += int.Parse(tok);

        return sum;
    }
```

۳.۱.۲ تست CalculateSumFormatException

همچنین لازم است در صورتیکه زیررشته موجود در عبارت بین علامت‌های جمع از نوع عدد صحیح نباشد Exception متناظر آن رخ دهد. ۱۹/۳

پاسخ.

```
public static int CalculateSum(string expression)
    {
        var toks = expression.Split(new char [ ] { '+' },
            StringSplitOptions.RemoveEmptyEntries);
        if (toks.Length < 2)
            throw new InvalidDataException();

        int sum = 0;
        foreach (var tok in toks)
            sum += int.Parse(tok);

        return sum;
    }
```

۴.۱.۲ تست TryCalculateSum

متد TryCalculate را به گونه‌ای پیاده‌سازی کنید که در صورت بروز خطا false برگرداند و در هیچ شرایطی Exception رخ ندهد. در این متد مقدار عددی محاسبه شده بصورت یک پارامتر از نوع out بازگردانده می‌شود. ۱۸/۴

پاسخ.

```
public static bool TryCalculateSum(string expression, out int value)
    {
        bool bSuccess = false;
        value = 0;
        try
        {
            value = CalculateSum(expression);
            bSuccess = true;
        }
        catch
        { }
        return bSuccess;

    }
```

۵.۱.۲ تست PIPrecision

یکی از راه‌های محاسبه عدد پی استفاده از دنباله زیر است.

$$۱ - \frac{۱}{۳} + \frac{۱}{۵} - \frac{۱}{۷} + \frac{۱}{۹} - \cdots = \frac{\pi}{۴}.$$

سوال این است که برای محاسبه عدد پی تا هفت رقم معنادار این دنباله را تا عبارت چندم باید محاسبه کنیم. راهنمایی: Math.PI عدد تا دقت بیش از ده رقم اعشار. Math.Round(x, 10): عدد ورودی را به ده رقم معنادار رند می‌کند. حال متد PIPrecision را به گونه‌ای پیاده‌سازی کنید که پاسخ این سوال را پیدا کند. ۱۷/۵

پاسخ.

```
public static int PIPrecision()
  {

      double piSum = 0;
      double denominator = 1;
      double sign = 1;
      int iteration = 0;
      while (Math.Round(4 * piSum, 7) != Math.Round(Math.PI, 7))
      {
          piSum += (sign / denominator);
          sign = sign * -1;
          denominator += 2;
          iteration++;
      }

      return iteration;

  }
```

۶.۱.۲ تست Fibonacci

Extension متد به نام Fibonacci را برای نوع داده‌ای int به گونه‌ای پیاده‌سازی کنید که عدد امnدر دنباله بیبوناچی را برگرداند و این تست پاس شود. ۱۶/۶

پاسخ.

```
 public static int Fibonacci(this int n)
  {

      return n <= 1 ? 1 : Fibonacci(n - 1) + Fibonacci(n - 2);

  }
```

۷.۱.۲ تست RemoveDuplicatesDebug

کلاس FullName متد RemoveDuplicates پیاده‌سازی شده‌اند ولی این تست پاس نمی‌شود. کلاس FullName را بگونه‌ای تغییر دهید که تست زیر پاس شود. ۱۵/۷

پاسخ.

```
public static void RemoveDuplicates<T>(ref T[] list)
  {
      List<T> newList = new List<T>();
      foreach(var item in list)
      {
          if (!Contains(newList, item))
              newList.Add(item);
      }
      list = newList.ToArray();
  }
```

۲.۲ مجموعه تست‌های ThreadsAndEvents

تست‌های این بخش مربوط به IEnumerable/foreach/yield و Event و Thread هستند

۱.۲.۲ تست GetElapsedTimes

لازم است متد GetElapsedTimes به گونه‌ای پیاده‌سازی شود که فاصله دسترسی به مقداری این مجموعه را برگرداند. به این شکل که مقدار اولی که بر میگرداند همیشه صفر است و مقادیر بعدی برابر با تعداد میلی ثانیه‌ای است که از زمان دسترسی به مقدار قبلی گذشته است. پارامتر ورودی این متد حداکثر طول این رشته است. در این متد می‌توانید از کلاس Stopwatch برای اندازه‌گیری زمان استفاده کنید. برای شیوه استفاده از آن به MSDN یا تست‌های پیاده‌سازی شده از که این کلاس استفاده می‌کنند مراجعه کنید. ۱۴/۸

پاسخ.

```
public static IEnumerable<long> GetElapsedTimes(int max=100)
  {
      Stopwatch sw = Stopwatch.StartNew();
      while (true)
      {
          var elapsed = sw.ElapsedMilliseconds;
          sw.Restart();
          yield return elapsed;

          if (max-- == 0)
              break;
      }
  }
```

۲.۲.۲ تست DuplicateNumberAdded

کلاس DuplicateNumberDetector را به گونه‌ای پیاده‌سازی کنید که یک متد AddNumber داشته باشد که یک عدد صحیح به عنوان ورودی دریافت می کند. چنانچه این متد با یک عدد تکراری صدا زده شود لازم است Event به نام DuplicateNumberAdded اتفاق بیافتد که به عنوان پارامتر تعداد کل دفعاتی که عدد تکراری اضافه شده را برگرداند. ۱۳/۹

پاسخ.

```
public class DuplicateNumberDetector
  {
      private List<int> _Numbers = new List<int>();

      public IEnumerable<int> Numbers => Numbers;

      public void AddNumber(int n)
      {
          bool bDuplicate = _Numbers.Contains(n);
          _Numbers.Add(n);
          if (bDuplicate && DuplicateNumberAdded != null)
              DuplicateNumberAdded(n);
      }

      public event Action<int> DuplicateNumberAdded;
  }
```

۳.۲.۲ تست MakeItFaster

متد MakeItFasterTest را به گونه‌ای پیاده‌سازی کنید که تعدادی Action از ورودی دریافت کرده و آنها را همزمان اجرا کند. این متد تنها زمانی باید برگردد که تمام delegate ها اجرا شده باشند. ۱۲/۱۰

پاسخ.

```
public static void MakeItFaster(params Action[] actions)
  {
      List<Task> tList = new List<Task>();
      foreach (var action in actions)
      {
          Task t = Task.Run(action);
```

```
            tList.Add(t);
        }
        Task.WaitAll(tList.ToArray());
    }
```

۳.۲ مجموعه تست‌های `MessageAnalysis`

فایل `chats.csv` حاوی پیام‌های گروه تلگرامی درس AP این ترم به شما داده شده است. مشابه تمرین ۱۲ `Reference` VisualBasic را به پروژه‌ی اصلی `(E2)` اضافه کنید. کلاس‌های لازم برای پارس کردن مشابه با تمرین ۱۲ به شما داده شده است.

۱.۳.۲ تست `MostRepliedMessage`

برای پاس شدن این تست شما باید از بین کل پیام‌های داده شده پیامی را که بیشترین پاسخ[5] را دریافت کرده در قابل یک شی از نوع `MessageData` برگردانید. ۱۱/۱۱

پاسخ.

```
public MessageData MostRepliedMessage()
  {

      int? messageId = Messages
          .Where(m => m.ReplyMessageId != null)
          .GroupBy(m => m.ReplyMessageId)
          .OrderByDescending(gp => gp.Count())
          .First()
          .Key;
      return Messages.FirstOrDefault(m => m.Id == messageId);

  }
```

۲.۳.۲ تست `MostPostedMessagePersons`

برای پاس شدن این تست شما باید ۵ نفری را که بیش‌ترین تعداد پیام را در گروه ارسال کرده‌اند به جز استاد `"Sauleh Eetemadi"` و سرتیای `"Ali Heydari"` به صورت یک آرایه از زوج مرتب‌ها[6] برگردانید. عضو اول این زوج مرتب نام نویسنده‌ی پیام، و عضو دوم آن زوج مرتب تعداد کل پیام‌هایی است که آن فرد در گروه ارسال کرده می‌باشد. ۱۰/۱۲

پاسخ.

```
public class DuplicateNumberDetector
  {
      private List < int > _Numbers = new List < int > ();

      public IEnumerable < int > Numbers => Numbers;

      public void AddNumber(int n)
      {
          bool bDuplicate = _Numbers.Contains(n);
          _Numbers.Add(n);
          if (bDuplicate && DuplicateNumberAdded != null)
              DuplicateNumberAdded(n);
      }

      public event Action < int > DuplicateNumberAdded;
  }
```

[5] reply
[6] Tuple

۳.۳.۲ تست `MostActiveAtMidNight`

برای پاس شدن این تست شما باید ۵ نفر اولی که بیش‌ترین تعداد پیام را در ساعات اولیه‌ی بامداد `00:00` تا `04:00` ارسال کرده‌اند به صورت آرایه‌ای از زوج مرتب‌ها برگردانید. عضو اول این زوج مرتب نام نویسنده‌ی پیام، و عضو دوم آن زوج مرتب تعداد کل پیام‌هایی است که آن فرد در گروه ارسال کرده می‌باشد. ۹/۱۳

پاسخ.

```
public Tuple < string, int > [ ] MostActivesAtMidNight()
  {
      return Messages
          .Where(m => m.DateTime.Hour >= 0 && m.DateTime.Hour <= 4)
          .GroupBy(m => m.Author)
          .Select(m => Tuple.Create(m.Key, m.Count()))
          .OrderByDescending(gp => gp.Item2)
          .Take(5)
          .ToArray();
  }
```

۴.۳.۲ تست `MostQuestionsWithNoAnswer`

با فرض این که هر پیامی که دارای کاراکتر علامت سوال (هم کاراکتر فارسی علامت سوال `'؟'` و هم کاراکتر انگلیسی علامت سوال `'?'`) باشد یک سوال محسوب می‌شود کدام فرد بیش‌ترین تعداد سوال بدون پاسخ را پرسیده است؟ نام آن فرد را برگردانید. ۸/۱۴

پاسخ.

```
public string StudentWithMostUnansweredQuestions()
  {
      return Messages
          .Where(m => m.Content.Contains('?') || m.Content.Contains('؟')
          .GroupBy(m => m.Author)
          .Select(m => new {Count = m.Count(), Author = m.Key, ids = m.Select(x
              => x.Id)})
          .Where(g =>
              Messages
                  .Select(m => g.ids.Any(x => x == m.ReplyMessageId)).Any())
          .OrderByDescending(gp => gp.Count)
          .First().Author;
  }
```

۴.۲ مجموعه تست‌های `Inheritance`

کلاس های `Person` ، `Employee` ، `Student` ، `Teacher` باید به گونه‌ای پیاده‌سازی شوند که ارث‌بری‌ها، متدهای `virtual` ، `abstract` ، `override` به درستی و مطابق با تست‌ها تعریف شوند.

۱.۴.۲ تست `Person`

`Person` یک کلاس `abstract` است با یک سازنده که نام و مونث بودن را به عنوان پارمتر دریافت می‌کند. این کلاس یک `virtual getter` به نام `Name` دارد که نام با پیشوند خانم یا آقا را برمی‌گرداند. علاوه بر این یک `getter` به نام `IsFemale` نیز دارد. همچنین دارای یک `abstract getter` به نام `LunchRate` ۷/۱۵

پاسخ.

```
public abstract class Person
  {
      protected readonly string _Name;
      public virtual string Name => $"{PreFix} {_Name}";
      protected string PreFix => IsFemale ? FemalePrefix : MalePrefix;
      protected const string FemalePrefix = "خانم";
      protected const string MalePrefix = "آقای";
```

```
        public readonly bool IsFemale;

        public Person(string Name, bool isFemale)
        {
            _Name = Name;
            IsFemale = isFemale;
        }

        public abstract int LunchRate { get; }
    }
```

۲.۴.۲ تست Student

کلاس `Student` از `Person` به ارث برده و نرخ ناهار را برابر ۲۰۰۰ تومان قرار می‌دهد. این کلاس را بگونه‌ای پیاده‌سازی کنید که تست زیر پاس شود. ۶/۱۶

پاسخ.

```
public class Student : Person
    {
        public Student(string Name, bool isFemale) : base(Name, isFemale)
        {}

        public override int LunchRate => 2000;
    }
```

۳.۴.۲ تست Employee

کلاس `Employee` از `Person` به ارث برده و نرخ ناهار را برابر ۵۰۰۰ تومان قرار می‌دهد. علاوه بر این یک متد `virtual` به نام `CalculateSalary` اضافه می‌کند که حقوق کارمند را برابر ساعتی ۵۰۰۰ تومان حساب می‌کند. این کلاس را بگونه‌ای پیاده‌سازی کنید که تست زیر پاس شود. ۵/۱۷

پاسخ.

```
public class Employee : Person
    {
        public Employee(string Name, bool isFemale) : base(Name, isFemale)
        {}

        public virtual int CalculateSalary(int hours) => hours * 5000;
        public override int LunchRate => 5000;
    }
```

۴.۴.۲ تست Teacher

کلاس `Teacher` از `Employee` به ارث برده و نرخ ناهار را برابر ۱۰۰۰۰ تومان قرار می‌دهد. علاوه بر این متد `CalculateSalary` را `override` می‌کند که حقوق استاد را برابر ساعتی ۲۰۰۰۰ تومان حساب می‌کند. این کلاس را بگونه‌ای پیاده‌سازی کنید که تست زیر پاس شود. ۴/۱۸

پاسخ.

```
public class Teacher: Employee
    {
      public Teacher(string Name, bool isFemale) : base(Name, isFemale)
      {}

      public override string Name => $"استاد {base._Name}";
      public override int CalculateSalary(int hours) => hours * 20000;
      public override int LunchRate => 10000;

```

```
  }
```

۵.۲ مجموعه تست‌های `CustomStringOperator`

کلاس `MyString` باید بگونه‌ای پیاده‌سازی شده و اپراتورهای لازم برای آن `overload` شده یا متدهای لازم `override` شوند که تست‌های این مجموعه پاس بشوند.

۱.۵.۲ تست `StringExplicitOperator`

اپراتور تبدیل نوع داده‌ای از `System.String` به `MyString` را پیاده‌سازی کرده و متدها و اپراتورهای لازم برای مقایسه این دو نوع را به گونه‌ای پیاده سازی کنید که این تست پاس شود. ۳/۱۹

پاسخ.

```
public class MyString
  {
    public MyString(string s)
    {
        this.Text = s;
    }

    private string Text;
    public static explicit operator MyString(string s)
    {
        return new MyString(s);
    }
  }
```

۲.۵.۲ تست `StringReverseExplicitOperator`

اپراتور تبدیل نوع داده‌ای از `MyString` به `System.String` را پیاده‌سازی کرده و متدها و اپراتورهای لازم برای مقایسه این دو نوع را به گونه‌ای پیاده سازی کنید که این تست پاس شود. ۲/۲۰

پاسخ.

```
public class MyString
  {
    public MyString(string s)
    {
        this.Text = s;
    }

    private string Text;
    public static explicit operator MyString(string s)
    {
        return new MyString(s);
    }
  }
```

۳.۵.۲ تست `StringPlusPlusOperator`

اپراتور `++` را برای کلاس `MyString` به گونه‌ای پیاده‌سازی کنید که رشته حرفی را به حروف بزرگ تبدیل کند. ۱/۲۱

پاسخ.

```
public static MyString operator++ (MyString s)
  {
      s.Text = s.Text.ToUpper();
      return s;
```

```
  }
```

۴.۵.۲ تست `StringMinusMinusOperator`

اپراتور `++` را برای کلاس `MyString` به گونه‌ای پیاده‌سازی کنید که رشته حرفی را به حروف کوچک تبدیل کند. ۰/۲۲

پاسخ.

```
public static MyString operator-- (MyString s)
  {
      s.Text = s.Text.ToLower();
      return s;
  }
```

۳ ارسال آزمون

در اینجا یک‌بار دیگر ارسال آزمون 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_E2)
$ git status
On branch fb_E2
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    E2/

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

همان‌طور که مشاهده می‌کنید فولدر `E2` 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_E2)
$ git add E2/*
```

حال دوباره وضعیت فایل‌ها و فولدرها را مشاهده می‌کنیم:

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_E2)
$ git status
On branch fb_E2
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   E2/E2.sln
    new file:   E2/E2/E2.csproj
    new file:   E2/E2/App.config
    new file:   E2/E2/Program.cs
    new file:   E2/E2/Properties/AssemblyInfo.cs
    new file:   E2/E2Tests/E2Tests.csproj
    new file:   E2/E2Tests/Properties/AssemblyInfo.cs
    new file:   E2/E2Tests/packages.config
    .
    .
```

```
.
```

همانطور که مشاهده می‌کنید فولدر `E2` 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_E2)
$ git commit -m "Implement Exam2"
[fb_E2 c1f21df] Implement Exam2
 15 files changed, 595 insertions(+)
 create mode 100644 E2/E2.sln
 create mode 100644 E2/E2/E2.csproj
 create mode 100644 E2/E2/App.config
 create mode 100644 E2/E2/Program.cs
 create mode 100644 E2/E2/Properties/AssemblyInfo.cs
 create mode 100644 E2/E2Tests/E2Tests.csproj
 create mode 100644 E2/E2Tests/Properties/AssemblyInfo.cs
 create mode 100644 E2/E2Tests/packages.config
.
.
.
```

۴.۳ ارسال تغییرات انجام شده به Remote repository

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

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_E2)
$ git push origin fb_E2
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_E2 -> fb_E2
```

۵.۳ ساخت Pull Request

با مراجعه به سایت Azure DevOps یک Pull Request جدید با نام `Exam2` بسازید به طوری که امکان `merge` کردن شاخه‌ی `fb_E2` 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.