دانشکده‌ی مهندسی کامپیوتر

برنامه‌سازی پیشرفته (سی‌شارپ)
آزمون عملی دوم
(مجموعه سوالات سری اول)

علی حیدری
استاد: سید صالح اعتمادی

۹ اردیبهشت ۱۳۹۸

فهرست مطالب

۱ آماده‌سازی

۱.۱ نکات مورد توجه

- در این آزمون شما می‌توانید به یکی از دو سری سوال پاسخ دهید در صورت پاسخ به مجموعه سوالات سری اول نمره‌ی شما از ۱۰۰ و در صورتی که به مجموعه سوالات سری دوم پاسخ دهید نمره‌ی شما از ۵۰ محاسبه خواهد شد.
- نمره‌ی مجموعه سوالات قابل جمع‌بندی نیست (در صورتی که هر دو مجموعه سوالات را حل کنید فقط یکی از آن‌ها بررسی می‌شود)
- دقت کنید که تمامی تست‌ها در ابتدا Comment شده‌اند و شما باید برای اجرا آن‌ها را از این حالت خارج 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 Comment کنید.
- استفاده از ویدیوهای آموزشی حین امتحان مجاز نیست.
- هرگونه استفاده از تلفن همراه حین امتحان مجاز نیست.
- می‌توانید از هر گونه متن کاغذی یا دیجیتالی (کتاب، جزوه، PDF) استفاده کنید.
- پاسخ هر سوال را در ریپازیتوری 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.
- 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 ضبط و ذخیره شود.

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_E1`	`E1`	`E1`	`E1`	`E1Tests`	`Exam1`

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 آزمون انجام دهید را مرور می‌کنیم.

✓ ابتدا به شاخه‌ی `master` بروید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_A7)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

✓ تغییرات انجام‌شده در `Remote 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
 .gitattributes                                |    63 +
 E1/E1.sln                                     |    37 +
 E1/E1/E1.csproj                               |    61 +
 E1/E1/App.config                              |     6 +
 E1/E1/Program.cs                              |    15 +
 E1/E1/Properties/AssemblyInfo.cs              |    36 +
 .
 .
 .
```

✓ یک شاخه‌ی جدید با نام `fb_E1` بسازید و تغییر شاخه دهید.

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (master)
$ git checkout -b fb_E1
Switched to a new branch 'fb_E1'
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_E1)
$
```

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:

```
E1
+---Project
|   +---Classes
|   |   |   GameBoard.cs
|   |   |
|   |   +---Animals
|   |   |       Crow.cs
|   |   |       Frog.cs
|   |   |       Partridge.cs
|   |   |       Snake.cs
|   |   |
|   |   \---Vehicles
|   |           Airplane.cs
|   |           Submarine.cs
|   |
|   +---Enums
|   |       Environment.cs
|   |
|   \---Interfaces
|           IAnimal.cs
|           ICrawlable.cs
|           IFlyable.cs
|           ISwimable.cs
|           IWalkable.cs
|
\---ProjectTests
    \---Classes
```

```
        +---Animals
        |       CrowTests.cs
        |       FrogTests.cs
        |       PartridgeTests.cs
        |       SnakeTests.cs
        |
        \---Vehicles
                AirplaneTests.cs
                SubmarineTests.cs
```

در فایل پایه دو پوشه وجود دارد شما باید فایل(های) موجود در پوشه‌ی Project را به پروژه‌ی اصلی (E1) و فایل(های) موجود در پوشه‌ی ProjectTests را به پروژه‌ی تست (E1Tests) اضافه کنید.

۲ پیاده‌سازی

۱.۲ مقدمه و شرح سوال

فرض کنید که می‌خواهیم یک بازی رایانه‌ای درست کنیم. این بازی یک شبیه‌ساز باغ وحش است. در این بازی پس از زدن دکمه‌ای از سوی کاربر هر حیوان موجود در باغ وحش حرکت می‌کند. نکته‌ای که وجود دارد این است که هر حیوان به شیوه‌ی خودش حرکت می‌کند. بعضی حیوانات برای حرکت کردن و جابه‌جایی راه می‌روند، برخی دیگر پرواز می‌کنند و دسته‌ای دیگر شنا می‌کنند و... از آن‌جایی که رفتار هر حیوان برای حرکت با توجه به محیطی که در آن قرار دارد متفاوت است بنابراین نمی‌توان یک متد یک‌سان از همه‌ی حیوانات فراخوانی کرد مگر آن که از واسط‌ها استفاده کنیم.

۲.۲ انواع داده‌ی شمارشی (enumrations)

۱.۲.۲ Environment

این نوع داده برای شما پیاده‌سازی شده و نیازی به انجام کاری در این قسمت ندارید. پیاده‌سازی به صورت زیر است:

```
namespace E1.Enums
{
    public enum Environment
    {
        Land,
        Watery,
        Air,
    }
}
```

۳.۲ واسط‌ها

۱.۳.۲ واسط IWalkable

این واسط برای راهنمایی پیاده‌سازی شده‌است.
این واسط دارای یک ویژگی از نوع `double` با نام `SpeedRate` و یک متد با نام `Walk` است که نوع داده‌ای بازگشتی این متد `string` است.

۲.۳.۲ واسط ISwimable

گام اول: برای این واسط یک ویژگی از نوع `double` با نام `SpeedRate` پیاده‌سازی کنید.
گام دوم: برای این واسط یک متد با مقدار بازگشتی از نوع `string` با نام `Swim` پیاده‌سازی کنید.

۳.۳.۲ واسط IFlyable

گام اول: برای این واسط یک ویژگی از نوع `double` با نام `SpeedRate` پیاده‌سازی کنید.
گام دوم: برای این واسط یک متد با مقدار بازگشتی از نوع `string` با نام `Fly` پیاده‌سازی کنید.

۴.۳.۲ واسط ICrawlable

گام اول: برای این واسط یک ویژگی از نوع `double` با نام `SpeedRate` پیاده‌سازی کنید.
گام دوم: برای این واسط یک متد با مقدار بازگشتی از نوع `string` با نام `Crawl` پیاده‌سازی کنید.

۵.۳.۲ واسط IAnimal

گام اول: برای این واسط سه ویژگی از نوع‌های `string` ، `int` و `double` به ترتیب با نام‌های `Name` ، `Age` و `Health` پیاده‌سازی کنید.
گام دوم: برای این واسط یک متد با مقدار بازگشتی از نوع `string` با نام `EatFood` پیاده‌سازی کنید.
گام سوم: برای این واسط یک متد با مقدار بازگشتی از نوع `string` با نام `Reproduction` با یک پارامتر ورودی از نوع `IAnimal` پیاده‌سازی کنید.
گام چهارم: برای این واسط یک متد با مقدار بازگشتی از نوع `string` با نام `Move` با یک پارامتر ورودی از نوع `Environment` پیاده‌سازی کنید.

۴.۲ کلاس‌ها

۱.۴.۲ کلاس Airplane

تست‌ها:

✓ `FlyTest`

گام اول: برای این کلاس یک ویژگی از نوع `string` با نام `Model` پیاده‌سازی کنید.
گام دوم: سازنده‌ی این کلاس را تکمیل کنید.
گام سوم: واسط `IFlyable` را برای این کلاس به گونه‌ای پیاده‌سازی کنید که رشته‌ی بازگشتی از متد `Fly` حاوی مدل و سرعت آن با قالب زیر باشد.
ابتدا مدل هواپیما سپس عبارت `" with "` سپس سرعت هواپیما و درنهایت عبارت `" speed rate is flying"` . مثال:

برنامه‌ی نمونه:

```
using System;
using E1.Classes.Vehicles;

namespace E1
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Airplane airplane = new Airplane(1200, "C130");

            Console.WriteLine(airplane.Fly());
        }
    }
}
```

خروجی:

```
C130 with 1200 speed rate is flying
```

۲.۴.۲ کلاس Submarine

تست‌ها:

✓ `SwimTest`

گام اول: برای این کلاس دو ویژگی از نوع‌های `string` و `double` به ترتیب با نام‌های `Model` و `MaxDepthSupported` پیاده‌سازی کنید.
گام دوم: سازنده‌ی این کلاس را تکمیل کنید.

گام سوم: واسط `ISwimable` را برای این کلاس به گونه‌ای پیاده‌سازی کنید که رشته‌ی بازگشتی از متد `Swim` حاوی مدل و بیشینه‌ی عمق پشتیبانی‌شده توسط زیر دریایی با قالب زیر باشد.
ابتدا مدل زیردریایی سپس عبارت `" is a "` سپس نام کلاس سپس عبارت `" and is swimming in "` سپس بیشینه عمق پشتیبانی شده و درنهایت عبارت `" meter depth"` . مثال:

برنامه‌ی نمونه:

```
using System;
using E1.Classes.Vehicles;

namespace E1
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Submarine submarine = new Submarine("Turtle", 100, 20.1);

            Console.WriteLine(submarine.Swim());
        }
    }
}
```

خروجی:

```
Turtle is a Submarine and is swimming in 100 meter depth
```

۳.۴.۲ کلاس Snake

تست‌ها:

- ✓ `CrawlTest`
- ✓ `MoveTest`
- ✓ `EatFoodTest`
- ✓ `ReproductionTest`

گام اول: سازنده‌ی این کلاس را تکمیل کنید.
گام دوم: واسط‌های `ICrawlable` و `IAnimal` را برای این کلاس به گونه‌ای پیاده‌سازی کنید که رشته‌ی بازگشتی از متدهای زیر به فرمت گفته شده باشد:

۱. `Crawl` :
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is crawling"`

۲. `EatFood` :
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is eating"`

۳. `Reproduction` :
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and reproductive with "` سپس نام حیوانی که از پارامتر وردی متد گرفته شده.

۴. `Move` : در صورتی که این حیوان بتواند در آن محیط حرکت کند خروجی همان متد متناظر با آن محیط را باز می‌گرداند در غیر این صورت عبارتی با فرمت زیر را باز می‌گرداند:
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and can't move in "` سپس نام محیطی که از ورودی گرفته شده و نمی‌تواند در آن حرکت کند و در نهایت رشته‌ی `" environment"` .

برنامه‌ی نمونه:

```cs
using System;
using E1.Classes.Animals;
using Environment = E1.Enums.Environment;

namespace E1
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Snake snake1 = new Snake("Afie", 12, 88.1, 3.2);
            Snake snake2 = new Snake("Kobra", 13, 84.1, 0.2);

            Console.WriteLine(snake1.EatFood());
            Console.WriteLine(snake1.Crawl());
            Console.WriteLine(snake1.Reproduction(snake2));
            Console.WriteLine(snake1.Move(Environment.Air));
            Console.WriteLine(snake1.Move(Environment.Watery));
            Console.WriteLine(snake1.Move(Environment.Land));

        }
    }
}
```

خروجی:

```
Afie is a Snake and is eating
Afie is a Snake and is crawling
Afie is a Snake and reproductive with Kobra
Afie is a Snake and can't move in Air environment
Afie is a Snake and can't move in Watery environment
Afie is a Snake and is crawling
```

۴.۴.۲ کلاس Crow

تست‌ها:

- ✓ `EatFoodTest`
- ✓ `MoveTest`
- ✓ `FlyTest`
- ✓ `ReproductionTest`

گام اول: سازنده‌ی این کلاس را تکمیل کنید.

گام دوم: واسط‌های `IFlyable` و `IAnimal` را برای این کلاس به گونه‌ای پیاده‌سازی کنید که رشته‌ی بازگشتی از متدهای زیر به فرمت گفته شده باشد:

۱. `Fly`:

نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is flying"`

۲. `EatFood`:

نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is eating"`

۳. `Reproduction`:

نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and reproductive with "` سپس نام حیوانی که از پارامتر وردی متد گرفته شده.

۴. `Move`: در صورتی که این حیوان بتواند در آن محیط حرکت کند خروجی همان متد متناظر با آن محیط را باز می‌گرداند در غیر این صورت عبارتی با فرمت زیر را باز می‌گرداند:

نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and can't move in "` سپس نام محیطی که از ورودی گرفته شده و نمی‌تواند در آن حرکت کند و در نهایت رشته‌ی `" environment"` .

برنامه‌ی نمونه:

```
using System;
using E1.Classes.Animals;
using Environment = E1.Enums.Environment;

namespace E1
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Crow crow1 = new Crow("Mr crow", 6, 18.1, 91.9);
            Crow crow2 = new Crow("Mrs crow", 4, 49.5, 8.7);

            Console.WriteLine(crow1.Fly());
            Console.WriteLine(crow1.EatFood());
            Console.WriteLine(crow1.Reproduction(crow2));
            Console.WriteLine(crow1.Move(Environment.Air));
            Console.WriteLine(crow1.Move(Environment.Watery));
            Console.WriteLine(crow1.Move(Environment.Land));
        }
    }
}
```

خروجی:

```
Mr crow is a Crow and is flying
Mr crow is a Crow and is eating
Mr crow is a Crow and reproductive with Mrs crow
Mr crow is a Crow and is flying
Mr crow is a Crow and can't move in Watery environment
Mr crow is a Crow and can't move in Land environment
```

۵.۴.۲ کلاس Frog

تست‌ها:

- ✓ `EatFoodTest`
- ✓ `ReproductionTest`
- ✓ `WalkTest`
- ✓ `MoveTest`
- ✓ `SwimTest`

گام اول: سازنده‌ی این کلاس را تکمیل کنید.
گام دوم: واسط‌های `IWalkable` ، `ISwimable` و `IAnimal` را برای این کلاس به گونه‌ای پیاده‌سازی کنید که رشته‌ی بازگشتی از متدهای زیر به فرمت گفته شده باشد:

۱. `Walk` :
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is walk"`

۲. `Swim` :
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is swiming"`

۳. `EatFood` :
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is eating"`

۴. `Reproduction` :
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and reproductive with "` سپس نام حیوانی که از پارامتر وردی متد گرفته شده.

۵. `Move` : در صورتی که این حیوان بتواند در آن محیط حرکت کند خروجی همان متد متناظر با آن محیط را باز می‌گرداند در غیر این صورت عبارتی با فرمت زیر را باز می‌گرداند:
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and can't move in "` سپس نام محیطی که از ورودی گرفته شده و نمی‌تواند در آن حرکت کند و در نهایت رشته‌ی `" environment"` .

برنامه‌ی نمونه:

```
using System;
using E1.Classes.Animals;
using Environment = E1.Enums.Environment;

namespace E1
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Frog frog1 = new Frog("Mr GoorGhoori", 12, 84.1, 0.2);
            Frog frog2 = new Frog("Mrs GoorGhoori", 13, 44.1, 1.2);

            Console.WriteLine(frog1.Walk());
            Console.WriteLine(frog1.Swim());
            Console.WriteLine(frog1.EatFood());
            Console.WriteLine(frog1.Reproduction(frog2));
            Console.WriteLine(frog1.Move(Environment.Air));
            Console.WriteLine(frog1.Move(Environment.Watery));
            Console.WriteLine(frog1.Move(Environment.Land));
        }
    }
}
```

خروجی:

```
Mr GoorGhoori is a Frog and is walking
Mr GoorGhoori is a Frog and is swimming
Mr GoorGhoori is a Frog and is eating
Mr GoorGhoori is a Frog and reproductive with Mrs GoorGhoori
Mr GoorGhoori is a Frog and can't move in Air environment
Mr GoorGhoori is a Frog and is swimming
Mr GoorGhoori is a Frog and is walking
```

۶.۴.۲ کلاس Partridge

تست‌ها:

- ✓ `EatFoodTest`
- ✓ `MoveTest`
- ✓ `WalkTest`
- ✓ `FlyTest`
- ✓ `ReproductionTest`

گام اول: سازنده‌ی این کلاس را تکمیل کنید.
گام دوم: واسط‌های `IWalkable` ، `IFlyable` و `IAnimal` را برای این کلاس به گونه‌ای پیاده‌سازی کنید که رشته‌ی بازگشتی از متدهای زیر به فرمت گفته شده باشد:

۱. `Fly` :
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is flying"`

۲. `Swim` :
نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is swiming"`

۳. `EatFood` :

نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and is eating"`

۴. `Reproduction` :

نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and reproductive with "` سپس نام حیوانی که از پارامتر وردی متد گرفته شده.

۵. `Move` : در صورتی که این حیوان بتواند در آن محیط حرکت کند خروجی همان متد متناظر با آن محیط را باز می‌گرداند در غیر این صورت عبارتی با فرمت زیر را باز می‌گرداند:

نام حیوان در ابتدای رشته سپس عبارت `" is a "` سپس نوع کلاس حیوان سپس عبارت `" and can't move in "` سپس نام محیطی که از ورودی گرفته شده و نمی‌تواند در آن حرکت کند و در نهایت رشته‌ی `" environment"` .

برنامه‌ی نمونه:

```
using System;
using E1.Classes.Animals;
using Environment = E1.Enums.Environment;

namespace E1
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Partridge partridge1 = new Partridge("Mr Kabk", 6, 14.1, 11);
            Partridge partridge2 = new Partridge("Mrs Kabk", 3, 49.5, 1.7);

            Console.WriteLine(partridge1.Fly());
            Console.WriteLine(partridge1.Walk());
            Console.WriteLine(partridge1.EatFood());
            Console.WriteLine(partridge1.Reproduction(partridge2));
            Console.WriteLine(partridge1.Move(Environment.Air));
            Console.WriteLine(partridge1.Move(Environment.Watery));
            Console.WriteLine(partridge1.Move(Environment.Land));
        }
    }
}
```

خروجی:

```
Mr Kabk is a Partridge and is flying
Mr Kabk is a Partridge and is walking
Mr Kabk is a Partridge and is eating
Mr Kabk is a Partridge and reproductive with Mrs Kabk
Mr Kabk is a Partridge and is flying
Mr Kabk is a Partridge and can't move in Watery environment
Mr Kabk is a Partridge and is walking
```

۷.۴.۲ کلاس GameBoard

تست‌ها:

✓ `MoveAnimalsTest`

گام اول: یک Property از نوع `List<IAnimal>` با نام `Animals` بنویسید.
گام دوم: سازنده‌ی کلاس را تکمیل کنید.
گام سوم: متدی با نام `MoveAnimals` بنویسید به طوری که نتیجه‌ی فراخوانی متد `Move` به ترتیب در `Environment` های `Air`,`Land`,`Watery` بر روی تک‌تکِ حیوانات موجود در `Animals` برگرداند.

برنامه‌ی نمونه:

```
using System;
using System.Collections.Generic;
using E1.Classes;
using E1.Classes.Animals;
using E1.Interfaces;
using Environment = E1.Enums.Environment;

namespace E1
{
    public class Program
    {
        public static void Main(string[] args)
        {
            List<IAnimal> animals = new List<IAnimal>()
            {
                new Snake("Afie", 12, 88.1, 3.2),
                new Crow("Mr crow", 6, 18.1, 91.9),
                new Frog("Mr GoorGhoori", 12, 84.1, 0.2),
                new Partridge("Mr Kabk", 6, 14.1, 11),
            };

            GameBoard<IAnimal> gameBoard = new GameBoard<IAnimal>(animals);

            List<string> moveAnimals = gameBoard.MoveAnimals();

            foreach (string moveAnimal in moveAnimals)
            {
                Console.WriteLine(moveAnimal);
            }
        }
    }
}
```

خروجی:

```
Afie is a Snake and can't move in Air environment
Afie is a Snake and is crawling
Afie is a Snake and can't move in Watery environment
Mr crow is a Crow and is flying
Mr crow is a Crow and can't move in Land environment
Mr crow is a Crow and can't move in Watery environment
Mr GoorGhoori is a Frog and can't move in Air environment
Mr GoorGhoori is a Frog and is walking
Mr GoorGhoori is a Frog and is swimming
Mr Kabk is a Partridge and is flying
Mr Kabk is a Partridge and is walking
Mr Kabk is a Partridge and can't move in Watery environment
```

۳ ارسال

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_E1)
$ git status
On branch fb_E1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

```
    E1/

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

همان‌طور که مشاهده می‌کنید فولدر E1 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_E1)
$ git add .
```

حال دوباره وضعیت فایل‌ها و فولدرها را مشاهده می‌کنیم:

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_E1)
On branch fb_E1
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   E1/E1.sln
    new file:   E1/E1/E1.csproj
    new file:   E1/E1/App.config
    new file:   E1/E1/Program.cs
    new file:   E1/E1/Properties/AssemblyInfo.cs
    new file:   E1/E1Tests/E1Tests.csproj
    new file:   E1/E1Tests/Properties/AssemblyInfo.cs
    new file:   E1/E1Tests/packages.config
    .
    .
    .
```

همانطور که مشاهده می‌کنید فولدر E1 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_E1)
$ git commit -m "Implement Exam1"
[fb_E1 c1f21df] Implement Exam1
 15 files changed, 595 insertions(+)
 create mode 100644 E1/E1.sln
 create mode 100644 E1/E1/E1.csproj
 create mode 100644 E1/E1/App.config
 create mode 100644 E1/E1/Program.cs
 create mode 100644 E1/E1/Properties/AssemblyInfo.cs
 create mode 100644 E1/E1Tests/E1Tests.csproj
 create mode 100644 E1/E1Tests/Properties/AssemblyInfo.cs
 create mode 100644 E1/E1Tests/packages.config
.
.
.
```

۴.۳ ارسال تغییرات انجام شده به Remote repository

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

```
Ali@DESKTOP-GS7PR56 MINGW64 /c/git/AP97982 (fb_E1)
$ git push origin fb_E1
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_E1 -> fb_E1
```

۵.۳ ساخت Pull Request

با مراجعه به سایت Azure DevOps یک Pull Request جدید با نام `Exam1` بسازید به طوری که امکان `merge` کردن شاخه‌ی `fb_E1` 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.