دانشکده مهندسی کامپیوتر

برنامه‌سازی پیشرفته
آزمون عملی اول(بخش دوم)

مدرس: سید صالح اعتمادی
طرح تمرین: امید میرزاجانی

چهارشنبه ۱۰ اردیبهشت ۹۹

فهرست مطالب

۱ مقدمه

۱.۱ موارد مورد توجه

- توجه داشته باشید که برای کسب نمره‌ی قبولی درس کسب حداقل نصف نمره‌ی هر سری تمرین و آزمون الزامی می‌باشد.
- 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!
- استفاده از ویدیوهای آموزشی حین امتحان مجاز نیست.
- هرگونه استفاده از تلفن همراه حین امتحان مجاز نیست.
- صدا و صفحه نمایش شما باید از طریق نرم‌افزار Flashback recorder به طور کامل از ابتدا تا انتهای امتحان ضبط و ذخیره شود.

۲ آماده سازی های اولیه

۱.۲ ساخت پروژه ی C#

برای ایجاد پروژه C# کافی است کد زیر را در ترمینال خود اجرا کنید:

```
mkdir Exam1_cs
cd Exam1_cs
dotnet new sln
mkdir Exam1_cs
cd Exam1_cs
dotnet new console
cd ..
dotnet sln add Exam1_cs\Exam1_cs.csproj
mkdir Exam1_cs.Tests
cd Exam1_cs.Tests
dotnet new mstest
dotnet add reference ..\Exam1_cs\Exam1_cs.csproj
cd ..
dotnet sln add Exam1_cs.Tests\Exam1_cs.Tests.csproj
```

۲.۲ قواعد نام گذاری

قواعد نام‌گذاری تمرین را از جدول ۱ مطالعه کنید.

جدول ۱: قراردادهای نام‌گذاری تمرین

Naming conventions		
Branch	Directory	Pull Request
`fb_Exam1B`	`Exam1B`	`Exam1B`

* در کل یک دیرکتوری داخل Assignments به نام Exam۱ بسازید و داخل آن،دو فولدر به نام های Exam۱B و Exam۱ داشته باشید و فایل های مربوطه را داخل دیرکتوری مربوطه بگذارید.

۳ تشریح مسئله

حالا که دیمو [۱] بسیار در امر برنامه نویسی ماهر شده است، می خواهد برنامه اسنپ را برای خود شبیه سازی کند. او که علاقه زیادی به برنامه نویسی شی گرا پیدا کرده، نقشه راه، یعنی اینترفیس ها و کلاس ها را به خوبی شناسایی کرده، اما برای پیاده سازی به کمک شنا نیاز دارد.

۴ ILocatable

چیزی که در برنامه های مسیریابی خیلی مهم است، موقعیت اماکن و افراد است.
لذا برای اجسامی که موقعیت(از نظر مکانی) دارند، این واسط استفاده خواهد شد. این واسط شامل سه ویژگی

- `X` از نوع `long`
- `Y` از نوع `long`
- `Z` از نوع `long`

و متد `Distance()` است که ورودی آن یک متغیر از نوع `ILocatable` ، و خروجی آن `long` است. پس از پیاده سازی صحیح این بخش، تست های `ILocatablePropertyTest` و `ILocatableMethodTest` پاس خواهند شد.

۵ IPerson

این واسط برای اشخاص به کار می رود و شامل دو ویژگی

- `Firstname` از نوع `string`
- `Lastname` از نوع `string`

است. پس از پیاده سازی صحیح این بخش، تست های `IPersonPropertyTest` و پاس خواهند شد.

۶ Place

این کلاس برای پیاده سازی مکان های مختلف نقشه به کار می رود و علاوه بر ویژگی های اینترفیس `ILocatable` ، یک ویژگی دیگر به نام `Name` از نوع `string` دارد.
با پیاده سازی صحیح سازنده این کلاس، تست های `PlaceInheritedTest` و `PlaceConstructorTest` پاس خواهند شد.
سپس متد `Distance` را به گونه ای پیاده سازی کنید که فاصله خودش را از ورودی حساب کند. پس از پیاده سازی صحیح این بخش، تست `PlaceDistanceTest` نیز پاس خواهد شد.

۷ Traffic

این کلاس برای پیاده سازیِ سفرهای رفته شده توسط مشتریان ایجاد شده است. این کلاس شامل دو ویژگی

- `source` از نوع `Place`
- `target` از نوع `Place`

است که مکان مبدأ و مقصد سفر را مشخص می کند.
پس از پیاده سازی صحیح این کلاس به همراه سازنده اش، تست `TrafficConstructorTest` پاس خواهد شد.

---

[۱] Dimo

۸ Customer

این کلاس برای پیاده سازی مشتریان برنامه اسنپ به کار می رود که منطقاً هر فردی که از این برنامه استفاده میکند باید اولاً شخص باشد، دوماً موقعیت مکانی داشته باشد. این کلاس علاوه بر ویژگی های اینترفیس های `IPerson` و `ILocatable` ، شامل دو ویژگی

- `accountBalance` از نوع `long`
- `History` از نوع `List<Traffic>`

است که `History` اطلاعات سفرهایی که آن مشتری تا بحال رفته را در خود ذخیره میکند و `accountBalance` موجودی حساب کاربری آن کاربر است.

۱.۸ Constructor

برای پیاده سازی سازنده این کلاس، دقت کنید که ورودی اول، یک رشته شامل نام و نام خانوادگی است.
همچنین مقادیر پیشفرض `accountBalance` ، `History` به ترتیب ۰ و یک لیست خالی خواهد بود. پس از پیاده سازی صحیح این کلاس و سازنده اش، تست های `CustomerInheritedInterfacesTest` و `CustomerConstructorTest` پاس خواهند شد.

۲.۸ Operator +

برای این کلاس عملگر + را به گونه ای طراحی کنید که وقتی آن را با یک عدد جمع زدیم، موجودی حساب کاربر را به همان مقدار اضافه کند.
پس از پیاده سازی صحیح این بخش، تست `CustomerChargeAccountTest` پاس خواهد شد.

۳.۸ Distance

این متد را به گونه ای پیاده کنید که مانند قبل، فاصله خودش را تا مکان ورودی حساب کند. پس از پیاده سازی این بخش، تست `CustomerDistanceTest` پاس خواهد شد.

۴.۸ Enumerable

به این کلاس، هر آنچه نیاز است اضافه کنید تا تست `CustomerIEnumerableTest` پاس شود. هدف از این تست آن است که وقتی یک حلقه تکرار روی آن میزنیم، سفرهای رفته شده که در `History` ذخیره شده اند، به صورت رشته هایی برگرداند که با نام مبدأ شروع شوند و در انتها نیز نام مقصد باشد و نیز بین این دو با یک کاراکتر ':' از هم جدا شوند. تست ها مسأله را شفاف تر میکنند.

۹ IDriver

این واسط برای پیاده سازی راننده های شرکت طراحی شده است و شامل ویژگی های

- `Color` از نوع `Color` که رنگ آن وسیله نقلیه را مشخص میکند
- `History` از نوع `List<Traffic>`

و متد GoToTarget است که دو ورودی به ترتیب از نوع `Customer` و `ILocatable` میگیرد و خروجی ندارد. پس از پیاده سازی صحیح این بخش، تست های `IDriverPropertyTest` و `IDriverMethodTest` پاس خواهند شد.

CarDriver ۱۰

Constructor ۱.۱۰

همانطورکه از معنا مشخص است، یک راننده اتومبیل، دارای ویژگی های `ILocatable` و `IPerson` و `IDriver` است. این کلاس را به همراه سازنده اش پیاده کنید تا تست های `CarDriverInheritedInterfacesTest` و `CarDriverConstructorTest` پاس شود.

VehicleColor ۲.۱۰

متد `VehicleColor` را برای این کلاس به گونه ای پیاده سازی کنید که خروجی از نوع `string` داشته باشد که با نام راننده شروع شود،سپس پس از عبارت `has a` رنگ وسیله و سپس نوع وسیله نقلیه بیاید.
پس از پیاده سازی این بخش، تست `CarDriverVehicleTest` پاس خواهد شد.

Distance ۳.۱۰

متد `Distance` را برای این کلاس به گونه ای پیاده سازی کنید که مانند قبل، فاصله خودش را تا مکان ورودی حساب کند. پس از پیاده سازی این بخش، تست `CarDriverDistanceTest` پاس خواهد شد.

GoToTarget ۴.۱۰

این متد اصولاً برای پیاده سازیِ انجام سفر ها طراحی شده است. به کمک آن دو ورودی به ترتیب از نوع `Customer` و `ILocatable` گرفته میشود که مشخص کننده مسافر و مقصد است. به ازای هر سفری که انجام میشود، از موجودی حساب آن کاربر، مقداری کم خواهد شد. این مقدار از حاصل ضرب مسافت بین مبدأ و مقصد در ۳۰۰۰ به دست خواهد آمد. پس از پیاده سازی صحیح این بخش، تستِ `CarDriverGoToTargetTest` پاس خواهد شد.

همچنین منطقاً پس از انجام سفر، موقعیت مسافر و راننده، به موقعیت مقصد تغییر خواهد کرد. پس از پیاده سازی صحیح این بخش، تستِ `CarDriverLocationTest` پاس خواهد شد.

امّا نکته ای که باید به آن توجه کرد، این است که سفر تنها در صورتی انجام میگیرد، که کاربر، موجودی حسابی کافی برای انجام آن سفر را داشته باشد. پس به متد، این قسمت را اضافه کنید که اگر شرایط انجام سفر نبود، استثنایی از نوع `StackOverflowException` رخ دهد. پس از پیاده سازی صحیح این بخش، تستِ `CarDriverExceptionTest` پاس خواهد شد.

Operator + ۵.۱۰

برای این کلاس عملگر + را به گونه ای طراحی کنید که وقتی آن را با یک Traffic جمع زدیم، آن سفر را به لیست سفرهای انجام شده خود اضافه کند. پس از پیاده سازی صحیح این بخش، تستِ `CarDriverOperatorTest` پاس خواهد شد.

TruckDriver ۱۱

Constructor ۱.۱۱

همانطورکه از معنا مشخص است، یک راننده کامیون، دارای ویژگی های `ILocatable` و `IPerson` و `IDriver` است. این کلاس را به همراه سازنده اش پیاده کنید تا تست های `TruckDriverInheritedInterfacesTest` و `TruckDriverConstructorTest` پاس شود.

VehicleColor ۲.۱۱

متد `VehicleColor` را برای این کلاس به گونه ای پیاده سازی کنید که خروجی از نوع `string` داشته باشد که با نام راننده شروع شود،سپس پس از عبارت `has a` رنگ وسیله و سپس نوع وسیله نقلیه بیاید.
پس از پیاده سازی این بخش، تست `TruckDriverVehicleTest` پاس خواهد شد.

Distance ۳.۱۱

متد `Distance` را برای این کلاس به گونه ای پیاده سازی کنید که مانند قبل، فاصله خودش را تا مکان ورودی حساب کند. پس از پیاده سازی این بخش، تست `TruckDriverDistanceTest` پاس خواهد شد.

GoToTarget ۴.۱۱

این متد مانند کلاس `CarDriver` پیاده سازی خواهد شد تنها با این تفاوت که قیمت مسیر را باید با نرخ ۷۲۰۰ حساب کنید. پس از پیاده سازی صحیح، تست های `TruckDriverExceptionTest` `TruckDriverLocationTest` `TruckDriverGoToTargetTest` پاس خواهد شد.

MotorCycleDriver ۱۲

Constructor ۱.۱۲

همانطور که از معنا مشخص است، یک راننده موتور سیکلت، دارای ویژگی های `ILocatable` و `IPerson` و `IDriver` است. این کلاس را به همراه سازنده اش پیاده کنید تا تست های `MotorCycleInheritedInterfacesTest` و `MotorCycleConstructorTest` پاس شود.

VehicleColor ۲.۱۲

متد `VehicleColor` را برای این کلاس به گونه ای پیاده سازی کنید که خروجی از نوع `string` داشته باشد که با نام راننده شروع شود،سپس پس از عبارت `has a` رنگ وسیله و سپس نوع وسیله نقلیه بیاید.
پس از پیاده سازی این بخش، تست `MotorCycleVehicleTest` پاس خواهد شد.

Distance ۳.۱۲

متد `Distance` را برای این کلاس به گونه ای پیاده سازی کنید که مانند قبل، فاصله خودش را تا مکان ورودی حساب کند. پس از پیاده سازی این بخش، تست `MotorCycleDistanceTest` پاس خواهد شد.

GoToTarget ۴.۱۲

این متد مانند کلاس `CarDriver` پیاده سازی خواهد شد تنها با این تفاوت که قیمت مسیر را باید با نرخ ۳۵۰۰ حساب کنید. پس از پیاده سازی صحیح، تست های `MotorCycleExceptionTest` `MotorCycleLocationTest` `MotorCycleGoToTargetTest` پاس خواهد شد.

Management ۱۳

مدیر پروژه خواسته که این کلاس به صورتی پیاده سازی شود که به نوعی راننده ها از هم تفکیک شود یعنی راننده های ماشین از راننده های موتور و کامیون جدا باشند.
برای ایجاد چنین مفهمومی، برای این کلاس یک ویژگی به نام `Drivers` تعریف کنید، که اولاً از نوع آرایه باشد، دوماً قابلیت ذخیره رانندگان را داشته باشد.(جهت یادآوری: رانندگان دارای خواص `IDriver` ، `IPerson` ، `ILocatable` بودند.)

SortDrivers ۱.۱۳

متد `SortDrivers` را به گونه ای پیاده سازی کنید که اگر در `Drivers` رانندگان اتومبیل ذخیره شده بود، آنان را بر اساس میزان مسافتی که در سفرها طی کرده اند، مرتب کند. پس از پیاده سازی صحیح تست `SortCarDriversTest` پاس خواهد شد.
اگر رانندگان موتور سیکلت ذخیره شده بود، آنان را بر اساس حروف الفبایی نام خانوادگی شان مرتب کند پس از پیاده سازی صحیح تست `SortMotorCycleDriversTest` پاس خواهد شد.
رانندگان کامیون ذخیره شده بود، آنان را بر اساس حروف الفبایی نام شان مرتب کند پس از پیاده سازی صحیح تست `SortTruckDriversTest` پاس خواهد شد.

Tracing ۲.۱۳

شرکت برای ردیابی رانندگانش نیاز دارد هر لحظه، بداند که رانندگانش در چه موقعیتی هستند. برای همین از شما خواسته اند که متد `WhereIsNow` را بگونه ای پیاده سازی کنید که با فراخوانی آن، یک آرایه از نوع رشته برگرداند و درآن موقعیت همه رانندگان به صورت یک رشته که با نام آن راننده شروع میشود و موقعیت X آن راننده را اطلاع میدهد. خواندن تست، مسأله را شفاف تر میکند.

Nearest ۳.۱۳

شرکت برای ردیابی رانندگانش نیاز دارد هر لحظه، بداند که رانندگانش در چه موقعیتی هستند. برای همین از شما خواسته اند که متد `WhereIsNow` را بگونه ای پیاده سازی کنید که با فراخوانی آن، یک آرایه از نوع رشته برگرداند و درآن موقعیت همه رانندگان به صورت یک رشته که با نام آن راننده شروع میشود و موقعیت X آن راننده را اطلاع میدهد. خواندن تست، مسأله را شفاف تر میکند.
پس از پیاده سازی صحیح، تستِ `WhereIsDriversTest` پاس خواهد شد.

Nearest ۴.۱۳

برای اینکه هزینه های شرکت به حداقل برسد، نیاز است که وقتی کسی درخواست سفر داد، نزدیک ترین راننده به کاربر، او را سوار کند. پس برای کمک به ما یک متد به نام `NearestDriver` بنویسید که دو ورودی از نوع `Customer` و `Place` داشته باشد و کاربر را به مقصد برساند. پس از پیاده سازی صحیح، تست `NearestTest` پاس خواهد شد.

قهرمان و پرقدرت باشید.