Programowanie 3 - Zaawansowane

Laboratorium 4 - Operators, Indexers, BCL:

Twoja firma otrzymała zlecenie na stworzenie systemu zarządzania zamówieniami dla międzynarodowego dostawcy przesyłek. Jako doświadczony developer C#, przydzielono Ci zadanie implementacji kluczowych komponentów systemu.

Etap01: Operacje na stringach i parsowanie danych.

W całej aplikacji użyj ustawień kulturowych `CultureInfo.InvariantCulture`.

Tworzenie klasy `Customer`: W folderze `Models` utwórz klasę `Customer` zgodnie z poniższą specyfikacją:

Property	Type
`FirstName`	`string`
`LastName`	`string`
`PhoneNumber`	`string`
`EmailAddress`	`string`
`SatisfactionRatings`	`double[]`

Wymagania:

- Właściwości klasy są dostępne tylko do odczytu.
- Klasa posiada jeden konstruktor, który przyjmuje wszystkie właściwości w odpowiedniej kolejności.
- Metoda `ToString` powinna być przesłonięta tak, aby obiekty były drukowane w formacie: `(John Doe, 1234567890, john.doe@gmail.com, [ 5.0 , 4.5 , 1.3 , 4.5 ])`.

Walidacja danych klientów:

- W folderze `Validators` zaimplementuj abstrakcyjną klasę `Validator`, która posiada metodę `Validate` przyjmującą parametr typu `string?` i zwracającą wartość logiczną `bool`.
- Zaimplementuj klasy dziedziczące po `Validator`:
  - `NameValidator` - Imię i nazwisko są uważane za poprawne, jeżeli składają się tylko z liter (znaki `[a-zA-Z]`). Przed zapisem w obiekcie wartości powinny być zapisane z wielkich liter.
  - `PhoneNumberValidator` - Numer telefonu jest poprawny, jeżeli składa się z dokładnie 9 cyfr. Przed zapisem w obiekcie każde wystąpienie cyfry 6 należy zastąpić cyfrą 9.
  - `EmailAddressValidator` - Adres e-mail jest poprawny, jeżeli kończy się na `.com` oraz zawiera znak `@`.

Parsowanie danych klientów: Dane klientów są przechowywane w plikach `csv`, gdzie separatorem pól jest `;`, a separatorem rekordów znak nowej linii (`\n`). Nieporządny współpracownik zapomniał oczyścić dane z białych znaków przed ich zapisem do pliku. Twoim zadaniem jest odczytanie tak zapisanych danych, oczyszczenie ich oraz zwalidowanie zgodnie z regułami biznesowymi.

W folderze `Services` zaimplementuj klasę `CsvParser` z metodą `ParseCustomers` przyjmującą parametr `content` typu `string` oraz zwracającą tablicę obiektów typu `Customer`.

Wymagania:

- Wszystkie rekordy, które nie spełniają co najmniej jednej reguły biznesowej, należy pominąć w wartości zwracanej.
- W przypadku nieprawidłowego rekordu, na konsoli wyświetlany jest komunikat w formacie: `[09/10/2024 19:18] Invalid Customer in line 5`.
- Do sformatowania daty użyj General date/time pattern (short time).

`customers.csv` w folderze `Data` opisuje format, w jakim zapisane są dane o klientach oraz zawiera przykładowe dane.

Przydatne linki:

- Immutability of strings.
- String interpolation.
- Using StringBuilder for fast string creation.
- Extract substrings from a string.
- StringSplitOptions Enum.
- String.Join Method.
- String.Format Method.
- Parsing numeric strings in .NET.
- DateTime Struct.
- TimeSpan Struct.
- Choose between DateTime, DateOnly, DateTimeOffset, TimeSpan, TimeOnly, and TimeZoneInfo.
- Standard date and time format strings.
- Standard numeric format strings.

Etap02: Definiowanie operatorów, krotki i formatowanie.

Tworzenie klasy `Package`: W folderze `Models` utwórz klasę `Package`, reprezentującą sześcienne paczki o określonej wadze.

Field	Type
`Size`	`double`
`Weight`	`double`

Wymagania:

- Pola `Size` oraz `Weight` powinny być tylko do odczytu (użyj słowa kluczowego `readonly`).
- Dodaj właściwość `Volume` (tylko do odczytu), która zwraca objętość paczki.
- Zaimplementuj operator dodawania paczek — wynikowa paczka ma objętość i wagę będące sumą operandów.
- Zaimplementuj operator porównania dwóch paczek. Dodaj w klasie wszystkie pozostałe potrzebne do tego metody i operatory.
- Umożliw dekonstrukcję paczki do krotki `(Weight, Size)`.
- Dodaj jawne rzutowanie krotki `(double, double)` na paczkę.

Tworzenie klasy `Location`: W folderze `Models` utwórz klasę `Location`, reprezentującą lokalizację paczki:

Property	Type
`X`	`double`
`Y`	`double`
`Name`	`string`
`Culture`	`CultureInfo`

Wymagania:

- Zaimplementuj operację odejmowania lokalizacji, która jako wynik zwraca wektor, który reprezentuje przesunięcie od pierwszej lokalizacji do drugiej.
- Przesłoń metodę `ToString` tak, aby zwracała lokalizację w formacie `[Name] at ([X]; [Y])`.
- Współrzędne powinny być formatowane zgodnie z kulturą, mieć szerokość 12, 4 liczby po przecinku i być wyrównane do prawej strony zajmowanego pola.

Przydatne linki:

- Operator overloading - predefined unary, arithmetic, equality and comparison operators.
- Equality operators - test if two objects are equal or not.

- Tuple types.
- User-defined explicit and implicit conversion operators.

Etap03: Typy wyliczeniowe, `null` oraz operacje na dacie i czasie.

Typ wyliczeniowy `Priority`: W folderze `Models` utwórz typ wyliczeniowy `Priority` z poniższymi wartościami (Typ ten ma umożliwiać stosowanie operatorów bitowych):

Priority	Value
`Standard`	`0`
`Express`	`1`
`Fragile`	`2`

Rozszerzenie klasy `Package`: Dodaj poniższe właściwości do klasy `Package` (wszystkie oprócz `Priority` mogą przyjmować wartość `null`):

Property	Type
`Sender`	`Customer?`
`Recipient`	`Customer?`
`Source`	`Location?`
`Destination`	`Location?`
`ShippedAt`	`DateTime`
`DeliveredAt`	`DateTime?`
`Priority`	`Priority`

Domyślną wartością `Priority` jest `Standard`.

Obliczanie kosztu i szybkości dostawy paczki: Dodaj właściwość `Cost`, która zwraca koszt dostawy w zależności od priorytetu i odległości pomiędzy lokalizacjami - odległość pomiędzy miejscem wysyłki oraz miejscem nadania (długość wektora w układzie kartezjańskim) pomnożona jest przez mnożnik, którego wartość wynosi:

- 100 - gdy priorytet zamówienia jest kombinacją flag `Standard` oraz `Fragile`,
- 200 - gdy priorytet zamówienia jest kombinacją flag `Express` oraz `Fragile`,
- 50 - w każdym innym przypadku.

Dodaj właściwość `DeliverySpeed`, która zwraca szybkość dostawy jako iloraz odległości pomiędzy lokalizacjami oraz czasu realizacji zamówienia wyrażonego w godzinach.

W przypadku braku możliwości określenia kosztu realizacji zamówienia, właściwości powinny zwracać wartość `null`.

Przydatne linki:

- Enumeration types.
- System.FlagsAttribute class.
- Nullable value types.
- Nullable reference types.

Etap04: Indeksatory i klasa Random.

Klasa `PackageManager`: W klasie `Repository` znajdującej się w folderze `Models` odkomentuj kolekcje `_locations` oraz `_customers`. Zaimplementuj metody `DrawLocation` oraz `DrawCustomer`, które przyjmują obiekt klasy `Random` oraz zwracają losowo wybrany element odpowiednich kolekcji.

W folderze `Services` utwórz klasę `PackageManager`, której implementacja obejmuje:

- Metodę `CreatePackage`, zwracającą obiekt typu `Package`. Jako ziarno do generowania rozmiaru i wagi użyj `12345`. Każda utworzona paczka jest zapamiętywana w stanie obiektu klasy (w wybranej przez siebie kolekcji).
  - Rozmiar i waga powinny być losowane z przedziału `[10, 100)`.
  - Data nadania jest równa bieżącej dacie przesuniętej w czasie (w przeszłość) o liczbę dni wylosowanych z przedziału `[1, 10]`.
  - Data dostarczenia w 25% przypadkach jest nieznana (wartość null), natomiast w pozostałych przypadkach jest przesunięta w czasie (w przyszłość) o liczbę dni wylosowanych z przedziału `[1, 10]`.
  - Wszystkie pozostałe pola referencyjne powinny zostać pobrane z klasy `Repository`.
- Metodę `MakeReport`, która dla każdej paczki utworzonej przez obiekt klasy `PackageManager` wypisze na konsoli komunikat postaci:
  - `Warsaw (at [Long date pattern]) => Berlin (at [Long date pattern]).`
  - `Warsaw (at [Long date pattern]) => Berlin (not delivered yet).` w przypadku, gdy `DeliveredAt` przyjmuje wartość `null`.
  - Daty powinny być sformatowane przy użyciu odpowiednich obiektów klasy `CultureInfo`.
- Indeksator, który przyjmuje parametr typu `System.Range` oraz zwraca paczki z kolekcji `Packages` odpowiadające podanemu zakresowi (w ciele indeksatora należy skorzystać z pętli `for` - nie można indeksować bezpośrednio argumentem wywołania).
- Indeksator, który przyjmuje parametry `from` oraz `to` typu `DateTime` oraz zwraca wszystkie zamówienia, których czas realizacji zawiera się w przedziale `[from, to]`.

Przydatne linki:

- Random Class.
- Random.Shared Property.
- Indexers.
- Index Struct.
- Range Struct.