FI-MATRIX

ZADÁNÍ

Vytvořte třídu reprezentující dvourozměrnou matici s rozměry *m* x *n* (tj. s libovolným počtem *m* řádků a *n* sloupců) pro datový typ *decimal*. Tuto třídu implementujte jako neměnnou (*immutable*), tj. po vytvoření matice nebude možné ji nijak dále modifikovat a veškeré změny budou mít za následek vytvoření matice nové. Protože tento přístup může zatěžovat systém při komplikovaných výpočtech vytvářením nových objektů, vytvořte i druhou třídu, u které lze hodnoty prvků matice měnit. Prvky matice se indexují čísly od 0 do (m-1) pro řádky, resp. od 0 do (n-1) pro sloupce a index řádku se uvádí před indexem sloupce.

Třída **neměnitelné** matice bude splňovat následující:

- třídu je možné vytvořit předáním dvourozměrného pole (*decimal[,]*) nenulových rozměrů
- prvky matice nejde po vytvoření měnit, jedinou možností je vytvoření nové matice
- třída matice signalizuje, zda je čtvercová, nulová nebo jednotková (implementujte pomocí vlastností)
- lze zjistit rozměry matice
- třída implementuje ***indexer jen pro čtení*** o dvou parametrech pro přístup k prvkům matice
- třída umí přičíst k sobě jinou matici, odečíst matici a vynásobit se skalárem
  - *třída se nemůže měnit, tedy jako výsledek vrací novou matici*
- nechť platí matice jsou si rovny ⇔ mají stejný rozměr ⋀ odpovídající prvky obou matic jsou si rovny
  - přepište odpovídající metody třídy, aby tato rovnost platila

Vytvořte druhou třídu, tentokrát **měnitelnou**, pro konstrukci matice. Tato třída bude provádět změny uvnitř sebe a výslednou matici pak vytvoří na požádání. Nechť platí:

- třídu nechť lze vytvořit zadáním rozměrů matice, dvourozměrného pole nebo zkopírováním existující matice
- lze zjistit rozměry matice, která se vytváří
- třída bude mít indexer ***pro čtení i nastavení prvku*** v zadaném řádku a sloupci matice
- umí nastavit hodnotu všech prvků matice na zadanou hodnotu najednou
- umí k vytvářené matici přičíst matici, odečíst matici a vynásobit matici skalárem
- třída bude mít metodu pro vrácení hotové matice jako instanci neměnné třídy matice uvedené výše

Vytvořte pomocnou třídu obsahující rozšiřující metodu *(extension method)* pro výpis matice na konzoli. Formát výpisu si můžete zvolit, nicméně by ho měl být schopen přečíst opravující.

Ošetřete aplikaci výjimkami (`ArgumentException`, `ArgumentOutOfRangeException`, `ArgumentNullException`, `InvalidOperationException`). Nedovolte uživateli dostat třídu do chybného stavu nebo na ní provést neplatnou operaci.

Nápověda

deklarace a vytvoření 2D pole

```
decimal[,] array2D = new decimal[2,3];
```

zjištění rozměru pole

```
int d1 = array2D.GetLength(0);  // == 2; (1. rozměr)
int d2 = array2D.GetLength(1);  // == 3 (2. rozměr)
int all = array2D.Length; //== 6 (počet prvků celkem)
```

spojení prvků pole do jednoho řetězce

```
int[] value = new int[] { 1, 2, 3 };
string.Join(";", value); // == "1;2;3"
```

Testovací případy

```
decimal[,] testData = new[,] {
  { 1m, 2m },
  { 3m, 4m }
}; // vytvoření 2D pole 2x2, m za číslem značí decimal
ImmutableMatrix matrix = new ImmutableMatrix(testData);
Console.WriteLine(matrix[0, 0]); // => vrací 1
testData[0, 0] = 999;
Console.WriteLine(matrix[0, 0]); // => mělo se vypsat 1, 999 je špatně (pole je referenční datový typ)
```

Podmínky kódování úkolů

- Bez dovolení nesmíte změnit zadání.
- Dodržujte principy a koncepty OOP, důsledně uplatňujte zapouzdření.
- Aplikace nesmí skončit neočekávanou chybou bez vnějšího vlivu, chybné vstupy musí být ošetřeny co nejdříve.
- Vyhýbejte se opakovanému psaní duplicitního kódu.
- V C# funguje porovnání řetězců pomocí operátoru == stejně jako Equals.
- Vhodně používejte konstrukce jazyka C#, např. vlastnosti a indexery.
- Jeden soubor ať obsahuje jednu třídu a název souboru odpovídá názvu třídy.
- Dodržujte jmenné konvence .NET:
  - názvy parametrů metod, soukromých datových položek a proměnný pomocí camelCase (malé počáteční písmeno), vše ostatní PascalCase (velké počáteční písmeno)
  - názvy rozhraní začínají velkým písmenem `I`, implementacím se nepřidává přípona `Impl`.
- Udržujte ve třídách pořádek, zvolte si jeden způsob řazení členů ve třídě a ten dodržujte,
  - v .NET je řazení v souboru obvykle: konstanty, dat. položky, vlastnosti, konstruktory, metody.
- Snažte se psát kód přehledně, volte vhodné názvy identifikátorů.
- Členěte kód na vhodné celky (třídy/metody).