E-napló projektfeladat

Elektronikus osztálynaplót (E-napló) kell megvalósítani. Az alkalmazásban a `ClassRecords` osztály magát a naplót reprezentálja, ahol felvehetők a diákok (`Student`). A `Student` osztályban "tárold" a diákok jegyeit `Mark`. A főbb funkciók a következők: a diákok osztályozhatók, (jegyeket kapnak), a jegyeik alapján általános és tantárgyak szerinti átlag számítható, két tizedesjegy pontossággal. A jegyeik ki is listázhatók a tesztesetekben megadott formában.

Az osztályok

`Subject` osztály:

Egy property-je van, a tantárgy nevének tárolására. Ennek alapján azonosítható a tantárgy.

`Mark` osztály:

A diák számára adott jegyeket reprezentálja, az osztályzat "értékét" a property-jei között egy `int` tartalmazza. További tulajdonságai a következők: `Subject` a tantárgy, amiből kapta a jegyet, `Date` a dátum (readonly), amikor szerezte a jegyet. Ez utóbbit a konstruktor állítja be a rendszeridő alapján.

A `ToString` metódusban a következő szöveget adja vissza osztályzattól függően: `excellent(5)`, `very good(4)`, `satisfactory(3)`, `borderline(2)`, `failed(1).`

`StudyResultEntry` osztály:

Speciális osztály, funkciója a diák és tantárgyi átlagának listázásához adatszerkezetet biztosítani. Property-jei a diák neve és az össztantárgyi átlaga, konstruktorból feltöltve.

`Student` osztály:

A diák adatait - jelen esetben csak a nevét (konstruktorból feltöltve) - és a jegyeit tárolja, metódusai ezeken dolgoznak. A diák azonosítása a nevén keresztül történik. A `ToString()` metódus a teszteseteknél látható módon a diák nevét és a jegyeit listázza ki szöveges formában.

Metódusai:

- `public void AddGrade(int grade, Subject subject)`

  Hozzáadja a marks listához ezt az osztályzatot.

- `public double CalculateAverage()`

  Kiszámolja a diák összes osztályzatának átlagát.

- `public double CalculateSubjectAverage(Subject subject)`

  Kiszámolja a paraméterként kapott tantárgy jegyeinek átlagát.

`ClassRecords` osztály:

A régi papíralapú napló egyes funkcióit reprezentálja. Property-je az osztály neve, valamint a diákok listája. Diákot adhatunk hozzá és el is távolíthatunk, előbbi esetben már létező nevű diákot nem

adhatunk hozzá, és eltávolítani csak olyat lehet, aki már ott van a listában (ismét név alapján). Osztályátlagot tud számolni általánosan és tantárgy alapján, meg tud keresni egy diákot név alapján, és ki tudja listázni a diákok neveit és átlagát a `StudyResultEntry` osztály objektumainak listájaként.

Publikus metódusok:

```
public bool AddStudent(Student student)

public bool RemoveStudent(Student student)

public double CalculateClassAverage()

public double CalculateClassAverageBySubject(Subject subject)

public Student FindStudentByName(string name)

public List<StudyResultEntry> ListStudentResults()

public string ListStudentNames()
```

Érdemes lehet egy privát metódusban eldönteni, hogy létezik-e az adott nevű diák. Ha nem, akkor null-t adjon vissza:

```
public Student GetStudentByNameOrNull (string name)
```

Hibakezelés

Törekedj az átfogó hibakezelésre! A teszteseteknél látható módon különböző `Exception`-t várunk, ha a megfelelő metódust `null` értékkel hívták meg, vagy ha a string paraméter üres. Az osztály szintű átlagszámítások során `ArithmeticException`-t várunk, ha valamiért nem lehet a számítást elvégezni (nincs jegy, nincs diák felvéve az osztályba, stb.).