Programozás 3. ZH - AAAAAA

<For English version, please scroll DOWN>
Készítsen el egy konzolos alkalmazást, melynek segítségével kézilabda játékosok Covid tesztelését tudjuk szimulálni. A teszteredményeket és a játékosokat egy lokális adatbázisban tároljuk, a játékosokat pedig véletlenszerűen generáljuk és XML-ben továbbítjuk az adatbázis felé.

SOLUTION NEVE: NEPTUN KÓD, NAGYBETŰKKEL (PL. TIW8GX)

```
<players>
  <player code="AH58HU">
    <familyName>Szucsánszki</familyName>
    <firstName>Nadine</firstName>
    <position>Centre</position>
  </player>
```

DLL1: NEPTUNKÓD.UTILS

a) TeamGenerator osztály: Legyen benne egy GetTeam metódus, amely egy darabszámot vár paraméterül, és eredményként egy XDocument példányt ad vissza, amely véletlenszerű adatokból (nevek, posztok, valamint személyiségi jogok miatt a teszt során használt helyettesítő kódok) álló XML –t állít elő. Ehhez a lenti tömböket kell használni (ctrl+C, ctrl+V –vel kimásolható, nem szükséges begépelni):

```
string[] familyNames = { "Szucsánszki", "Schatzl", "Márton", "Kovacsics", "Háfra", "Klujber" };
string[] firstNames = { "Zita", "Nadine", "Gréta", "Anikó", "Noémi", "Katrin" };
string[] codes = { "AH58HU", "BL23IK", "AX95OP", "OR42CH", "QL23ZK", "YW44SC" };
string[] positions = { "RightWing", "LeftWing", "Pivot", "Centre", "Left Back", "Right Back" };
```

b) CovidTester osztály: ebben a covid teszteket végző metódusok lesznek
   1. Minden tesztelő metódus bool eredményű, string paraméterű, és a CovidTestMethod attribútummal megjelölt. A metódusok paraméterként majd a játékosok string változatát fogják megkapni.
   2. Működésük véletlenszerű, a tesztek végeredménye 1:X valószínűséggel true, az X értéke metódusonként más. Ehhez használjon véletlenszám-generátort.
   3. Ciklus nélküli megvalósítást várunk el, string = IEnumerable<char>.
   4. PcrTester: X értéke a paraméterként kapott stringben az E és T betűk összesített darabszámának háromszorosa.
   5. AntibodyTester: X értéke a paraméterként kapott stringben az A és S betűk összesített darabszámának ötszöröse.

c) TestExecutor osztály
   1. `private static MethodInfo PickRandomMethod(object covidTester)`
      Segédmetódus, a paraméterül kapott ismeretlen objektumból véletlenszerűen visszaadja az egyik olyan metódust, ami CovidTestMethod attribútummal megjelölt (ciklus nélküli megvalósítást várunk el).
   2. `private static int GetKeyValue(object obj)`
      Segédmetódus, a paraméterül kapott ismeretlen objektumból visszaadja az egyetlen olyan tulajdonság értékét, ami Key attribútummal megjelölt (amennyiben ilyen nincs, nem int típusú, vagy egynél több ilyen tulajdonság van, akkor következzen be kivétel) (ciklus nélküli megvalósítást várunk el).
   3. `public static Dictionary<int, bool> ExecuteTests (object covidTester, IEnumerable<object> people)`
      Egy tesztfuttatást szimuláló metódus. Az első paraméter majd egy CovidTester lesz, a másodikban pedig játékos példányok lesznek majd. De most még ezen típusokra itt nem hivatkozhatunk!
      A gyűjteményen végighaladva minden egyes elemre a segédmetódusok segítségével kiválaszt egy véletlenszerű tesztelő metódust az első paraméterből, az aktuális elemből pedig kiolvassa az azonosító értékét. A véletlenszerű tesztelő metódust a `MethodInfo` példány `Invoke` metódusával tudjuk meghívni (az Invoke-on keresztül meghívott véletlen metódus paramétere az aktuális gyűjtemény-elem ToString() metódusának eredménye legyen:
      `randomMethod.Invoke(covidTester, new object[] { person.ToString() }))`.
      Az ExecuteTests eredménye egy olyan Dictionary, ami mindegyik azonosítóhoz (int) a tesztelő metódus eredményét (bool) rendeli hozzá.

DLL2: NEPTUNKÓD.DB

a) Player entity osztály
    1. int Id – elsődleges kulcs, adatbázis által szolgáltatott növekményes érték
    2. FamilyName, FirstName, Code, Position – max 100 hosszú, megadása kötelező
    3. A játékoshoz tartozó tesztek gyűjteménye (navigation property)
    4. A kötelező dolgokon felül legyen benne egy családnév+keresztnév+poszt adatokat visszaadó ToString() és egy olyan konstruktor, ami a fenti XML egyetlen XElement-jéből képes Player példányt létrehozni
b) CovidTest entity osztály
    1. int Id – elsődleges kulcs, adatbázis által szolgáltatott növekményes érték
    2. DateTime Date
    3. bool IsPositive
    4. int PlayerId + Player navigation property
c) PlayerContext osztály: A szintén ebben a Class Library projektben definiált adatbázis file-ba code first módszerrel generálja le a táblákat és állítsa be a navigation property-ket; adattal nem kell feltölteni a táblákat.

CONSOLE APP: NEPTUNKÓD.PROGRAM

a) `static void FillDatabase(PlayerContext ctx)`
    1. Hozzon létre egy CovidTester és egy TeamGenerator példányt.
    2. A TeamGenerator segítségével generáljon le egy 10 elemű csapatot; a visszakapott XML-ben lévő játékosokat pedig mentse el az adatbázisba.
    3. Szimuláljon le 10 tesztelési kört: ciklussal tízszer egymás után hívja meg a TestExecutor.ExecuteTests metódust; az első paraméter a CovidTester példány legyen, a második pedig az adatbázisnak a játékosokat tároló táblája.
    4. A cikluson belül az ExecuteTests által visszaadott Dictionary példányt (több CovidTest példánnyá átalakítva) egymás után mentse el a CovidTest táblába. A dátumokat úgy állítsa be, hogy minden tesztelési kör között 5 nap telik el, az első tesztelési kör a mai nap történik.
b) `private static void QueryDatabase(PlayerContext ctx).`
   Linq segítségével jelenítse meg az alábbiakat:
    1. A játékosok és a tesztek darabszáma
    2. Statisztika a tesztekről (hány pozitív, hány nem pozitív – egy query!)
    3. Statisztika a pozitív tesztekről poszt szerint, lazy load segítségével (poszt, pozitív tesztek darabszáma)
    4. Lista az elvégzett tesztekről, join segítségével (testId, testDate, isPositive, playerCode)

A teljes solution-t (minden könyvtárral/file-al együtt) NEPTUNKÓD.ZIP nevű ZIP file-ként kell feltölteni, a Moodle/Teams/FF valamelyikére (ilyen sorrendben haladjunk, és a következőre kapcsolódási hiba esetén ugorjunk).

Jó munkát kívánunk!

Programming 3. ZH - AAAAAA

Create a console app that can be used to simulate Covid testing for handball players. The test results and the players will be stored in a local database, the players will be randomly generated and forwarded towards the database in an XML format.

```
<players>
  <player code="AH58HU">
    <familyName>Szucsánszki</familyName>
    <firstName>Nadine</firstName>
    <position>Centre</position>
  </player>
```

SOLUTION NAME: NEPTUN CODE, WITH CAPITALS (E.G. TIW8GX)

DLL1: NEPTUNCODE.UTILS

a) TeamGenerator class: Should have a GetTeam method, that wants a number (of players) as a parameter, and it returns an XDocument containing random data (names, positions, and substituting codes that are used during the tests for privacy reasons). For the random generation, you must use the arrays below (just copypaste using ctrl+C, ctrl+V , no need to type manually):

```
string[] familyNames = { "Szucsánszki", "Schatzl", "Márton", "Kovacsics", "Háfra", "Klujber" };
string[] firstNames = { "Zita", "Nadine", "Gréta", "Anikó", "Noémi", "Katrin" };
string[] positions = { "RightWing", "LeftWing", "Pivot", "Centre", "Left Back", "Right Back" };
string[] codes = { "AH58HU", "BL23IK", "AX95OP", "OR42CH", "QL23ZK", "YW44SC" };
```

b) CovidTester class: this will contain the methods that simulate the covid tests
   1. All tester methods returns bool, works with a string parameter, and must be marked with the CovidTestMethod attribute. The parameter will be the string representation of a player.
   2. Their operation is random, there is a 1:X probability that they return true, the value of X is different in the different methods. Use a random generator for this.
   3. In these methods we expect code without loops, string = IEnumerable<char>.
   4. PcrTester: The value of X is three times the number of E and T characters in the parameter string.
   5. AntibodyTester: The value of X is five times the number of A and S characters in the parameter string.

c) TestExecutor class
   1. `private static MethodInfo PickRandomMethod(object covidTester)`
      Helper method that chooses randomly one of the methods that are marked with the CovidTestMethod attribute (we expect code without loops).
   2. `private static int GetKeyValue(object obj)`
      Helper method that returns the value of the only one property that is marked with the Key attribute (if there is no such property, or it is not int, or there is more than one, an exception must happen) (we expect code without loops).
   3. `public static Dictionary<int, bool> ExecuteTests (object covidTester, IEnumerable<object> people)`
      This method simulates a test run. The first parameter will be a CovidTester instance, the second one will be a collection of player instances. Right now, we cannot use these types here!
      Going through the collection, for every item use the helper methods to select a random tester method from the first parameter, and read the ID value from the current item. You can call the tester method using the `Invoke` method of the `MethodInfo` instance (the parameter of the random method called via the „Invoke” should be the ToString() of the current person: `randomMethod.Invoke(covidTester, new object[] { person.ToString() })`).
      The result of the ExecuteTests is a Dictionary, where the test result (bool) is associated to the ID values (int).

DLL2: NEPTUNCODE.DB

a) Player entity class
    1. int Id – primary key, using database provided increment values
    2. FamilyName, FirstName, Code, Position – maximum 100 characters, obligatory
    3. The collection of tests for the current player (navigation property)
    4. In addition to the obligatory stuff, it should have a ToString() that retuns the string familyname+firstname+position, and a constructor that can create a Player instance using a single XElement from the XML above
b) CovidTest entity class
    1. int Id – primary key, using database provided increment values
    2. DateTime Date
    3. bool IsPositive
    4. int PlayerId + Player navigation property
c) PlayerContext class: Using code first method, generate the tables into the MDF file that is also in this class library project; configure the navigation properties; do not fill the tables with data.

CONSOLE APP: NEPTUNCODE.PROGRAM

a) `static void FillDatabase(PlayerContext ctx)`
    1. Create a CovidTester and a TeamGenerator instance.
    2. Use the TeamGenerator to generate a team of 10 players; save the players that are returned in the XML into the database.
    3. Simulate 10 test runs: in a loop, call the TestExecutor.ExecuteTests method ten times; the first parameter should be the CovidTester instance, the second parameter should be the table in the database that stores the players.
    4. Inside the loop, use the Dictionary returned by the ExecuteTests method (transformed into multiple CovidTest instances) to save the test results into the CovidTest table. Set the dates so that the first date is today, and every test runs are executed five days apart.
b) `private static void QueryDatabase(PlayerContext ctx).`
    Use Linq to display the followings:
    1. The number of the players and tests
    2. Statistics about the tests (number of positives, number of non-positives – one query!)
    3. Statistics about the positive tests according to positions, using lazy load
    (position, number of positive tests)
    4. List about the performed tests, using join (testId, testDate, isPositive, playerCode)

You have to upload your full solution as it is (with all files/folders included) as a NEPTUNCODE.ZIP archive file using Moodle/Teams/FF (use this order, move to the next one in case of connection errors).

We wish you a successful test!