A feladat megoldására és benyújtására 120 perc áll rendelkezésre. A létrehozott projekt megnevezése tartalmazza az Ön nevét, Neptun kódját és a dolgozat azonosítójt (pl. AliceBob_ABC123_M). A megoldását tartalmazó mappát (a teljes solution-t) tömörítve, a http://zh.nik.lan címen elérhető felületen keresztül nyújtsa be.
Ügyeljen a fordítási hibától mentes kódra, ellenkező esetben a megoldás sajnos nem értékelhető. Nem lehetséges LINQ kifejezések használata, lekérdezések írásakor az órán tanult algoritmusokkal kell megvalósítani a teljes értékű megoldáshoz. Minden változó típusát definiálni kell, nem megengedett var és dynamic kulcsszó használata.

A Kelet-Greslin-i határátkelő népszerűsége jelentősen nőtt, nem kis részt köszönhetően az Ön által előzőleg elkészített szoftvernek. Ebből fakadóan ismét Önt kérték fel a rendszer továbbfejlesztése érdekében, melyhez implementálja a következő feladatokban leírtakat! A teljes értékű megoldáshoz elvárás a kurzuson tárgyalt módszerek és technikák alkalmazása.

1 • Hozza létre az `PersonalInfoException` osztályt, ami a kivétel osztály leszármazottja. Az osztálynak egyetlen egyparaméteres konstruktora van, ami egy paraméterül kapott értékkel (`message`, karakterlánc) meghívja az ősosztály konstruktorát.

2 Hozza létre a `PersonType` nevű osztályt, ami egy személyt reprezentál. Az osztály valósítsa meg az `IComparable` interfészt.

- Tárolja el egy adott személy útlevélszámát (`passportNumber`, egészérték), nevét (`name`, karakterlánc), életkorát (`age`, egészérték), és országát (`country`, `CountryType` típusú).
  - Az országtípust az osztályon kívül hozza létre (publikusan), a kategória értékei lehetnek: `Antegria, Arstotzka, Impor, Kolechia, Obristan`.
  - Az egyes adattagokhoz készítsen publikus gettereket és szettereket. Az útlevélszám értéke 100000 és 999999 között lehet csak, az életkor legalább 14, a név pedig nem lehet üres sztring, ezeket a szetteren belül ellenőrizze. Rossz értékek esetén dobjon el egy `PersonalInfoException` kivételt a megfelelő szöveggel.
- Legyen az osztálynak egy négy paraméteres konstruktora, amely a szettereken keresztül inicializálja az adattagokat.
- Legyen az osztálynak egy három paraméteres konstruktora, amely az útlevélszámot, életkort és országot kapja paraméterként, a név ekkor mindig `[REDACTED]` értékű. A megvalósításnál a lehető legkevesebb kódismétlést alkalmazzon.
- A `ToString()` metódus adja vissza a személy adatait az alábbi formátumnak megfelelően:

```
Jorji Costava, 123456, 55, Obristan
```

- Legyen az osztálynak egy statikus `PersonType Parse(string input)` metódusa, amely olyan formátumú sztringeket tud `PersonType` objektummá parszolni, mint ami a `ToString()` metódusnál definiálva lett. Ha nem ilyen formátumú bemenet érkezik, akkor a metódus dobjon egy `PersonalInfoException` kivételt a megfelelő szöveggel.
- Írja felül a `bool Equals(object obj)` metódust. Akkor legyen egyenlő két `PersonType` objektum, ha az útlevélszámuk megegyezik.
- Az osztály valósítsa meg az `IComparable` interfészt. Ennek érdekében implementálja a szükséges metódust. Ha a metódus bemeneti objektuma karakterlánc típusú, akkor azt hasonlítsa a saját

A feladat megoldására és benyújtására 120 perc áll rendelkezésre. A létrehozott projekt megnevezése tartalmazza az Ön nevét, Neptun kódját és a dolgozat azonosítójt (pl. AliceBob_ABC123_M). A megoldását tartalmazó mappát (a teljes solution-t) tömörítve, a http://zh.nik.lan címen elérhető felületen keresztül nyújtsa be.
Ügyeljen a fordítási hibától mentes kódra, ellenkező esetben a megoldás sajnos nem értékelhető. Nem lehetséges LINQ kifejezések használata, lekérdezések írásakor az órán tanult algoritmusokkal kell megvalósítani a teljes értékű megoldáshoz. Minden változó típusát definiálni kell, nem megengedett var és dynamic kulcsszó használata.

---

nevéhez, különben ha `PersonType` típusú, akkor elsődlegesen a névhez, másodlagosan (azaz ha a nevek egyeznek) az útlevélszámhoz. Ha más típusú a paraméter, akkor dobjon el egy kivételt a megfelelő szöveggel.

3 A `PersonType` osztály néhány metódusához készítsen unit teszteket[1].

- Több tesztesettel is tesztelje a Parse metódust olyan sztringek esetén, melyek a fenti szabályok szerint parszolhatók.
- Több tesztesettel is tesztelje a Parse metódust olyan sztringek esetén, melyek a fenti szabályok szerint nem parszolhatók.
- Minden lehetséges kimenetre tesztelje a `PersonType` objektumokat összehasonlító metódust.

4 Egy határátlépési kísérlet dokumentálására készítsen egy `CrossingAttempt` nevű osztályt, amely megvalósítja az `IComparable` interfészt.

- Az osztály publikusan olvasható, de csak privát módon módosítható auto-property-k használatával tárolja el egy átlépni kívánó személyét (`Person`, `PersonType` típusú) és eredményességét (`Person`, `bool`). Ezeket egy kétparaméteres konstruktorból állítsa be.
- Az osztálynak legyen egy statikus `CrossingAttempt Parse(string input)` metódusa, amely képes a formátumú sztringeket beparszolni:

```
Jorji Costava, 123456, 55, Obristan - false
```

- Ez az osztály is valósítsa meg az `IComparable` interfészt. Ügyeljen arra, hogy a bemeneti paraméter csak karakterlánc vagy `CrossingAttempt` típusú lehet, ezeknek megfelelően végezze el a hasonlítást (a lehető legkevesebb kódismétléssel: használja ki, hogy a `PersonType` osztály is implementálja az `IComparable` interfészt). Ha más típusú a paraméter, akkor dobjon el egy kivételt a megfelelő szöveggel.

5 Készítsen egy `DailyCrossingAttempts` nevű osztályt, ami egy nap határátlépési kísérleteit dokumentálja.

- Tárolja el a napi határátlépési kísérletek (`attempts`, `CrossingAttempt` típusú) tömbjét egy privát adattagként.
- Legyen az osztálynak egy rendezettséget vizsgáló privát `bool IsSorted()` metódusa.
- Az osztály privát `void Sort()` metódusa egy, órán tanult rendező metódust (buborék, beillesztéses vagy minimum-kiválasztásos rendezés) implementálva idő szerint növekvő módon rendezi a határtátkelési kísérletek eredményeit.
- Az egyparaméteres konstruktorban állítsa be a paraméterül kapott `CrossingAttempt` típusú tömbre az `attempts` tömböt. Amennyiben nem rendezett a tömb, rendezze.

[1] Szükséges NuGet-ek:
- NUnit
- NUnit3TestAdapter
- Microsoft.NET.Test.Sdk

A feladat megoldására és benyújtására 120 perc áll rendelkezésre. A létrehozott projekt megnevezése tartalmazza az Ön nevét, Neptun kódját és a dolgozat azonosítójt (pl. AliceBob_ABC123_M). A megoldását tartalmazó mappát (a teljes solution-t) tömörítve, a http://zh.nik.lan címen elérhető felületen keresztül nyújtsa be.
Ügyeljen a fordítási hibától mentes kódra, ellenkező esetben a megoldás sajnos nem értékelhető. Nem lehetséges LINQ kifejezések használata, lekérdezések írásakor az órán tanult algoritmusokkal kell megvalósítani a teljes értékű megoldáshoz. Minden változó típusát definiálni kell, nem megengedett var és dynamic kulcsszó használata.

- Legyen egy másik, szintén egyparaméteres konstruktora is, amely egy szting tömböt kap paraméterül, melyekből `parse`-olja az `attempts` tömböt. Amennyiben nem rendezett a tömb, rendezze.
- Legyen egy publikus `bool PartOf(DailyCrossingAttempts other)` metódusa, ami azt mondja meg, hogy az adott osztálypéldány részhalmaza-e az `other` példány által tárolt halmaz.
- Legyen egy publikus `int FindPerson(string s)` metódusa, ami a paraméterül kapott névhez tartozó útlevélszámot adja vissza (bináris kereséssel), ha létezik a halmazban, és -1-et, ha nem.
- Legyen egy publikus `int Count(Predicate<CrossingAttempt> predicate)` metódusa, ami megszámolja, hogy a halmazban hány olyan elem van, ami megfelel a paraméterül kapott predikátumnak.
- Legyen egy publikus `CrossingAttempt[] Select(Predicate<CrossingAttempt> predicate)` metódusa, ami kiválogatja azon elemeket a halmazból, amik megfelelnek a paraméterül kapott predikátumnak.
- Legyen egy publikus `double AverageAge(Predicate<CrossingAttempt> predicate)` metódusa, ami kiszámolja az átlagéletkorukat azon személyeknek a halmazban, amik megfelelnek a paraméterül kapott predikátumnak.
- Legyen egy publikus `CrossingAttempt OldestSuchPerson(Predicate<CrossingAttempt> predicate)` metódusa, ami visszaadja azt a határátlépési kísérletet, amelynél a predikátumnak megfelelő személy a legöregebb.

6 Készítsen egy `BorderCrossings` nevű osztályt, ami több nap határátlépési kísérleteit dokumentálja.

- Tárolja el a napi határátlépési kísérletek (`daysOfActivity`, `DailyCrossingAttempts` típusú) tömbjét egy privát adattagként. Ezt a konstruktorból állítson be.
- Legyen egy publikus `int HowManyPassed()` metódusa, ami visszaadja azt, hogy hány embernek sikerült a határátlépés a tárolt napokon.
- Legyen egy publikus `int MostPassedDay(int age)` metódusa, ami visszaadja azt, hogy melyik indexű napon sikerült a legtöbb, legalább age éves személynek a határátlépés.
- Legyen egy publikus `double FailedAverageAge(CountryType ct)` metódusa, ami visszaadja azt, hogy mennyi azoknak az átlag életkora, akiknek nem sikerült a határátlépés.
- Legyen egy privát `CrossingAttempt[] Intersection(CrossingAttempt[] one, CrossingAttempt[] other)` metódusa, ami a kapott két hamaz metszetét állítja elő.
- Legyen egy publikus `CrossingAttempt[] TheDefinitionOfInsanity()` metódusa, ami azon sikertelen határátkelési kísérletek halmazát adja vissza, amik mindegyik napon megjelentek.

A feladat megoldására és benyújtására 120 perc áll rendelkezésre. A létrehozott projekt megnevezése tartalmazza az Ön nevét, Neptun kódját és a dolgozat azonosítójt (pl. AliceBob_ABC123_M). A megoldását tartalmazó mappát (a teljes solution-t) tömörítve, a http://zh.nik.lan címen elérhető felületen keresztül nyújtsa be.
Ügyeljen a fordítási hibától mentes kódra, ellenkező esetben a megoldás sajnos nem értékelhető. Nem lehetséges LINQ kifejezések használata, lekérdezések írásakor az órán tanult algoritmusokkal kell megvalósítani a teljes értékű megoldáshoz. Minden változó típusát definiálni kell, nem megengedett var és dynamic kulcsszó használata.

---

7 A `Program` osztályban valósítsa meg a következő metódusokat:

- A `DailyCrossingAttempts GenerateDataForOneDay(int numberOfAttempts, ref Random rnd)` metódus egy napi határtákelők adatait generálja le. Az első paraméter az átkelők számát adja meg, a második egy kívül elkészített `Random` objektumot, melyet a véletlenszám generáláshoz használjon fel:
  - Egy személy útlevélszáma egy hatszámjegyű szám,
  - Egy személy kora 14 és 91 közötti,
  - Országa a definiált 5 közül kerülhet ki
  - Átkelési sikeressége 50%
  - Neve ez alkalommal nem nyílvános.
- A `BorderCrossings GenerateData(int days, ref Random rnd)` metódus több (days) napnyi határtákelő halmaz adatait generálja le, a kapott `Random` objektum segítségével.
- A void Main() metódusban a fenti két metódus segítségével generáljon le egy hétnyi adathalmazt, majd tesztelje rá az osztályokban megírt metódusokat! (Pl.: Hány 40 éven felüli jutott át a határon, hányan jutottak át összesen, hányan voltak Kolechiából sikertelenek, stb.)