A feladat megoldására és benyújtására 120 perc áll rendelkezésre. A létrehozott projekt megnevezése tartalmazza az Ön nevét (ékezetek nélkül), Neptun kódját és a dolgozat azonosítóját (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 nem értékelhető.

Egy hobbi futó gyakran elindul félmaratoni futó versenyeken, ahol több barátja is részt vesz. Minden verseny után egy txt fájlban rögzíti, hogy a baráti köréből ki milyen eredményeket ért el. Készítsen egy olyan konzolos alkalmazást, amely segít bizonyos lekérdezéseket elvégezni a meglévő txt fájlok alapján.

1 Hozzon létre egy `TimeException` kivétel osztályt. Az osztálynak csak egy konstruktora legyen, amellyel beállítható az ősosztályban definiált `message` adattag értéke.

2 Hozza létre a `Time` osztályt, mely egészként tárol óra, perc és másodperc értékeket.

- Az egyes adattagokhoz készítsen publikus gettereket és szettereket. Ha szetteren keresztül 0-nál kisebb, vagy óra esetén háromnál nagyobb, perc és másodperc esetén pedig 59-nél nagyobb értékeket szeretnének megadni, akkor dobjon el egy `TimeException` kivételt.
- Legyen az osztálynak egy három paraméteres konstruktora, amely a szettereken keresztül inicializálja az adattagokat.
- Legyen az osztálynak egy két paraméteres konstruktora, amely perc és másodperc értéket kap csak bemenetként. Ez a konstruktor az óra értékét állítsa 0-ra, és a megvalósításnál a lehető legkevesebb kódismétlést alkalmazzon.
- Írja felül a string ToString() metódust, amely `01:56:17` formában jeleníti meg az időeredményt, ha legalább egy órán át tartott a futás. Egy órán belüli idő esetén `59:57` formátumú legyen a megjelenítés.
- Legyen az osztálynak egy statikus Time Parse(string input) metódusa, amely olyan formátumú sztringeket tud `Time` objektummá parszolni, mint ami a ToString() metódusnál definiálva lett. Ha nem ilyen formátumú bemenet érkezik, akkor a metódus dobjon egy `TimeException` kivételt.
- Írja felül a bool Equals(object? obj) metódust. Akkor legyen egyenlő két `Time` objektum, ha minden adattaguk megegyezik.
- Az osztály valósítsa meg az `IComparable` interfészt. Ennek érdekében implementálja a szükséges metódust, amely segítségével összehasonlíthatóvá válik két `Time` objektum.

3 A `Time` osztály néhány metódusához készítsen unit teszteket.

- 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 `Time` objektumokat összehasonlító metódust.

4 Egy futó aktuális eredményének eltárolására hozza létre a `RunnerWithTime` osztályt.

- Az osztály publikusan olvasható, de csak privát módon módosítható auto-property-k használatával tárolja el egy futó nevét és időeredményét.

- Az osztálynak legyen egy statikus RunnerWithTime Parse(string input) metódusa, amely képes `Jani,01:56:17` és `Zsuzsi,57:38` formátumú sztringeket beparszolni.
- Ez az osztály is valósítsa meg az `IComparable` interfészt. Az összehasonlítás alapja legyen az elért időeredmény, ha ez azonos, akkor a nevük szerinti alfabetikus sorrendet vigye figyelembe.
- Írja felül a string ToString() metódust úgy, hogy `Jani (01:56:17)` formátumban jelenítse meg a futó aktuális eredményét.
- Írja felül a bool Equals(object? obj) metódust. Két példány akkor azonos, ha minden eltárolt adata azonos.

5 Egy futóverseny adatainak tárolása érdekében hozza létre a `RaceResults` osztályt.

- Az osztálynak egyetlen adattagja legyen, melyben a futók eredményei vannak benne.
- Az osztálynak RaceResults(int runnersCount, string[] inputs) szignatúrájú konstruktora legyen. A konstruktor a megfelelő parszer segítségével feltölti az osztály adattagját. Ha a beérkező adatok idő szerint nem rendezettek, akkor rendezze az adatokat idő szerint növekvő módon.
- Legyen az osztálynak egy rendezettséget vizsgáló privát bool IsSorted() metódusa.
- Az osztály privát void Sort() metódusa a javított beillesztéses rendezést implementálva idő szerint növekvő módon rendezi a futók eredményeit.
- Legyen az osztálynak privát int LowerBound(Time time) és int UpperBound(Time time) metódusa. Az első metódus visszaadja annak az elemnek az indexét, amely az első olyan elem, melynek idő értéke nem kisebb a paraméterként beérkező időnél. A második metódus az első olyan elemnek az indexét adja vissza, amely idő értéke nagyobb a paraméterként beérkező időnél.
- A publikus RunnerWithTime[] Between(Time lower, Time upper) metódus visszaadja az összes olyan eredményt, amely a `lower`-nél nem kisebb és az `upper`-nál nagyobb idővel rendelkezik.
- A publikus bool Contains(Predicate<RunnerWithTime> predicate, out RunnerWithTime runnerPerformance) metódus megadja, hogy egy adott feltételnek megfelelő futó található az eltárolt adatok között. Ha nincs eltárolva akkor a kimeneti paraméter értéke `null`, egyébként pedig a futóról tárolt adat.

6 A `RaceResults` osztály Between metódusához készítsen olyan unit tesztet, amely azt teszteli, hogy az elvárt kimenetet szolgálatja-e a metódus.

7 Hozza létre a `Races` osztályt.

- Az osztály publikusan olvasható és privát módon módosítható auto-property segítségével tárolja el `RaceResults` típusú elemek tömbjét.
- Az osztály egy Races(RaceResults[] raceResults) szignatúrájú konstruktorral példányosítható.
- Legyen az osztálynak egy Time BestPerformance(string name) szignatúrájú publikus metódusa, amely megadja, hogy egy futónak mi a legjobb időeredménye az eddigi versenyei alapján. Ha az adott futó egy versenyen se vett részt, akkor `null` legyen a visszatérési érték.
- Legyen az osztálynak egy privát RunnerWithTime[] Union(RunnerWithTime[] first, RunnerWithTime[] second) metódusa, amely az unióját határozza meg a két bemenetnek. Ha egy elem mindkét bemeneti tömbben benne van, akkor mindkét előfordulás kerüljön be a

kimenetbe. A bemenetekről feltételezhető, a kimenettel szemben pedig elvárás, hogy időeredmény szerint növekvően rendezve legyenek bennük az elemek.

- Legyen az osztálynak egy publikus RunnerWithTime[] AllBetween(Time lower, Time upper) metódusa, amely idő szerint növekvően visszaadja az összes olyan eredményt, amely bármely versenyen a megadott két időérték közé esik.

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

- A string[] ReadFile(string path) metódus egy fájlból beolvassa a benne eltárolt adatokat. A fájl első sorában egy egész érték található, amely megadja, hogy utána hány érdemi sorban vannak adatok. Feltételezhetjük, hogy az első sort követően minden sorban olyan formátumban vannak adatok, ami alapján megvalósítható a `RunnerWithTime` típusra történő parszolás.
- A Races ReadFolder(string path) metódus egy könyvtár összes `txt` kiterjesztésű fájlja alapján előállít egy `Races` típusú példányt.
- A void Main() metódus egy könyvtárban lévő fájlok alapján létrehoz egy `Races` példányt, majd megmondja, hogy egy futónak mi a legjobb eddig időeredménye, illetve kiírja, hogy kik és milyen idővel teljesítettek eddig félmaratont `01:45:00` és `02:00:00` idő között.