Programozás 2

2. Zárthelyi dolgozat

2021. november 23.

1. (1 pont) Készítsünk egy programot, ami meghatározza, hogy egy hallgatónak hány közös órája van a többi hallgatóval. Legyen egy Hallgato osztály. Egy hallgatónak van neptun kódja, ami egyedi, és egy felvett tárgyak adatszerkezete. Minden tantárgy egyedi, nincsenek duplikált elemek. Az egyszerűség kedvéért az tantárgyak előre meghatározottak, és egy számmal vannak reprezentálva.

1 – Programozás 1	2 – Programozás 2	3 – Kalkulus 1 4 – Kalkulus 2
5 – Diszkrét matematika	6 – Elektronika 1	7 – Elektronika 2
8 – Digitális technológia	9 – Adatszerkezetek	10 – Közgazdaságtan
11 – Valószínűségszámítás és statisztika		12 – Mesterséges intelligenca

   Egy hallgató tárgyait az addSubjects() metódussal adjuk meg, aminek a bemeneti paramétere egy lista, ami az adott tantárgyhoz tartozó számokat tartalmazza.

   Legyen egy Hallgatok osztályunk is. Ebben az osztályban egy adatszerkezetet válósítunk meg, ami dinamikusan bővíthető és hallgató objektumok tárolására alkalmas.

   A Hallgatok osztály rendelkezzen két metódussal:
   - az add() metódussal hozzá lehessen adni hallgatókat az adatszerkezethez
   - a findClassmates() metódussal, ami megkapja egy hallgató neptun kódját, és meghatározza, hogy a többi hallgatónak hány közös órája van a megadott hallgatóval.

   Példa a működésre:

```
Hallgato hallgato1 = new Hallgato(“RNK0I2”);
hallgato1.addSubjects(List.of(1, 3, 5, 7, 9, 12));
Hallgato hallgato2 = new Hallgato(“K9R3MN”);
hallgato2.addSubjects(List.of(2, 8, 11, 10, 5, 6));
Hallgato hallgato3 = new Hallgato(“IJKRST”);
hallgato3.addSubjects(List.of(3, 5, 4, 7, 9, 12));
Hallgato hallgato4 = new Hallgato(“EUQOW0”);
hallgato4.addSubjects(List.of(3, 5, 4, 7, 9, 10));
```

```
Hallgatok hallgatok = new Hallgatok();
hallgatok.add(hallgato1);
hallgatok.add(hallgato2);
hallgatok.add(hallgato3);
hallgatok.add(hallgato4);

Hallgatok.findClassmates("K9R3MN");
```

Kimenet:

```
RNK0I2: 1
IJKRST: 1
EUQOW0: 2
```

Ügyeljünk arra, hogy a megadott hallgatót ne hasonlítsuk össze saját magával!

2. (1 pont) Egy koronavírusos beteg esetében nagyon fontos, hogy naponta többször meg legyen mérve a véroxigén szintje, mivel ezzel időben ki lehet mutatni a tüdő kapacitásának romlását, így a beteg időben megkaphatja a megfelelő ellátást. A véroxigén szintet egy 0-100-ig terjedő, csak egész számokat tartalmazó százalékos skálán mérik. Az átlag, egészséges érték 100-95 százalék között van, ha viszont többszöri mérésre is 95 százalék alatti az érték, akkor a beteg sürgősen oxigén kezelésre szorul.

   A feladathoz tartozik egy betegek.txt fájl. Ebben a fájlban soronként egy beteg adatai vannak felsorolva, úgy, hogy először meg van adva a beteg keresztneve, utána egy ";" karaktert követően pedig a beteghez tartozó mért értékek, "," karakterekkel elválasztva. Például:

```
Péter;97,96,97,99,100
József;100,94,92,95,99,97,91
```

   Olvassuk be a fájl adatait és tároljuk el őket.

   Készítsünk el egy checkSaturation() metódust, ami egyesével megkapja a fájl egy adott sorát, és megvizsgálja a betegekhez tartozó véroxigén szintjeiket, és ha három olyan érték is van egy betegnél, ami 95 alatti, akkor írja ki a program, hogy a beteg sürgős ellátásra szorul. Amennyiben ez az eset nem áll fenn, írja ki a program, hogy a beteg értékei megfelelőlek.

   Példa:

```
Péter állapota jó!
József sürgős ellátására szorul!
```

3. (1 pont) Egy weboldalt üzemeltető cég felmérést készített a weboldaluk felhasználóival, amiben megkérdezték, hogy melyik évben születtek. Ez alapján szeretnék csoportosítani a felhasználóikat, hogy a megfelelő generációnak szóló reklámokat helyezhessék el az oldalukon.

   Készítsünk egy programot, ami megvalósít egy szótár adatszerkezetet, amiben felhasználó nevekhez rendelünk hozzá egy generációt. A feladat egyszerűségéért feltételezzük, hogy a megadott évszamok az 1920 -2012 közötti tartományba esnek.

   A feladathoz tartozik egy felmeres.txt fájl. A fájlban soronként egy felhasználó adatai vannak úgy, hogy először van a felhasználó neve, majd egy szóközzel elválasztva a születési éve. Például:

   Szofi83 2000
   JkAdam 1982
   NagyneEva00 1946
   BenjiFiju99 2008

   Olvassuk be a fájl tartalmát és az adatokat tároljuk el egy szótár adatszerkezetben a megfelelő módon.

   A fájl beolvasása után járjuk be a feltöltött adatszerkezetet, és írjuk ki egy generaciok.txt fájlba a felmérés eredményeként kapott generációkat és a generációkba tartozó felhasználók felhasználó nevét felsorolva, a példában megadott módon. A generációk az alábbiként vannak meghatározva:

   Veterán: 1920-1939
   Baby boomer: 1940-1959
   X: 1960-1983
   Y: 1984-1994
   Z: 1995-2009
   Alfa: 2010-2012

   A generaciok.txt fájl tartalma a program lefutása után:

   Baby boomer: NagyneEva00
   X: JkAdam
   Z: Szofi83,BenjiFiju99

   A kimenetnél a sorrend tetszőleges, nem kell a példa bemenethez tartozó kimenetnek megegyezni a példabelivel!

4. (1 pont) A feladat4 nevű mappába másolja át az első feladatban elkészített Hallgato és Hallgatok osztályt. Írja meg a TestMain.java nevű forrást, amiben alaposan leteszteli a Hallgato osztályhoz tartozó addSubjects(), és a Hallgatok osztályhoz tartozó add() és findClassmates() metódusokat. Lehet több tesztet is írni egy metódushoz, de legalább az alapvető működését tesztelje le!

   Néhány tipp:

   a. addSubjects():
      - Mi történik, ha nem egyedi azonosítót adunk meg?
      - Valóban felvevődnek az értékek?

   b. add():
      - Amikor felveszünk egy elemet, ténylegesen bekerül a listába?

   c. findClassmates():
      - Mi történik, ha nem létező neptun kódot adunk meg?
      - Mi történik, ha a hallgató saját magával is összehasonlításra kerül?
      - Megfelelő értékeket kapunk vissza a metódus megfelelő hívása után?