1. heti feladatok

1. órai feladat Egy állatkertet szeretnénk reprezentálni, ahol háromféle állat van, akiket különböző méretű ketrecekben tárolunk el. Egy ketrecben maximum 10 állat lehet, de létezhet olyan ketrec is, ami ennél kevesebb állat befogadására képes. A lehetséges állatfajták: kutya, panda és nyúl.

- Készítsen egy osztályt egy állat reprezentálására **Animal** néven.
  - Tárolja el az alábbi adatokat:
    - *name* – szöveg
    - *gender* – logikai érték (*igaz* ha hím, egyébként *hamis*)
    - *weight* – egész szám
    - *species* – felsorolás típus (Dog, Panda, Rabbit)
  - A mezők értékét a konstruktorban lehessen beállítani és tulajdonságokkal lehessen lekérdezni.
- Készítsen egy osztályt, mely állatokat tároló ketrecet reprezentál **Cage** néven.
  - Legyen egy belső tömbje, ami az állatokat tárolja. Ennek a méretét a konstruktorban lehessen megadni.
  - Legyen egy mezője, ami a ketrecben lévő állatok aktuális száma.
  - Legyenek ilyen publikus metódusai:
    - *bool Add(Animal animal)*, ami paraméterként kap egy állatot, és ezt elhelyezi a tömbbe (ha elfér). Igaz értékkel tér vissza, amennyiben elfért az adott állat a ketrecben.
    - *void Delete(string name)*, ami törli a paraméterként megadott nevű állatot.
  - Lehetnek további metódusai is az osztálynak, amik a később megfogalmazott funkcionalitásokat támogatják.
- A főprogramban készítsen el egy négy **Cage** objektumot tartalmazó tömböt, és az azt töltse fel minta adatokkal.
- Írjon metódusokat a megfelelő osztályokba, amikkel meg tudja határozni az alábbi kérdésekre a válaszokat:
  - Megadott ketrecben hány darab megadott fajú állat található?
  - Megadott ketrecben van-e megadott fajú és nemű állat?
  - Megadott ketrecben melyek a megadott fajú állatok?
  - Megadott ketrecben mennyi a megadott fajú állatok átlagos tömege?
  - Megadott ketrecben van-e legalább egy azonos fajú, de ellenkező nemű egyedekből álló páros?
  - Melyik ketrecben található a legtöbb megadott fajú állat?
- Legyen lehetőség arra, hogy egy ketrecben lévő állatok adatait ki tudja írni a konzolra.
- Lehessen egy ketrecet előállítani egy szöveges fájl alapján, ahol a fájl egyes soraiban állatok adatai vannak megadott sorrendben egymástól vesszőkkel elválasztva.
- Lehessen az állatkertben lévő összes ketrecet feltölteni fájlokból úgy, hogy az alkalmazás paramétereként átadjuk egy könyvtár elérési útját és az adott könyvtárban található szöveges fájlok alapján állít elő a programunk ketreceket.

2. gyakorló feladat Maratoni futóverseny (42 km, a 195 métertől eltekintünk) szimulálására készítsen alkalmazást! Az implementáció során az alábbi osztályok használata javasolt.

- **Runner** osztály, mely egy futót reprezentál.
  - Minden futónak tudjuk az alábbi adatait:
    - *name* – szöveg
    - *pace*[1] – egész szám, 3 és 9 közötti érték lehet
    - *time* – hány perce fut a versenyen
    - *gaveUp* – logikai
  - Készítsen két paraméteres konstruktort a futó nevének és tempójának megadásával.
  - Minden adattaghoz készítsen gettert és settert, ha szükséges.
  - *int Move()* metódus szimulálja a versenyen egy perc elteltét. Ha az aktuális percben a futó egy kilométernek pont a végére ért, akkor 1-et ad vissza a metódust, egyébként pedig nullát. Haladás közben a futó lehetséges, hogy feladja a versenyt az eddig futási idejének függvényében az alábbi szabály szerint:
    - Több mint 60, de legfeljebb 90 percnyi futást követően egy ezred valószínűséggel.
    - Több mint 90, de legfeljebb 120 percnyi futást követően kettő ezred valószínűséggel.
    - Több mint 120, de legfeljebb 180 percnyi futást követően három ezred valószínűséggel.
    - Több mint 180 percnyi futást követően öt ezred valószínűséggel.
  - *string Display()* metódus megadja a futó nevét, és ha a futó feladta a versenyt, akkor ezt is hozzáfűzi a nevéhez.
- **Team** osztály, mely futók egy csapatát reprezentálja. A versenyen lehet indulni egyéniben, valamint 2, 3 és 5 fős csapatokban.
  - Minden csapatról tudjuk, hogy mely futók alkotják, illetve azt is, hogy éppen ki fut a csapattagok közül, mekkora távot és mennyi idő alatt tett meg a csapat.
  - A nem egyéniben indulók esetén a váltások az alábbi szabály szerint történnek:
    - 2 fős csapatnál 21 km megtétele után.
    - 3 fős csapatnál az első futó 10 km-t, a második 20 km-t, a harmadik pedig 12 km-t fut le.
    - 5 fős csapatnál 8, 18, 25 és 36 km-nél történnek a váltások.
  - Minden csapatról lehessen publikus getteren keresztül lekérni, hogy mennyi ideje futnak, mekkora össztávot tettek meg, és versenyben vannak-e még egyáltalán, azaz nem adta fel egyik futójuk sem a versenyt.
  - *void Move()* metódus aktualizálja a csapat által megtett távot.
  - *bool End()* megadja, hogy az adott csapat versenyben van-e még, azaz nem adták fel, de még nem is értek be a célba.
  - *string MemberNames()* megadja a csapat futóinak névsorát.
  - *string Display()* egy sorban arra a pozícióra tesz egy X-et, amennyi kilométert a csapat már épp lefutott, valamint kiírja az aktuális futó nevét.
- **Race** osztály, mely eltárolja a versenyen induló csapatokat.
  - *void Move()* a verseny egy percének elteltét szimulálja.
  - *bool End()* eldönti, hogy vége van-e már a versenynek, azaz nincs még versenyben lévő csapat.
  - *string Display()* a konzolon megjeleníti, hogy melyik csapat hol tart a versenyben, és láthatjuk azt is, hogy melyik csapatban éppen ki fut.

[1] tempó

– *Team[] Results()* a verseny végén visszaadja az elért eredményeket, mégpedig rendezett módon.