# Absztrakt és heterogén kollekció feladatok

## feladat:

Készítsen egy **Jármű** osztályt, ami tárolja a jármű gyártmányát, típusát, sebességét és meghajtását. (private) A járműnek van egy paraméteres konstruktora mely minden adattag értékét megkapja.

Készítse el az olvasható tulajdonságokat minden adattagra (public) és a ToString() metódust is!

A jármű string visszatérési értékű halad metódusa legyen absztrakt!

Származtasson egy **Személygépkocsi** osztályt a járműből! Az osztály plusz adattagja a fogyasztás, az ajtók száma, illetve, hogy a kocsi hány személyes. (private) Készítse el a szükséges konstruktort és az olvasható tulajdonságokat (public), definiálja felül a ToString() és a halad metódust – (A kocsi x km/h sebességgel gurul).

Származtasson egy **Hajó** osztályt a járműből! Az osztály plusz adattagja a vízkiszorítás és a merülés legyen! (private) Készítse el a szükséges konstruktort és az olvasható tulajdonságokat (public), definiálja felül a ToString() és a halad metódust – (A hajó x csomó sebességgel úszik).

A főprogramban példányosítson egy minimum 20 járműből álló flottát, ami vegyesen tartalmazzon hajókat és személygépkocsikat, a szükséges adatokat generálhatja is! Az autók esetén a meghajtás lehet elektromos, gáz, benzin vagy gázolaj, hajók esetén ez kézi vagy gépi értéket tároljon. Ügyeljen arra, hogy a kézi meghajtású hajók sebessége reális legyen! (pl.: kajak/kenu esetén: 3-6 km/h, több személyes evező csónak esetén: 4-10 km/h)

1. Írja ki a járművek összes adatát!
2. Hány környezetbarát meghajtású jármű van a flottában? Kocsi esetén az elektromos, hajó esetén a kézi számít környezetbarát megoldásnak!
3. Milyen gyártmányú járműből van a legtöbb?
4. Írja ki a gépi meghajtású hajók minden adatát!
5. Hányan mehetünk a hétvégén a Balatonra, ha a flotta összes kocsiját kibéreljük és nincs másik járművünk?
6. Melyik kocsival érhetünk le a leghamarabb a Balatonra?

## feladat:

Készítsen egy **Kutya** osztályt, ami tárolja a kutya nevét, korát, hogy éhes-e, illetve, hogy van-e gazdája. (private) A kutyához készítsen egy paraméteres konstruktort, ami minden adattag értékét kívülről kapja, egy másik paraméteres konstruktort, ami csak a nevét és a korát kapja paraméterül, a másik két adattag értékét véletlen generálja!

Készítse el az olvasható tulajdonságokat minden adattagra (protected) és a ToString() metódust is!

A kutya dolgozik metódusa (protected) egy string-et adjon vissza: ha a kutyának van gazdája, akkor azt, hogy „Őrzi a házat”, egyébként hogy „Kóborol”, a kutya ToString() metódusa a dolgozik metódust is hívja meg!

Származtasson egy **Vadászkutya** osztályt a kutyából! Az osztály plusz adattagja a zsákmány, ez egy egész szám és a vadászatok során elejtett prédák számát tárolja. (private) Készítse el az osztály konstruktorait, az egyikben a zsákmányt is kapja paraméterül, mint ahogy a többi adattagot, a másikban, ahol csak a nevet és a kort kapjuk parméterül, ott a zsákmány értékét állítsa 0-ra! Készítsen az adattagnak olvasható tulajdonságot! (public) Definiálja felül a kutya dolgozik metódusát úgy, hogy a gazdával bíró kopók esetén a zsákmány értékét növelje, illetve a visszatérési értéke a házőrzés helyett a „Vadászik” szöveg legyen, ha nincs gazdája akkor maradjon az ős osztályban megírt Dolgozik metódus eredménye!

Származtasson egy **Terelőkutya** osztályt a kutyából! Az osztálynak nincs plusz adattagja. Készítse el az osztály konstruktorait, használja az ős osztályban már megírtakat! Készítsen Definiálja felül a kutya dolgozik metódusát úgy, hogy a gazdával bíró pásztorkutya esetén a visszatérési érték a házőrzés helyett a „Tereli a nyájat” szöveg legyen, ha nincs gazdája akkor maradjon az ős osztályban megírt Dolgozik metódus eredménye!

A főprogramban példányosítson egy falka kutyát, a falka minimum 25 kutyát jelent! A kutya fajtáját véletlen sorsolja, kora random 1 és 15 közötti érték, neve egy kutyaneveket tartalmazó lista véletlen eleme!

1. Írd ki az összes kutyát!
2. Határozd meg, hogy vadász vagy terelőkutyából van-e több! (Vagy ugyan annyi van-e)
3. Melyik a legfiatalabb terelőkutya?
4. Mennyi a kutyák átlagéletkora? (Két tizedesjegyre kerekítve)
5. Van-e az átlagnál fiatalabb vadászkutya?
6. Írd ki a terelőkutyák számát és a „névsorukat”!

## feladat:

Készítsen egy **Állat** osztályt, ami tárolja az állat korát és hogy barátságos-e. (private) Az állathoz készítsen egy alapértelmezett konstruktort, ami a két adattag értékét véletlen generálja! Kora 1 és 20 közötti érték legyen!

Készítse el az olvasható tulajdonságokat minden adattagra (public) és a ToString() metódust is!

Az állat string visszatérési értékű HangotKiad() metódusa legyen absztrakt!

Az állat Mozgas() metódusa egy string-et adjon vissza: „Az állat mozog”, az állat ToString() metódusa a mozgás és a hangot kiad metódust is hívja meg!

Származtasson egy **Kutya** és egy **Macska** osztályt az állatból! Definiálja felül a HangotKiad() metódust – (pl.:"a kutya ugat: vau vau!")!

Származtasson egy **Madár** osztályt az állatból! Definiálja felül a következő metódusokat:

* HangotKiad() metódust – „A madár csiripel: csirip csirip!”,
* Mozgás() metódus – „A madár repül.”

A főprogramban példányosítson minimum 16 állatot, ami vegyesen tartalmazza az állatokat, a szükséges adatokat generálhatja is!

1. Írja ki az összes állatot! Hívja meg a HangotKiad() metódusukat is!
2. Hány barátságos állat található a listában?
3. Keresse meg az első Madár típusú állatot a listából!
4. Keresse meg a legidősebb állatot a listában és adja vissza az állat referenciáját! (Maga a keresett állat lesz a visszatérési érték.)
5. Mennyi az átlag életkor a kutyáknál és mennyi összesen minden állatnál? (Három tizedesjegyre kerekítve)
6. Van barátságtalan macska?
7. Válogassa szét az állatokat két külön listába, az egyik lista a barátságos, a másik a barátságtalan állatokat tartalmazza! Írja ki mindkét lista tartalmát!

## feladat:

Készítsen egy programot, amiben 3 osztály található. Legyen egy Ingatlan, egy Telek, és egy Kertesház nevű osztály. Ezeket az alábbiak szerint készítse el:

Készítsen **Ingatlan** osztályt!

* Adattagok: típus (szöveges, lehetséges értékei pl.: "panellakás", "sorház", "ikerház", stb.), cím (szöveges), telekméret (egész szám), irányár (egész szám). (private)
* Minden adattaghoz készítsen property-t. (public) Az Irányár settere csak akkor engedje az irányár tagváltozót beállítani, ha a megadott érték nagyobb, mint 100.000 Ft. Amennyiben rossz értéket adtak meg, ne változtasson a konstruktorban beállított értéken. A telekméret settere a beállításkor vizsgálja, hogy a telekméretnek legalább 10 m2 területűnek kell lenni. Ha kisebb területet adtak meg, ne módosítsa az eredeti értéket!
* Legyen egy konstruktora, ami paraméterében kapja az összes értéket.
* Az osztálynak legyen egy másik konstruktora is, ami csak a cím és telekméret paramétereket fogadja. Ez a konstruktor az irányárat állítsa 1.000.000 Ft-ra és a típust „ismertelenre” alapértelmezettként.
* Készítse el a ToString metódust, ami visszaadja az osztály adattagjait a mintának megfelelően.: "Budapest, III. kerület-i, 75 m2-es panellakás eladó 140.000.000 Ft-ért."

Készítsen **Telek** osztályt, melyet az ingatlan osztályból származtasson!

* Újabb adatmezője: beépíthető-e (logikai), ehhez készítsen property-t.
* Legyen egy konstruktora, melynek paraméterei: cím, telekméret és a négyzetméterenkénti árát a teleknek forintban értelmezve. A konstruktor hívja meg az ősosztály konstruktorát a megfelelő paraméterek továbbadásával. Az árat állítsd be a telekméret \* négyzetméterenkénti ár képlet alapján, a típus pedig legyen mindig "Telek"! A beépíthetőséget véletlenszerűen döntse el!
* Továbbá definiálja felül az osztály a ToString metódusát úgy, hogy a visszaadott sztring-be kerüljön be az is, hogy a telek beépíthető-e. Minta: "Pécel-i, 500 m2-es telek eladó 13.000.000 Ft-ért (Beépíthető)."

Készítsen **Kerteshaz** osztályt, melyet a telek osztályból származtasson!

* Újabb adatmezője: lakóterület. Ennek kezeléséhez is készítsen property-t. Beállításkor csak akkor lehessen új értéket megadni, ha az érték nagyobb 10-nél és kisebb a teljes telekméret 80%-ánál!
* Készítsen egy getKertTerület metódust is, ez adja vissza a kiszámított kert területet! A kert területe kiszámítható, ha a telekméretből levonjuk a lakóterület méretét (mindkettő m2-ben van értelmezve).
* Legyen egy konstruktora, ami megkapja a cím, telekméret, lakóterület és a négyzetméterenkénti ár paramétereket! A konstruktor ezeket részben adja tovább az ősosztály konstruktorának, és végezze el a lakóterület és típus tagváltozók beállítását is!
* Definiálja felül a ToString metódust, hogy az újabb lakóterület tagváltozó tartalma is bekerüljön a visszaadott sztringbe.

A főprogramban vegyen fel egy legalább 3 elemű kollekciót, ami Ingatlan-ok tárolására alkalmas! A felhasználótól beolvasott adatok alapján készítsen objektumokat és ezekkel töltse fel a kollekciót! Ezt a következőképp oldja meg: először olvassa be, hogy milyen típusú ingatlant szeretne felvenni (telek vagy kertes ház), majd olvassa be az adott típusú ingatlanhoz tartozó összes adatot! Ezek után már létrehozhatók az objektum példányok és ezeket kell a listában elhelyezni!

Listázza ki az összes ingatlant!

Listázza ki az összes olyan eladó kertesház adatait, melyek irányára 50 és 70 millió Ft közt van!

Határozza meg a legnagyobb kertméretű kertes házat, és a legnagyobb beépíthető telket! Írja ki a képernyőre ezeknek az objektumoknak az adatait!

## feladat:

Készítsen alkalmazást, melyben egy garázsban tárolt tárgyakat kezel!

Készítsen **Tárgy** osztályt, az osztály legyen absztrakt!

* Adattagok: név és mennyiség. (private) Mindkét adattaghoz készítsen gettert és settert, a mennyiség nem lehet negatív! (public)
* Az osztály konstruktora paraméterül kapja a tárgy nevét és mennyiségét!
* Készítsen absztrakt, string visszatérési értékű mindenAdat metódust, ezt nem kell megvalósítania!

Készítsen **Kincs** osztályt, melyet a tárgy osztályból származtasson!

* Az osztály plusz adattagja a kincs értéke legyen! (private) Az értékhez tartozik egy getter! (public)
* Az osztály konstruktora minden adattag értékét paraméterként kapja meg!
* A felüldefiniált mindenAdat metódus az osztály adattagjainak értékét egy string-be fűzve adja vissza!

Készítsen **Lom** osztályt, melyet a tárgy osztályból származtasson!

* Az osztály plusz adattagja egy logikai értékű mező, amiben azt tároljuk, hogy a tárgy jó-e még valamire. (private) Készítsen hozzá gettert és settert! (public)
* A felüldefiniált mindenAdat metódus az osztály adattagjainak értékét egy string-be fűzve adja vissza! A logikai változó értékét szövegesen jelenítse meg!

Készítsen **Garázs** osztályt!

* Az osztálynak két adattagja legyen. (private) Az egyik a garázsban lévő tárgyak eladásából származó bevételt tároló egész típusú mező, a másik a garázsban lévő tárgyak listája.
* A bevételt tartalmazó adattaghoz készítsen gettert és settert! (public) Az írható tulajdonság a bevétel növelésére szolgáljon!
* A konstruktorban állítsa a bevételt 0-ra és hozza létre a listát!
* Készítsen tárgyBerak metódust, ami a paraméterül kapott tárgyat elhelyezi a garázsban. Ha ilyen tárgyból már van, akkor csak a tárolt mennyiséget növelje, egyébként szúrja be a listába új elemként!
* Készítsen tárgyKidob metódust, melynek paramétere a kidobandó tárgy neve! Törölje a tárgyat a listából, feltételezheti, hogy a tárgy létezik!
* Készítsen mindenKincs és mindenLom metódusokat, melyek rendre a garázsban tárolt kincsek, illetve lomok listáját adja vissza egy string-be fűzve!
* Készítsen további metódusokat az alábbi feladatok megoldásához!

1. Valaki kerékpárt akar venni, ha van a garázsban, írjuk ki az adatokat, különben mondjuk, hogy sajnos nincs!
2. Készítsen függvényt, mely visszaadja a garázsban tárolt kincsek összértékét!
3. Mennyi a valamire még jó lomok aránya az összes lomhoz viszonyítva? (egy tizedesjegyre kerekítve)
4. Készítsen függvényt, ami visszaadja a legértékesebb kincs minden adatát!
5. Készítsen metódust, amivel lomtalanítani tudunk! Csak a semmire se jó lomokat dobja ki!
6. Készítsen metódust, amivel eladhat egy tárgyat! A metódus paraméterei, hogy miből mennyit szeretne a vevő, visszatérési értéke az eladási ár (érték\*eladott\_mennyiség). A metódus csökkentse a garázsban lévő mennyiséget, ha a megmaradt mennyiség 0, akkor „dobja ki a tárgyat” a garázsból!

**A főprogramban:**

Példányosítson egy garázst! Helyezzen el a garázsban néhány tárgyat:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| típus | név | mennyiség | érték | jó még valamire |
| kincs | toronyóra lánccal | 1 | 5000 |  |
| kincs | kerékpár | 2 | 12000 |  |
| lom | szakadt befőttesgumi | 1000 |  | nem |
| kincs | porcelán étkészlet | 5 | 15000 |  |
| lom | lyukas függöny | 3 |  | nem |
| kincs | bélyeggyűjtemény | 1 | 25000 |  |
| kincs | hintaló | 3 | 5000 |  |
| lom | rossz fűnyíró | 2 |  | igen |
| lom | köteg újságpapír | 1 |  | igen |

Hívja meg az a-f feladatok megoldására készített metódusokat, a futás eredményét írja a képernyőre a feladat sorszáma után!

A lomtalanítás előtt és után listázza ki a garázsban lévő lomokat!

A garázsvásár lebonyolítása során listázza ki a kincseket. Kérje be a felhasználótól, hogy miből mennyit kíván vásárolni, az eladott kincs értékének és mennyiségének megfelelően növelje a garázs bevételét! A felhasználó addig vásárolhat, amíg el nem köszön!

## feladat:

Készítsen alkalmazást, melyben felhasználók adatait kezeli!

Készítsen **Felhasználó** nevű osztályt, az osztály legyen absztrakt!

* Adatmezői: név (string), jelszó (string), munkakör (string), bejelentkezve (bool). Mindegyik mező rejtett, és a név mező pedig csak olvasható! Legyen az osztálynak egy statikus lista adattagja (string) ami a rendszerben történő eseményeket naplózza!
* Legyen konstruktora, ami paraméterében kapja a felhasználó nevét, és ezzel inicializálja a név mező értékét! A jelszót olvassa be a felhasználótól, és alapértelmezetten ne legyen bejelentkezve a felhasználó!
* Legyen egy Adatok() metódusa, ami formázottan visszaadja a felhasználó adatait! Ez a metódus legyen virtuális!
* Legyen egy SzerepkorLeiras(), ami szöveg típusú absztrakt metódus!
* Legyen egy Bejelentkezés() metódusa! Ez olvasson be egy sztringet (jelszót) és ha ez egyezik a jelszó adatmező értékével, akkor állítsa be igazra a bejelentkezve adatmező értékét! Csak akkor kezdje meg a működését, ha még nem volt bejelentkezve a felhasználó! Ha rossz jelszót adtak meg, akkor ismételjen kérjen be sztringeket, amíg jót nem adtak meg, vagy 3-nál többször próbálkozott a felhasználó! Ha 3-nál több próbálkozás történt, írjon ki hibaüzenetet! Sikeres bejelentkezés esetén adjon vissza igaz értéket, egyébként hamisat!
* Legyen egy Kijelentkezés() metódusa! Ez vizsgálja meg a bejelentkezve adatmező aktuális értékét. Ha ez hamis, akkor írja ki, hogy nem is volt bejelentkezve a felhasználó, ellenkező esetben állítsa hamisra az értékét, és írja, ki, hogy sikeres kijelentkezés!
* Legyen egy protected láthatóságú NaploHozzaad(string uzenet) metódusa, ami a statikus adattaghoz hozzáad egy új eseményt a következő formában: dátum – uzenet! A dátumot DateTime.Now segítségével kérje le!
* Egészítse ki a konstruktort, amikor új felhasználót veszünk fel a statikus adattaghoz a megfelelő metódus felhasználásával adjon hozzá egy új üzenetet: pl: „Felhasználó létrehozva: Klárika.”. Egészítse ki a bejelentkezés és kijelentkezés metódust is a naplózással!

Például: (A megfelelő név beillesztésével)

"Sikeres bejelentkezés: Géza."

"Sikertelen bejelentkezési próbálkozások (3): Géza."

"Sikertelen kijelentkezési kísérlet: Géza.”

"Kijelentkezés: Géza.”

* Készítsen egy NaplotMegjelenit() statikus metódust, ami a minta szerint formázottan kiírja a rendszernapló adatait!

Készítsen **Rendszergazda** osztályt, melyet a felhasználó osztályból származtasson!

* Legyen egy konstruktora, ami paraméterében a felhasználó nevét kapja, és ezt továbbadja az ősosztály konstruktorának! Emellett állítsa be a munkakör adatmező tartalmát a "Rendszergazda" sztringre!
* Definiálja felül a SzerepkorLeiras() metódust, úgy, hogy a következőt adja vissza eredményül: „A rendszergazda feladata a rendszer karbantartása és biztonságának felügyelete.”

Készítsen **Ügyintéző** osztályt, melyet a felhasználó osztályból származtasson!

* Tartalmazzon az örökölt adatmezők mellett egy sorszám adatmezőt is, ami az ügyintéző sorszámát jelzi!
* Legyen egy konstruktora, ami nevébe kapja a felhasználó nevét, és egy sorszámot! A név paramétert továbbadja az ős konstruktorának, a sorszám adatmezőt pedig beállítja a paraméternek megfelelően!
* Definiálja felül az Adatok() virtuális metódust! Az alapadatok mellett írja ki az újonnan felvett sorszám adatmező tartalmát is!
* Definiálja felül a SzerepkorLeiras() metódust, úgy, hogy a következőt adja vissza eredményül: „Az ügyintéző feladata az ügyek kezelése és nyilvántartása (Sorszám: 3).”

**A főprogramban:**

Hozzon létre heterogén kollekciót, melynek elemei az alábbi felhasználók legyenek!

|  |  |  |
| --- | --- | --- |
| típus | név | sorszám |
| ügyintéző | Klárika | 3 |
| ügyintéző | Ilus | 2 |
| rendszergazda | Géza |  |
| ügyintéző | Marika | 4 |

Keresse meg a listában a rendszergazdát és hívja meg a rendszergazda Bejelentkezés() metódusát! Ha ez igaz értéket ad vissza, akkor írassa ki a felhasználókat tároló kollekció eleminek adatait és jelentkeztesse ki a rendszergazdát!

Jelenítse meg a rendszernaplót a megfelelő statikus metódus meghívásával!

Hívja meg két véletlenszerűen választott felhasználó SzerepkorLeiras() metódusát! Oldja meg, hogy ne lehessen ugyanannak a két felhasználónak a metódusát meghívni!

### A képen szöveg, képernyőkép, Betűtípus látható Automatikusan generált leírásMinta