**ÖRÖKLŐDÉS, INTERFACE, POLIMORIZMUS**

1. *Feladat*

Írj egy **Arlap**interfészt, ami egy ***MennyibeKerul()*** metódust tartalmaz, mely egy lebegőpontos értékkel (az áru árával) tér vissza.

• Írj egy **Peksutemeny**absztrakt osztályt, ami implementálja az **Arlap**interfészt.

**–** Az osztály a következő lebegőpontos adattagokkl rendelkezik: **mennyiseg, alapar**. Az alapár csak ebből az osztálból legyen látható, míg a mennyiség legyen látható a leszármazott osztályokban is (használd a lehető legszűkebb láthatóságot).

**–** Az osztály rendelkezzen paraméteres konstruktorral, ami beállítja az adattagok értékeit.

**–** Legyen ezen felül egy **megkostol**public láthatóságú absztrakt függvénye, ami nem tér vissza értékkel.

**–** Az osztály valósítsa meg az implementált interfész metódusát. Egy péksütemény értéke az alapárának és a mennyiségének szorzatából számolható ki.

**–** Az osztály legyen továbbá szöveges formára alakítható, kiírva az adattagok értékét. (pl. "X db - Y Ft", ahol Y azt jelenti, mennyibe kerül összesen).

• Írj egy **Pogacsa**osztályt, ami a **Peksutemeny**leszármazottja.

**–** Az osztálynak egy két paraméteres konstruktora legyen, ami a mennyiséget és az alapárat kéri el, majd állítja be.

**–** Pogácsa megkóstolásakor mindig csökkenjen felére a mennyisége.

**–** Az osztály legyen továbbá szöveges formára alakítható. Az objektum tulajdonságain kívül írja ki azt is, hogy Pogacsa osztályról van szó. (pl. "Pogacsa X db - Y Ft", ahol Y azt jelenti, mennyibe kerül összesen). Itt használd fel az ososztály *ToString* metódusát is!

• Írj egy **Kave**osztályt, ami implementálja az **Arlap**interfészt.

**–** Az osztály egy *habosE* private láthatóságú logikai adattaggal rendelkezik.

**–** Legyen továbbá egy *CSESZEKAVE* konstansa, ami egy kávé árát adja meg, ennek értéke 180.

**–** Az osztály rendelkezzen egy paraméteres konstruktorral, ami beállítja az adattag értékét. A metódusai az alábbiak szerint legyenek megvalósítva: egy rendes kávé ára a *CSESZEKAVE* konstans értéke, a habos kávé esetén ennek 1,5-szerese.

**–** Az osztály legyen szöveges formára alakítható. (pl. "Habos/Nem habos kave - X Ft").

• Írj egy **Pekseg**nevű osztályt.

**–** Az osztálynak legyen egy tárolója (tetszőleges kollekció, pl. lista), ahova **Arlap** típusú objektumokat tárol.

Az osztály rendelkezzen egy **Vasarlok**statikus metódussal, ami egy fájl elérési útját várja paraméterül, visszatérése pedig void. A metódus feladata, hogy a fájlból beolvasott sorokat feldolgozza, és létrehozzon belolük **Pogacsa**, vagy **Kave**objektumpéldányokat. A fájl egy sorában az adott objektum tulajdonságai szerepelnek. A létrehozott objektumpéldányokat közös tárolóban tárold le.

**–** Készíts továbbá egy ***EtelLeltar***statikus metódust, ami végigmegy a tárolóban tárolt elemeken, és az összes **Pogacsa** típus objektum információit kiírja egy "leltar.txt" fájlba.

**–** Hívd meg a main függvényben sorban a fenti két metódust. A **Vasarlok**metódus paraméterét parancssori argumentumból kérd be.

**–** Minden esetleges kivételt (foleg: ***IOException***, vagy a bemenet beolvasáakor/konvertálásakor eloforduló kivételek) kezelj le vagy kivétel speciﬁkációval *try* blokkban!

Egy minta fájl felépítése az alábbi:

* Pogacsa 10 150
* Kave habos
* Kave nem\_habos

1. *Feladat*

Írj **KisGepjarmu**interfészt, ami egy **HaladhatItt**metódust deklarál. A metódus egy logikai értékkel térjen vissza (haladhat -e ezen az úton a gépjármű), és egy egész számot (sebességet) kérjen paraméterként.

• Készíts egy absztrakt **Jarmu**osztályt.

**–** Egy járműnek legyen aktuális sebessége (int) és rendszáma (string). Az aktuális sebesség látszódjon a leszármazott osztályokban is (használd a lehető legszűkebb láthatóságot), míg a rendszám adattagot csak ebből az osztályból lehessen elérni. Írj konstruktort két paraméterrel, ami beállítja az adattagokat.

**–** Készíts egy **GyorshajtottE**absztrakt metódust, ami egy sebességkorlátot (int) kér paraméternek, és logikai értékkel tér vissza attól függően, hogy az adott jármű gyorshajtott-e.

**–** Készíts **ToString**metódust, ami az alábbi módon alakítja szöveges formára az objektumot: "rendszam - X km/h" (ahol rendszam a jármu rendszáma, X pedig az akutális sebessége).

• Kézsíts egy **Robogo**osztályt, ami a **Jarmu**osztályból származik és implementálja a **KisGepjarmu**interfészt.

**–** A robogónak legyen egy maximális sebesség adattagja. A robogó konstruktora a rendszámot, az aktuális sebességet és a maximális sebességet kérje el paraméterül, és ez alapján hozza létre az objektumot.

**–** A ***GyorshajtottE***metódus nézze meg, hogy a jármu aktuális sebessége fölötte van-e a paraméterként kapott korlátnak, és ennek megfelelően térjen vissza logikai értékkel.

**–** A **HaladhatItt**metódust hamis értékkel térjen vissza, ha a robogó maximális sebessége nagyobb, mint a kapott paraméter, ellenkező esetben igaz legyen a visszatérés.

**–** Bővítsd ki az örökölt *ToString* metódust, hogy az alábbiakat adja vissza: "Robogo: rendszam - X km/h" (ahol rendszam a jármu rendszáma, X pedig az akutális sebessége) Használd fel az ősosztály *ToString* metódusát is!

• Kézsíts egy **AudiS8**osztályt, ami a **Jarmu**osztályból származik.

**–** Az **Audinak** legyen egy **lezerblokkolo** (*bool*) paramétere. Konstruktora a rendszámát, az aktuális sebességét kérje el, és hogy van-e lézerblokkolója, és ezek alapján hozza létre az objektumot.

**–** A **GyorshajtottE**metódus nézze meg, hogy a jármű aktuális sebessége fölötte van–e a paraméterként kapott korlátnak, és ennek megfelelően térjen vissza logikai értékkel. Ha a jármű rendelkezik lézerblokkolóval, úgy ehelyett mindig hamissal térjen vissza.

**–** Bovítsd ki az örökölt T*oString* metódust, hogy az alábbiakat adja vissza: "Audi: rendszám - X km/h" (ahol rendszám a jármu rendszáma, X pedig az akutális sebessége). Használd fel az ősosztály *ToString* metódusát is!

• Kézsíts egy **Orszagut**osztályt.

**–** Az osztálynak legyen egy tárolója (tetszőleges kollekció, pl. lista), amiben **Jarmu** típusú objektumokat tárol.

**–** Legyen tovább egy statikus **JarmuvekJonnek**metódusa. Ez egy fájl elérési útját várja paraméterül. A metódus feladata, hogy a fájlból beolvasott sorokat feldolgozza, és létrehozzon belolük **Robogo** és **AudiS8** objektumpéldányokat, amiket hozzáad a tárolóhoz.

**–** Legyen egy statikus ***KiketMertunkBe***metódus is, ami végigmegy a tárolón, és kiírja egy "buntetes.txt" fájlba a benne lévo járműveket szöveges formában, és az Audi típusúakra pluszban kiírja azt is, hogy gyorshajtottak-e, míg a robogó típusúakra azt, hogy haladhatnak-e ezen az úton. Mindkét esetben 90 legyen a paraméter.

A main metódusban hívd meg a **JarmuvekJonnek**metódust egy elérési úttal, majd hívd meg a **KiketMertunkBe** metódust is.

Minden esetleges kivételt (főleg az IOException, de ﬁgyelj a bemenet feldolgozása közben tömbtúlindexelésekre és a számok átalakítása közben fellépő hibákra is) kezelj le vagy kivétel speciﬁkációval, vagy try blokkban!

Egy minta fájl felépítése az alábbi: (típus;rendszám;aktuális sebesség;blokkoló/max. sebesség)

robogo;"";40;60

audi;AAA-111;200;true

robogo;"";80;65

audi;AAA-111;130;false

1. *Feladat*

Írj egy **Szuperhos**interfészt, ami egy **L*egyoziE***metódust tartalmaz. A metódus paramétere egy **Szuperhos**, és egy logikai értékkel tér vissza. Legyen egy **MekkoraAzEreje**metódusa is, ami nem kér paramétert, és a **Szuperhos**erejét fogja visszaadni.

• Írj **Milliardos**interfészt, ami egy visszatérés nélküli **KutyutKeszit** metódust tartalmaz

• Írj egy **Bosszuallo**absztrakt osztályt, ami implementálja a **Szuperhos**interfészt.

**–** Az osztály a következő *protected* láthatóságú adattagokkal rendelkezik: egy lebegőpontos szupererő, és egy logikai vanEGyengesege.

**–** Az osztály rendelkezzen paraméteres konstruktorral, ami beállítja az adattagokat.

**–** Legyen egy public **MegmentiAVilagot**absztrakt metódusa, ami egy logikai értékkel tér vissza. Valósítsd meg továbbá az interfész metódusait. Az erő lekérdezésekor add vissza a **Bosszuallo**szupererejét. Egy Bosszuallo egy Bosszuallo szuperhőst akkor tud legyőzni, ha annak van gyengesége, és ereje kisebb, mint az övé. Batman-t csak akkor tudja legyőzni, ha ereje kétszer nagyobb.

**–** Az osztály legyen szöveges formára alakítható, kiírva az adattagok értékét.

• Írj egy ***Vasember***osztályt, ami a ***Bosszuallo***leszármazottja, és megvalósítja a ***Milliardos***interfészt.

**–** Az osztálynak egy default konstruktora legyen, ami beállítja a Vasember tulajdonságait. A Vasember szuperereje 150, és van gyengesége.

**–** Ha a Vasember kütyüt készít, akkor szuperereje nőjön egy 0-10 közötti véletlenszerű lebegőpontos számmal.

**–** A Vasember akkor menti meg a világot, ha a szuperereje nagyobb, mint 1000.

**–** Az osztály legyen továbbá szöveges formára alakítható. Az adattagok értékein kívül írja ki azt is, hogy a Vasemberrol van szó.

• Írj egy **Batman**osztályt, ami implementálja a **Szuperhos**és **Milliardos**interfészeket.

**–** Az osztálynak legyen egy lebegőpontos lelemenyesseg adattagja.

**–** Az osztály rendelkezzen egy default konstruktorral, ami 100-ra állítja az adattag értékét.

**–** A metódusai az alábbiak szerint legyenek megvalósítva: Batman ereje a leleményességének kétszeresével egyezik meg, és bármilyen Szuperhost képes legyőzni, akinek ereje kisebb, mint Batman leleményessége. Ha Batman kütyüt készít, akkor a leleményessége 50-el nő.

**–** Az osztály legyen szöveges formára alakítható, ami kiírja, hogy Batmanrol van szó, és megadja a leleményességét.

• Írj egy **Kepregeny**nevű osztályt. Az osztály rendelkezzen egy ***Szereplok***statikus függvénnyel, ami egy fájl elérési útját várja paraméterül, visszatérése pedig void. A metódus feladata, hogy a fájlból beolvasott sorokat feldolgozza, és létrehozzon belolük **Batman**, vagy **Vasember**objektumpéldányokat, majd ezekre meghívja a **KutyutKeszit** metódust annyiszor, ahányszor az aktuális sor írja. Ezeket egy közös kollekcióban tárold le. Készíts továbbá egy **Szuperhosok**statikus metódust, ami végigmegy a tárolóban tárolt szuperhosökön, és kiírja őket. Hívd meg a main függvényben sorban a fenti két metódust. Minden esetleges kivételt (főleg: *IOException*) kezelj le vagy kivétel speciﬁkációval, vagy try blokkban!

Egy minta fájl felépítése az alábbi:

Vasember 5

Batman 8

1. *Feladat*

Írj egy **EroErzekeny** interfészt, ami egy **LegyoziE** metódust tartalmaz. A metódus paramétere egy **EroErzekeny** objektum (interfész), és egy logikai értékkel tér vissza. Legyen egy **MekkoraAzEreje** metódusa is, ami nem kér paramétert, és az **EroErzekeny** erejét fogja visszaadni.

• Írj **Sith** interfészt, ami egy visszatérés nélküli **EngeddElAHaragod** metódust tartalmaz

• Írj egy **Jedi** absztrakt osztályt, ami implementálja az **EroErzekeny** interfészt.

– Az osztály a következő *protected* láthatóságú adattagokkal rendelkezik: egy lebegőpontos erő, és egy logikai atallhatE (átállhat -e a sötét oldalra).

– Az osztály rendelkezzen paraméteres konstruktorral, ami beállítja az adattagokat. Legyen egy public **MegteremtiAzEgyensulyt** absztrakt metódusa, ami egy logikai értékkel tér vissza. Valósítsd meg továbbá az interfész metódusait. Az erő lekérdezésekor add vissza a Jedi erejét. Egy Jedi egy másik Jedi objektumot akkor tud legyőzni, ha az átállhat a sötét oldalra, és ereje kisebb, mint az övé. Az Uralkodo objektumot cask akkor tudja legyőzni, ha ereje kétszer nagyobb.

– Az osztály legyen szöveges formára alakítható, kiírva az adattagok értékét.

• Írj egy **AnakinSkywalker** osztályt, ami a **Jedi** leszármazottja, és megvalósítja a **Sith** interfészt.

– Az osztálynak egy default konstruktora legyen, ami beállítja Anakin tulajdonságait. Anakin ereje 150, és átállhat a sötét oldalra.

– Ha Anakin elengedi a haragját, akkor ereje egy 0-10 közötti véletlenszerű lebegőpontos számmal nő.

– Anakin akkor teremti meg az egyensúlyt az erőben, ha ereje nagyobb, mint 1000.

– Az osztály legyen továbbá szöveges formára alakítható. Az adattagok értékein kívül írja ki azt is, hogy a Anakin Skywalkerről van szó.

• Írj egy **Uralkodo** osztályt, ami implementálja az **EroErzekeny** és **Sith** interfészeket.

– Az osztálynak legyen egy lebegőpontos **gonoszsag** adattagja.

– Az osztály rendelkezzen egy default konstruktorral, ami 100-ra állítja az adattag értékét. A metódusai az alábbiak szerint legyenek megvalósítva: az Uralkodo ereje a gonoszságának kétszeresével egyezik meg, és bármilyen **EroErzekeny**-t képes legyőzni, akinek ereje kisebb, mint az **Uralkodo** gonoszsága. Ha az **Uralkodo** elengedi a haragját, akkor gonoszsága 50-el nő.

– Az osztály legyen szöveges formára alakítható, ami kiírja, hogy az Uralkodóról van szó, és megadja a gonoszságát.

• Írj egy **StarWars** nevű osztályt. Az osztály rendelkezzen egy **Szereplok** statikus függvénnyel, ami egy fájl elérési útját várja paraméterül, visszatérése pedig void. A metódus feladata, hogy a fájlból beolvasott sorokat feldolgozza, és létrehozzon belőlük **AnakinSkywalker**, vagy **Uralkodo** objektumpéldányokat, majd ezekre meghívja az **EngeddElAHaragod** metódust annyiszor, ahányszor az aktuális sor írja. Ezeket egy közös kollekcióban tárold le. Készíts továbbá egy **Sithek** statikus metódust, ami végigmegy a tárolóban tárolt objektumokon, és kiírja őket. Hívd meg a main függvényben sorban a fenti két metódust. Minden esetleges kivételt (főleg: *IOException*) kezelj le vagy kivétel speciﬁkációval, vagy *try* blokkban!

Egy minta fájl felépítése az alábbi:

Anakin 5

Uralkodo 8

1. *Feladat*

• Írj egy **Űrhajó** interfészt, ami egy **LegyorsuljaE** metódust tartalmaz. A metódus paramétere egy **Űrhajó** objektum, és egy logikai értékkel tér vissza. Legyen egy **MilyenGyors** metódusa is, ami nem kér paramétert, és az **Űrhajó** gyorsaságát fogja visszaadni.

• Írj **Hiperhajtomu** interfészt, ami egy visszatérés nélküli **HiperUgras** metódust tartalmaz.

• Írj egy **LazadoGep** absztrakt osztályt, ami implementálja az **Űrhajó** interfészt.

– Az osztály a következő protected láthatóságú adattagokkal rendelkezik: egy lebegőpontos **sebesség**, és egy logikai **meghibasodhatE**.

– Az osztály rendelkezzen paraméteres konstruktorral, ami beállítja az adattagokat.

– Legyen egy public **ElkapjaAVonosugar** absztrakt metódusa, ami egy logikai értékkel tér vissza.

– Valósítsd meg továbbá az interfész metódusait. A gyorsaság lekérdezésekor add vissza a LazadoGep sebességét. Egy LazadoGep egy másik LazadoGep objektumot akkor tud legyorsulni, ha az meghibásodhat, és a gyorsasága kisebb, mint az övé. A MilleniumFalcon objektumot csak akkor tudja legyőzni, ha gyorsasága kétszer nagyobb.

– Az osztály legyen szöveges formára alakítható, kiírva az adattagok értékét.

• Írj egy **XWing** osztályt, ami a **LazadoGep** leszármazottja, és megvalósítja az **Hiperhajtomu** interfészt.

– Az osztálynak egy default konstruktora legyen, ami beállítja az X-Wing tulajdonságait. Az X-Wing sebessége 150, és meghibásodhat.

– Ha az X-Wing hiperugrást végez, akkor sebessége egy 0-100 közötti véletlenszerű lebegőpontos számmal nő.

– Az X-Wing et akkor kapja el a vonósugár, ha meghibásodhat, és sebessége kisebb, mint 10000.

– Az osztály legyen továbbá szöveges formára alakítható. Az adattagok értékein kívül írja ki azt is, hogy egy X-Wingrol van szó.

• Írj egy **MilleniumFalcon** osztályt, ami implementálja az **Űrhajó** és **Hiperhajtomu** interfészeket.

– Az osztálynak legyen egy lebegőpontos tapasztalat adattagja.

– Az osztály rendelkezzen egy default konstruktorral, ami 100-ra állítja az adattag értékét. A metódusai az alábbiak szerint legyenek megvalósítva: a Millenium Falcon gyorsasága a tapasztalatának kétszeresével egyezik meg, és bármilyen Űrhajó-t képes legyorsulni, akinek gyorsasága kisebb, mint a Falcon gyorsasága. Ha a Millenium Falcon hiperugrást végez, akkor tapasztalata 500-al nő.

– Az osztály legyen szöveges formára alakítható, ami kiírja, hogy a Millenium Falconról van szó, és megadja a tapasztalatát.

• Írj egy **StarWars** nevű osztályt. Az osztály rendelkezzen egy **Űrhajók** statikus függvénnyel, ami egy fájl elérési útját várja paraméterül, visszatérése pedig void. A metódus feladata, hogy a fájlból beolvasott sorokat feldolgozza, és létrehozzon belolük **XWing**, vagy **MilleniumFalcon** objektumpéldányokat, majd ezekre meghívja a **HiperUgras** metódust annyiszor, ahányszor az aktuális sor írja. Ezeket egy közös kollekcióban tárold le. Készíts továbbá egy **Hangar** statikus metódust, ami végigmegy a tárolóban tárolt objektumokon, és kiírja őket. Hívd meg a main függvényben sorban a fenti két metódust. Minden esetleges kivételt (foleg: *IOException*) kezelj le vagy kivétel speciﬁkációval, vagy *try* blokkban!

Egy minta fájl felépítése az alábbi:

XWing 4

MilleniumFalcon 18

1. *Feladat*

• Írj egy **Ingatlan** interfészt, ami egy paraméter nélküli **OsszesKoltseg** metódust tartalmaz. A metódus az ingatlan költségét adja majd vissza.

• Írj egy **Berelheto** interfészt, aminek három metódusa van. Az egyik a **MennyibeKerul**, ami egy egész számot (hónapok száma) kap paraméterül, és egy számmal (a bérleti díjjal) tér majd vissza. A másik a paraméter nélküli **LefoglaltE**, ami egy logikai értékkel tér majd vissza, hogy lefoglalták -e már az adott dolgot. A harmadik **Lefoglal**, ami egy számot (lefoglalni kívánt hónapok száma) vár paraméterül, és logikai értékkel tér vissza attól függően, hogy sikeres volt -e a foglalás.

• Írj egy **Lakas** absztrakt osztályt, ami implementálja az **Ingatlan** interfészt.

– Az osztály a következő adattagokkal rendelkezik: terület (terulet), szobák száma (szobaSzam), lakók száma (lakok), négyzetméter ár (nm/ár). Az adattagok legyenek láthatóak a leszármazott osztályokban is! Az osztály rendelkezzen paraméteres konstruktorral, ami beállítja az adattagok értékét.

– Készíts egy egész paramétert váró **Bekoltozik** absztrakt metódust, ami embereket költöztet a lakásba. A metódus logikai értékkel tér vissza attól függően, hogy sikeres volt -e a beköltözés.

– Valósítsd meg az interfész metódusát, ami a terület és a négyzetméterenkénti ár alapján visszaadja a lakás teljes költségét.

– Készíts egy paraméter nélküli **LakokSzama** metódust, ami visszadja a lakásban lakók számát.

– Készíts egy ToString metódust, ami stringgé alakítja az osztály tulajdonságait.

• Írj egy **Alberlet** osztályt, ami a **Lakas**-ból származik és implementálja a **Berelheto** interfészt.

– Az örökölteken kívül egy foglalt hónapok szama adattaggal rendelkezik, ami megadja, hogy hány hónapra van lefoglalva az albérlet. Készíts parameters konstruktort, ami az ősosztály hasonló konstruktorát használva beállítja az adattagok értékét. A foglalt hónapok és a lakók száma kezdeti értéke 0 legyen!

– Implementáld az interfész első metódusát. Az albérlet egy főre jutó havi költsége a lakás összes költségének és a lakók számának hányadosa, a metódus ez alapján adja vissza a kívánt időszak költségét. Ha nincs a lakásban lakó, úgy -1 legyen a visszatérési érték!

– Implementáld az interfész második metódusát. Az albérlet akkor nincs lefoglalva, ha a foglalt hónapok száma 0.

– Implementáld az interfész harmadik metódusát. A lakás foglaltságától függően, módosítsd a foglalt hónapok számát (ha még nem volt lefoglalva), és térj vissza megfelelő értékkel.

– Implementáld az ősosztály absztrakt metódusát az alábbiak szerint: az albérletben egy szobában maximum 8 fő lakhat, és egy főre minimum 2 területnek kell jutnia. A beköltözés, és a metódus visszatérése ettől a feltételtől függjön. Ha a feltétel teljesül, úgy módosítsd a lakók számát is.

– Írd felül a ToString metódust, hogy már lefoglalt hónapok száma is szerepeljen benne. A metóduson belül használd az ősosztály toString-jét is.

• Írj egy **CsaladiApartman** osztályt, ami a **Lakas**-ból származik.

– Az örökölteken kívül egy **GyerekekSzama** adattaggal rendelkezik, ami megadja, hogy az összes lakóból mennyi a gyerek. Készíts paraméteres konstruktort, ami az ősosztály hasonló konstruktorát használva beállítja az adattagok értékét. A lakók és gyerekek száma kezdeti értéke is 0 legyen.

– Készíts egy paraméter nélküli **GyerekSzuletik** metódust. A metódus ellenőrizze le, hogy van-e két felnőtt lakója az apartmannak, és ha igen, úgy növelje a lakók és gyerekek számát is 1-el. Térjen vissza logikai értékkel attól függően, hogy megszületett –e a gyerek.

– Implementáld az ősosztály absztrakt metódusát az alábbiak szerint. Az apartman egy szobájában maximum 2 fő lakhat, és egy főre minimum 10 területnek kell jutnia. Gyerekek ebből a szempontból fél főnek számítanak csak. A beköltöző lakók mind felnőttek. A metódus visszatérése a fenti feltételtől függjön. Ha a feltétel teljesül, úgy módosítsd a lakók számát is.

– Írd felül a *ToString* metódust, hogy a gyereke száma is szerepeljen benne. A metóduson belül használd az ősosztály ToString-jét is.

• Írj egy **Garazs** osztályt, ami implementálja mindkét fenti interfészt.

– Az osztály a következő adattagokkal rendelkezik: terület, négyzetméter ár (nm/ár), fűtött -e, foglalt hónapok szama, áll-e benne auto. Készíts paraméteres konstruktort, ami beállítja az adattagok értékét. A garázsban kezdetben ne álljon autó, és a foglalt hónapok száma 0 legyen!

– Implementáld az **Ingatlan** interfész metódusát! A garázs összes költségét a terület és a négyzetméterenkénti ár alapján kapot.

– Implementáld a **Berelheto** interfész metódusait! A garázs havi költsége a terület és ár alapján számolható, ezt még 1,5-el kell szorozni, ha fűtött is. Ez alapján visszadható a kívánt időszak költsége. A garázs akkor foglalt, ha a foglalt hónapok száma nagyobb, mint 0, vagy áll benne autó. A garázs foglaltságától függően módosítsd a foglalt hónapok számát (ha még nem volt lefoglalva), és térj vissza megfelelő értékkel.

– Készíts egy paraméter nélküli **AutoKiBeAll** metódust. Ha már áll autó a garázsban, úgy az álljon ki, ha pedig nem áll, úgy álljon be. A metódus ne térjen vissza semmmivel.

– Készíts egy ToString metódust, ami stringgé alakítja az osztály tulajdonságait.

• Írj egy **Tarsashaz** nevű osztályt.

– Az osztálynak egy adattagja legyen: egy kollekció (pl. lista), ami képes **Lakas** és **Garazs** objektumokat is tárolni, valamint két egész szám, hogy maximálisan hány lakás és garázs lehet a társasházban. Írj konstruktort, ami a lakások és garázsok maximális számát kapja paraméternek, és beállítja a megfelelő adattagoat. Kezdetben a nincs lakás és garázs a házban.

– Írj két metódust **LakasHozzaad** és **GarazsHozzaad** néven. Mindegyik metódussal a neki megfelelő objektumot helyezhetjük majd el a házban, így paraméternek mindegyik a megfelelő típust várja. A metódusok adják hozzá a kollekcióhoz a paraméterként kapott objektumot, ha még nem értük el az adott típusból a maximális mennyiséget. Visszatérési értékük logikai legyen attól függően, hogy sikerült-e a hozzáadás, vagy sem.

– Írj egy paraméter nélküli **OsszesLako** metódust, ami visszadja, hogy a házban található lakásokban összesen hány lakó van.

– Írj egy paraméter nélküli **IngatlanErtek** metódust. A metódus számolja végig a házban lévő és használatban lévő lakások és garázsok értékét, és ezek összegével térjen vissza. Lakások esetén azok vannak használatban, amikben legalább 1 lakó lakik, míg garázs esetén az, ami le van foglalva.

• Írj egy **Hazmester** nevű osztályt. Az osztály rendelkezzen egy **Karbantart** statikus függvénnyel, ami egy fájl elérési útját várja paraméterül, visszatérési értéke pedig void. A metódus példányosítson egy **Tarsashaz**-at. A metódus feladata ezután, hogy a fájlból beolvasott sorokat (ingatlanokat) feldolgozza, és létrehozzon belőlük **Alberlet**, **CsaladiApartman** és **Garazs** objektumokat. Ezeket az obejtumokat a megfelelő függvény használatával adja is hozzá a létrehozott társasházhoz. Miután végzett a fájl feldolgozásával, írja ki konzolra, hogy mennyi a társasház összes értéke. Minden esetleges kivételt (főleg *FileNotFoundException* és *IOException*) kezelj le kivételspeciﬁkációval, vagy *try* blokkba. Egy minta fájl felépítése az alábbi:

Alberlet 50.2 5 13000

CsaladiApartman 62.8 2 40000

Garazs 10.3 5000 futott

1. *Feladat*

• Írj **Arak** absztrakt osztályt, ami két konstanst tartalmaz: EGYAGYAS és KETAGYAS, amiknek értékei 8000 és 12000.

• Írj egy **Kedvezmenyes** interészt. Legyen az interfésznek egy **KedvezmenytKer** absztrakt metódusa, aminek visszatérési típusa void, paramétere nincs.

• Készíts egy absztrakt **Szoba** osztályt, ami implementálja az **Arak** absztrakt osztályt.

– A következő adattagokkal rendelkezik: berletiDij, fekvoHely, lakok.

– Írj paraméteres konstruktort, ami a lakók számát kapja paraméternek, és azok alapján beállítja az adattagokat. Mindig annyi fekvőhely legyen, ahány lakó. A berletiDij értéke a konstansoktól függ, 2-nél több fekvőhely esetén minden plusz személyre az egyágyas felárral nő.

– A szobának legyen ToString metódusa, ami kiírja az adattagok értékeit.

– Legyen az osztálynak egy kikoltozik metódusa, ami egy egész számot vár paraméternek, és ennyivel csökkenti a lakók számát (de 0 alá ne csökkentse).

– Írj egy **Atkoltozik** absztrakt metódust is, ami egy Szoba és egy egész szám paraméterrel rendelkezik, és visszatérése void.

• Kézsítsd el az alábbi 3 osztályt:

– **Egyagyas**, ami a **Szoba** osztályból származik. Konstruktora ne kérjen paramétert, és egy 1 fős szobát hozzon létre az ősosztály konstruktora segítségével. Az **Atkoltozik** metódus **Ketagyas** illetve **Lakosztaly** típusú szobákba (objektum típusának eldöntése: *typeof(oszály amit vizsgálunk)* segítségével) költözhessen át a megadott számú ember, ha a célszobában van elég szabad hely. A két objektum vonatkozó adattagjait módosítsd ennek megfelelően.

A szobának legyen ToString metódusa, ami az ősosztály által kiírtak mellett azt is megmondja, hogy ez egy egyágyas szoba.

– **Ketagyas**, ami a **Szoba** osztályból származik, és implementalja a **Kedvezmenyes**. Konstruktora ne kérjen paramétert, és egy 2 fős szobát hozzon létre az ősosztály konstruktora segítségével. Kétágyas szobára kedvezmény kérhető, ha csak 1 ember lakik benne, ilyenkor a bérleti díjat változtassa az egyágyasnak díjnak megfelelően. Az **Atkoltozik** metódus **Lakosztaly** típusú szobáka (objektum típusának eldöntése: *typeof(oszály amit vizsgálunk)*) költözhessen át a megadott számú ember, ha a célszobában van elég szabad hely. A két objektum vonatkozó adattagjait módosítsd ennek megfelelően. A szobának legyen ToString metódusa, ami az ősosztály által kiírtak mellett azt is megmondja, hogy ez egy kétágyas szoba.

– **Lakosztaly**, ami a **Szoba** osztályból származik. Konstruktora a lakók számát kérje paraméterként, és a megadott férőhellyel hozza létre a szobát az ősosztály konstruktora segítségével. Lakosztályból nem költözik ki senki, az atkoltozik függvény ezt közölje a felhasználóval. A szobának legyen toString metódusa, ami az ősosztály által kiírtak mellett azt is megmondja, hogy ez egy lakosztály.

• Kézsíts egy **Panzio** osztályt, amiben:

– legyen egy statikus **SzobatKiad** függvény. A függvény egy fájl elérési útját várja paraméterül. A metódus feladata, hogy a fájlból beolvasott sorokat feldolgozza, és létrehozzon belőlük **Egyagyas**, **Ketagyas** vagy **Lakosztaly** objektumpéldányokat, amiket egy közös tárolóba ment le.

– legyen egy statikus **Berel** metódus is, ami végigmegy a tárolón, és kiírja konzolra a rajta lévő szobákat. Ha egy szobára kérhető kedvezmény, úgy előtte tegye ezt meg. Minden esetleges kivételt (főleg *FileNotFoundException* és *IOException*) kezelj le vagy kivétel speciﬁkációval, vagy *try* blokkban!

– Ha hozott létre **Egyagyas** szobát és **Lakosztalyt** is ilyen módon a függvény, úgy csökkentsd az első létrehozott **Lakosztaly** lakóinak számát 1-el, és költöztesd át az első létrehozott **Egyagyas** szobából a lakót ide.

– A main metódusban hívd meg a **SzobatKiad** függvényt egy parancssori argumentumból bekért elérési úttal, majd hívd meg a berel metódust is.

Egy minta fájl felépítése az alábbi:

egyagyas ketagyas lakosztaly;10