Programozás I. - 2. ZH, 2015.05.04	**Név: .....**
A. csoport	**EHA: .....**
25 pont	**Gyakorlatvezető: .....**

1. Készíts egy **Hallgato** osztályt! Tárold egy hallgató **nevét**, **EHA kódját**, valamint egy valós értékben a hallgató **szorgosságát** (ez 0-1 közötti érték lehet, beállításkor ellenőrizd le, ha 0-nál kisebbet adnak meg 0 legyen az értéke, ha 1-nél nagyobbat akkor pedig 1). Ezen felül egy String halmazban tárold le a hallgató **kedvenc tantárgyait**.
Egy darab paraméteres konstruktora legyen, mely minden adattagot paraméterként megkap. Minden adattag privát láthatóságú, ezek megfelelő publikus metódusok segítségével legyenek lekérdezhetőek és beállíthatóak. Legyen egy **isKedvencTantargy** metódusa, mely egy tantárgy nevét várja paraméterül, és logikai értékkel tér vissza annak megfelelően, hogy a hallgató kedvencei között van-e az adott tantárgy.
Készíts el egy **ErdektelensegbeFulladt** nevű kivétel osztályt, mely rendelkezzen paraméter nélküli és paraméteres konstruktorral is! ***(7 pont)***

2. Készíts egy **Kurzus** nevű **absztrakt** osztályt! Privát adattagokban tárold le a kurzus **nevét**, **maximális létszámát**, valamint egy valós értékben az **oktató előadóképességét** (0-1 közötti érték, ezt is ellenőrizd le beállításkor). Emellett tárold le egy listában a **jelentkezett hallgatókat.** Készíts paraméteres konstruktort és getter/setter metódusokat az osztályhoz!
A fentiek mellett az osztály rendelkezzen egy **absztrakt kurzusTartas** metódussal is, mely **ErdektelensegbeFulladt** típusú kivételt dobhat. **(4 pont)**

3. Örököltess a fenti osztályból egy **Eloadas** és egy **Gyakorlat** osztályt! Készíts paraméteres konstruktort hozzájuk, mely az adattagokat várja paraméterül.
Előadásra nem kötelező járni, ezért csak a hallgató szorgosságán múlik, részt vesz-e rajta. Ezért az előadás esetében az alábbi módon valósítsd meg a **kurzusTartas** metódust: végig kell vizsgálni az egyes hallgatókat. Amennyiben egy hallgató egyik kedvenc tantárgyáról van szó, részt vesz a kurzuson (marad a listában). Ellenkező esetben a hallgató **szorgosságának** és a kurzust tartó oktató **előadóképességének** az átlagát kell venned. Generálj egy véletlen számot 1-10 között. Amennyiben ez nagyobb-egyenlő mint a kapott átlag tízszerese, a hallgató marad a kurzuson. Ellenkező esetben ki kell törölnöd a listából a hallgatót. Amennyiben a művelet végén a hallgatók létszáma kevesebb, mint a kurzust felvett hallgatók számának 10%-a, dobj **ErdektelensegbeFulladt** típusú kivételt.

Gyakorlatokra kötelező járni, ezért a metódus kicsit másképp működik. Ha a hallgató egyik kedvenc tantárgyáról van szó, továbbra is részt vesz a kurzuson. Ellenkező esetben, a gyakorlat kötelezősége miatt, a hallgató szorgossága minimum 0.6 lesz (ha kevesebb a szorgossága, ennyivel kell számolnod). Itt is vedd a **szorgosság** és az **előadóképesség** átlagát. Generálj egy véletlen számot 1-10 között. Amennyiben ez nagyobb-egyenlő mint a kapott átlag tízszerese, a hallgató marad a kurzuson. Ellenkező esetben ki kell törölnöd a listából a hallgatót.
Ha a bent maradt hallgatók száma kevesebb, mint a kurzust felvett hallgatók 20%-a, dobj **ErdektelensegbeFulladt** típusú kivételt. Ellenkező esetben vizsgáld meg a bent maradt hallgatók valódi szorgosságát. Menj végig az egyes hallgatókon, és ha a **szorgosságuk** valamint az oktató **előadóképességének** átlaga kisebb, mint 0.5, az adott hallgató nem fog figyelni a gyakorlaton. Amennyiben a bent maradt hallgatók legalább fele nem figyel, szintén dobj **ErdektelensegbeFulladt** típusú kivételt. **(8 pont)**

4. Hozz létre egy futtatható osztályt! Olvasd be a “hallgatok.csv” fájlból az egyes hallgatók adatait. A fájlban lévő adatok pontosvesszővel vannak elválasztva, pl:

```
Kiss Eugén;KIEXXXX.SZE;Kalkulus 1;Programozás 1
```

A sorok végén bármennyi kedvenc tantárgy szerepelhet. Hozz létre hallgatókat a fájl sorai alapján, és tárold őket egy listában.
Hozz létre egy tetszőleges gyakorlat (20 fős) és előadás (300 fős) objektumot. Add át nekik a hallgatók listáját, és hívd meg mindkettő **kurzusTartas** metódusát! **(6 pont)**

Figyelj a kivételkezelésre! A véletlen szám generálásához használhatod a java.util.Random osztályt, és annak nextInt metódusát! Az összes osztályt helyezzük a kurzusok csomagba!

Programozás I. - 2. ZH, 2015.05.04	**Név: .....**
B. csoport	**EHA: .....**
25 pont	**Gyakorlatvezető: .....**

Az ősi japán legenda szerint a rókatündér egy elátkozott lény, akibe ha beleszeret egy férfi, az elhalálozik. De az átok megtörhető, ha a rókatündér és választottja között a szerelem igaz.

1. Készíts egy **Ember** interfészt, melynek 1 metódusa van: **igazSzerelme()**, amely egy **Ember** típusú paramétert vár, és logikai értékkel tér vissza. Hozd létre a **NemIgazSzerelemException** kivétel osztályt, amely az *Exception* osztályból származik, és van egy paraméteres konstruktora (Stringet vár). ***(4 pont)***

2. Készítsd el a **Férfi** osztályt, amely implementálja az Ember interfészt. Egy férfiról tároljuk a **nevét**, illetve a "**szerelemkódját**", amely egy pozitív egész szám. Ezekhez legyenek getter/setter metódusok. Továbbá legyen megvalósítva az **igazSzerelme()** metódus, az alábbiak szerint: amennyiben a paraméterben kapott Ember objektumnak ugyanaz a "szerelemkódja", mint a sajátja, akkor térjen vissza igazzal, különben hamissal. Valósítsd meg a **toString()** metódust is!

Készítsd el a **Rókatündér** osztályt, amely szintén implementálja az **Ember** interfészt. Egy rókatündérről tároljuk el a "**szerelemkódját**", **áldozatai számát** (kezdetben 0), valamint hogy **elkeseredett-e** (logikai). Az **igazSzerelme** függvény adjon vissza igaz értéket, amennyiben a rókatündér elkeseredett, vagy pedig ha a szerelemkódja megegyezik a paraméterben kapott Ember szerelemkódjával. Legyen egy metódusa, amely eggyel növeli az áldozatok számát (**aldozatNovel()**), valamint, ha az áldozatok száma meghaladja a 10-et, állítsa igazra az elkeseredettséget jelző változót. Készíts egy paraméter nélküli konstruktort, továbbá az osztály legyen String-gé alakítható, valamint készíts getter metódust a szerelemkódhoz. **(7 pont)**

3. Keltsd életre az ősi japán legendát. Készíts egy **Legenda** osztályt, amelyben tárolj egy **Rókatündér** objektumot, valamint egy listát, amelyben az udvarló férfiakat tárolod. Ezeket az adatokat a konstruktorból inicializáld. Ezeken kívül legyen még egy **igazSzerelem** nevet viselő *Férfi* objektum, amelyet null-ra inicializálj! Készíts egy **tortenet()** metódust, amely végighalad az udvarlók listáján, és **70%** eséllyel meghívja a **udvarol()** metódust, amelynek egy paramétere van, a férfi, aki udvarolni próbál a rókatündérnek. Ez a metódus ***NemIgazSzerelemException-t*** dobhat! Az udvarol metódus vizsgálja meg, hogy a rókatündérnek az udvarló igaz szerelme-e, illetve, hogy az udvarlónak igaz szerelme-e a rókatündér. Amennyiben mindkét állítás teljesül, úgy állítsd be a jelenlegi férfira az igazSzerelme változó, és lépj ki a függvényből. Amennyiben az udvarlónak nem igaz szerelme a rókatündér, dobj egy ***NemIgazSzerelemException*** kivételt. A kivételt kezelő kódrészben töröld az adott Férfi objektumot a listából, valamint hívd meg a rókatündér ***aldozatNovel()*** metódusát is. Miután vége a ciklusnak, írd ki a `"tortenet.txt"` fájlba a rókatündér igaz szerelmét, amennyiben van igaz szerelme. Amennyiben nincs, írd ki, hogy `"A rókatündér nem talált igaz szerelemre."` **(8 pont)**

4. Hozz létre egy main metódust a ***Legenda*** osztályban, amely az "udvarlok.txt" fájlból olvassa be az udvarlók adatait egy listába. Az udvarlók kettőspont karakterrel vannak elválasztva, pl.:

```
Kiss Eugén:2847278
Ady Endre:202020
M. Tóth Károly:101516
```

Ezen kívül hozz létre egy ***Rókatündér*** objektumot is, majd keltsd életre a legendát *(egy Legenda objektum létrehozásával, amelynek paraméterben átadod a rókatündért, valamint az udvarlókat tartalmazó listát)* a tortenet() metódus meghívásával. **(6 pont)**

Figyelj a kivételkezelésre! A véletlen szám generálásához használhatod a java.util.Random osztályt, és annak nextInt metódusát! Az összes osztályt helyezzük a japanlegendak.rokatunder csomagba!

Programozás I. - 2. ZH, 2015.05.04	**Név: .....**
C. csoport	**EHA: .....**
25 pont	**Gyakorlatvezető: .....**

1. Készíts egy **Jatek** osztályt! Tárold egy a játék **nevét**, **nehézségi szintjét** (0-10 közötti egész szám, beállításkor ezt ellenőrizd le. Ha 0-nál kisebb értéket kap, 0 legyen, ha 10-nél nagyobbat, 10 legyen) Továbbá tárold le azt, hogy a játék **milyen típusú eszközön játszható** (szövegként)! Hozz létre az adattagoknak megfelelően egy paraméteres konstruktort! Az adattagok **privát** láthatóságúak legyenek, getter/setter metódusokkal.

Hozz létre egy **EltortKontroller** és egy **NemTamogatottJatek** nevű kivétel osztályt, melyek rendelkezzenek egy default és egy paraméteres konstruktorral is! ***(4 pont)***

2. Hozz létre egy **Konzol** nevű, **absztrakt** osztályt! Tárold le a konzol **nevét**, valamint legyen egy absztrakt **jatek** metódusa, mely egy **Jatek,** valamint egy **egész szám** (**skill**) típusú paramétert vár, **EltortKontroller és nemTamogatottJatek** típusú kivételeket dobhat. Az adattag privát legyen, megfelelő publikus metódusok segítségével azonban legyen lekérdezhető és módosítható! Készíts egy, az adattagnak megfelelő paraméteres konstruktort is!

Készíts egy **Jatekos** nevű osztályt is! Tárold el a játékos **nevét**, **skilljét** (0-10 közötti egész szám, beállításkor ezt is ellenőrizd le). Tárolj továbbá egy darab **Konzolt** és egy **Jatek** listát is a játékosnál! Készíts megfelelő paraméteres konstruktort, valamint getter/setter metódusokat.
Legyen egy **jatszik** metódusa, amely végigmegy a játékos által birtokolt játékokon és mindegyikkel megpróbál játszani (mindegyikre meghívja a birtokolt **Konzol jatek metódusát**). Amennyiben sikeres a játék (nem dobott kivételt), a játékos **skill**-je növekedjen eggyel. Ellenkező esetben írd ki az alapértelmezett hibakimenetre, hogy melyik játékos mely játékot nem tudta végigjátszani. **(7 pont)**

3. Származtass a Konzol osztályból egy **XBOX** és egy **Nintendo** osztályt is! Mindkettő egy **default konstruktorral** rendelkezzen, melyek beállítják a konzol nevét ("XBOX One" vagy "Nintendo Wii")!
XBOX esetében a **jatek** metódus megvalósítása a következő. Megkapja paraméterben a **játékot**, valamint a **játékos skilljét**. Azonban csak azokkal a játékokkal lehet játszani, melyek az adott konzolhoz valóak. Ha egy játék nem ehhez a konzolhoz való, dobj **NemTamogatottJatek** típusú kivételt! Ellenkező esetben vizsgáld le, hogy a játékos skillje hogy alakul az aktuális játék nehézségéhez. XBOX esetén "hardcore" játékosokról van szó, olyan játékot képes végigjátszani melynek nehézsége maximum 2-vel nehezebb, mint a játékos aktuális skillje. Amennyiben nem tudja végigjátszani a játékot, dobj **EltortKontroller** típusú kivételt!

Nintendo esetében alkalmi játékosokról van szó. A **játszik** metódus az **XBOX-hoz hasonlóan** működjön, azonban a játékos csak olyan játékokat tud végigjátszani, melyek **nehézségi szintje** maximum a játékos **skilljével** egyezik meg. **(8 pont)**

4. Hozz létre egy futtatható osztályt! Olvasd be a "jatekok.csv" fájlból az egyes játékok adatait! A játékok adatai pontosvesszővel vannak elválasztva, pl:

```
Grand Theft Auto V;6;XBOX One
Super Monkey Ball: Banana Blitz;2;Nintendo Wii
```

Hozz létre játékokat az egyes sorok alapján, és tárold őket egy listában.
Ezek után hozz létre két tetszőleges játékos objektumot, az egyiknek Nintendo, a másiknak XBOX konzolja legyen. Mindkettőnek hívd meg a **jatszik** metódusát! **(6 pont)**

Figyelj a kivételkezelésre! Az összes osztályt helyezzük a konzolok csomagba!

Programozás I. - 2. ZH, 2015.05.04	**Név: .....**
D. csoport	**EHA: .....**
25 pont	**Gyakorlatvezető: .....**

1. Írj egy interfészt **`KiralyiTanacsTag`** néven. Minden királyi tanács tagnak legalább két dolgot kell tudnia: tanácsot adni, és összeesküdni. Ezeket kösd ki az interfészben. A **`tanacsotAd`** metódus egy **`Tanacsado`** objektumot adjon vissza, és ne várjon paramétert. Az **`osszeeskuves`** metódus ne adjon vissza semmit, egy **`Tanacsado`** objektumot várjon paraméterül, és dobhasson **`TulBecsuletes`** kivételt. Hozd létre az említett saját **TulBecsuletes** kivételt egy új osztályban, ez mindig a "Az egyik tanácstag túl becsületes" üzenetet adja. ***(5 pont)***

2. Írd meg a **`Tanacsado`** osztályt, amely megvalósítja az imént megírt interfészt. Minden tanácsadó rendelkezzen a következőkkel: egy privát igaz/hamis **`alattomos`** érték, és egy privát **`osszeeskuvesek`** lista, ami **Tanacsado** objektumokat tárol, megmondja, kik ellen esküdött össze a tanácsadó. Legyen egy konstruktora is, amely az alattomosságot paramterben várja, a listát pedig egyszerűen üres listára inicializálja. Az **osszeeskuvesek** listának írj getter/setter metódust is. Valósítsd meg az interfész metódusait is a következőképpen: Az **osszeeskuves** metódus ellenőrizze le, hogy az aktuális tanácsadó alattomos-e. Ha nem az, akkor dobjon **TulBecsuletes** kivételt, ezt kapja el, és írja ki az üzenetét a default hibakimenetre. Ha alattomos, akkor adja hozzá az osszeeskuvesek a saját osszeeskuvesek listájához a paraméterül kapott másik tanácsadót.

   A **tanacsotAd** metódusban a tanácsadó 30% eséllyel úgy dönthet, hogy véletlenszerűen választ egy tanácsadót a saját **osszeeskuvesek** listájából, és átáll az ő oldalára. Tehát ilyenkor törli őt a saját listájából, és visszaadja őt eredményként (Ezzel feladva a többi tanácsadót, aki ugyanúgy összeesküdött ellene.). A maradék 70%-ban, vagy ha a listája eleve üres, akkor egy új, eddig nem létező **Tanacsado** objektumot ad vissza, amely **alattomosra** van beállítva. (Ilyenkor tehát csak kitalál valakit, aki nem is létezik.) ***(7 pont)***

3. Készíts egy **`Kiraly`** osztályt is, aki egy privát **`KiralyiTanacs`** listával rendelkezik, ebben is **Tanacsado** objektumok tárolódnak. Ennek a listának is legyen getter/settere.
   Legyen egy **`ules`** metódusa, ami nem vár paramétert, és nem is ad vissza semmit. Itt a király az összes tanácsadójától sorban tanácsot kér azok **tanacsotAd** metódusát meghívva.
   Minden ilyen tanács után a király ellenőrzi, hogy van-e az ebből visszakapott objektumnak megfelelő tanácstagja. Ha van, akkor az összes tanácstagnak végig kell nézni az **osszeeskuvesek** listáját, és akiké ezt a tanácstagot (tehát a tanács által visszaadottat) tartalmazza, azt rögtön kiteszi a tanácsából (azaz törli a király **KiralyiTanacs** listájából).

   Ezután, még szintén az **ules** metódusban a tanács összes maradék tagjára hívjuk meg az **osszeeskuves** metódust, egy másik, véletlenszerűen választott tagra a **KiralyiTanacs** listában maradtakból. Ha már csak egyetlen tagja van a tanácsnak, akkor ne is próbáljon összeesküdni. ***(8 pont)***

4. Írj egy futtatható osztályt is, amely a main függvényében hozzon létre két új királyt, legyen két új Tanácsadót tartalmazó listája, és olvasson be egy `input.txt` fájlt.
   Az input.txt felépítése a következő: az első sor tartalmazza, hogy összesen hány tanácsadót ad meg a fájl, az összes további sor pedig tartalmazza először is a tanácsadóhoz tartozó király sorszámát (1 vagy 2), ez fogja eldönteni, hogy melyik király tanácsába tartozik, utána egy szóközzel elválasztva pedig az `alattomos` vagy `nem alattomos` szavakat. Tehát ezek nincsenek idézőjelek között, pontosan így szerepelnek. Ez fogja eldönteni, hogy a tanácsadó alattomos-e. Olvasd be ezeket, és hozz létre ennek megfelelően új tanácsadókat, és add hozzá a megfelelő királyhoz tartozó listához.

A listát ne felejtsd el átadni a királynak, amikor elkészült. Hívd meg az első király ülés metódusát háromszor. ***(5 pont)***

Példa input.txt:

```
4
1 alattomos
2 nem alattomos
1 alattomos
1 nem alattomos
```