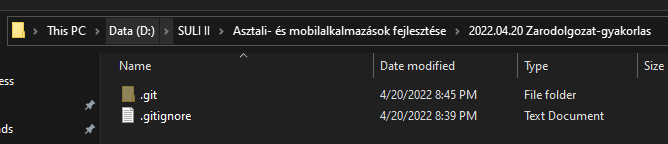
Vizsgafelkészítő

A vizsgán mind konzolos, mind grafikus felületen kell programot írni. Ez a dokumentum ezekre a feladattípusokra tartalmaz mintákat.

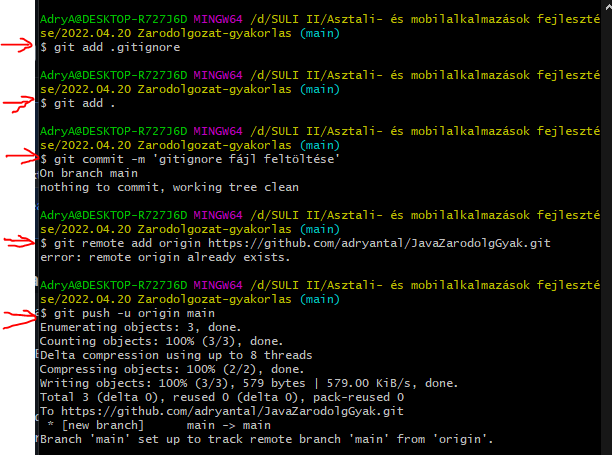
**A vizsgán jóval kevesebb feladat lesz, ebben most van minden!**

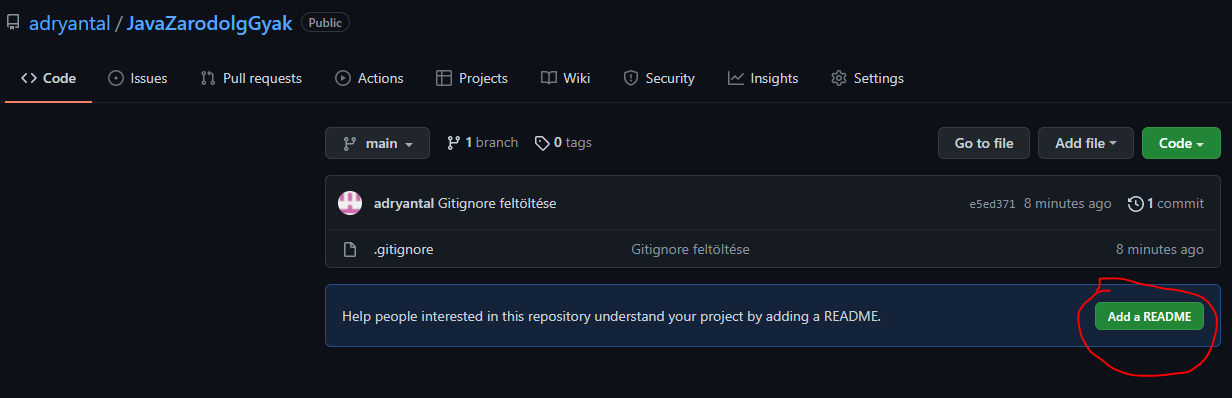
Mindegyik feladatot **GitHUB linken** kell bemutatni. A linkeket egyelőre nem kell feltölteni, de legyen meg! Projektenként több commit kell, **gitignore és readme fájllal**. Ez utóbbi tartalmazza a készítő nevét, akkor is, ha az kiderül a linkből.

1. Lokális mappa létrehozása
2. Git Bash elindítása 🡪 git init parancs
3. Gitignore.io 🡪 kulcsszavak: Java, Netbeans 🡪 Create 🡪 a megjelenő szöveg fájlba mentése .gitignore néven, és a lokális mappába mentem:

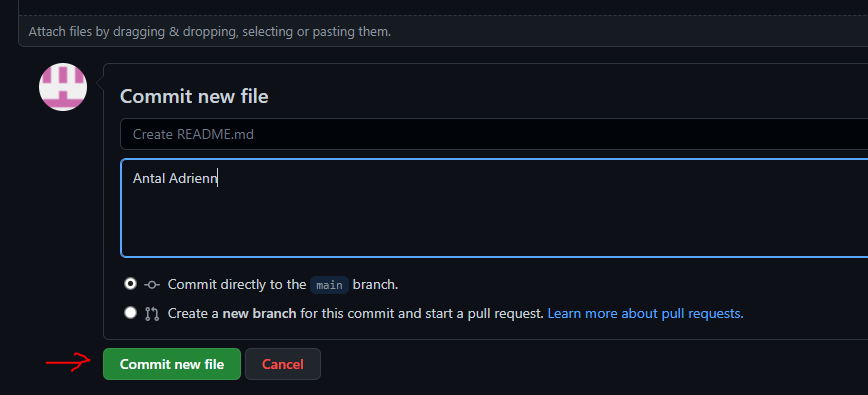


1. online repo létrehozása, majd:

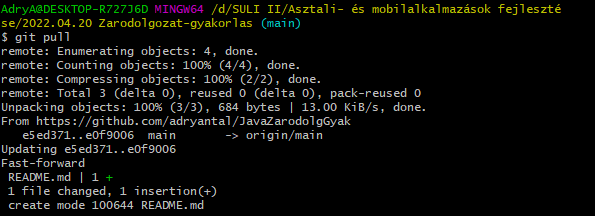




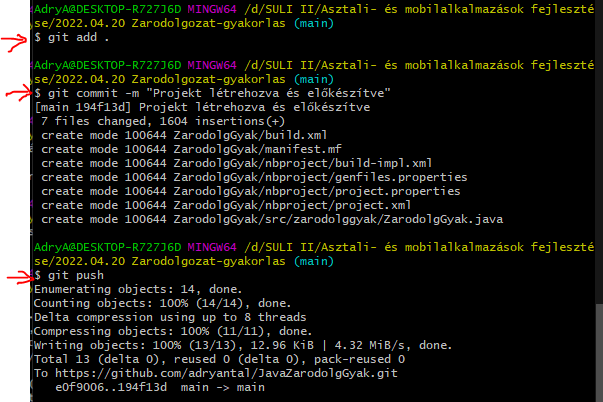
Legörgetek, majd (a Create README.md helyére tetszőleges commit nevet írok be):

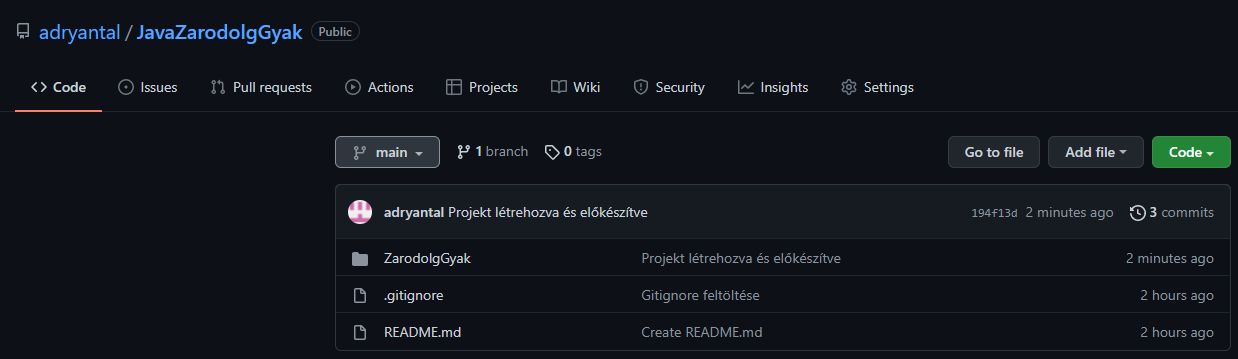


Leszedem a ReadMe fájlt, összesync-elem a lokális repóval:



Létrehozom a Java projektet, majd felcommittolom:





Gitignore <https://www.c-sharpcorner.com/article/steps-to-ignore-files-using-gitignore-in-details/>

A konzolos feladatban jelölöm a commit-ok helyét. Lehet több commit, de tartalmában kapcsolódjon az adott feladathoz. Ne haladj össze-vissza! A commit szövege legyen releváns, nem odaillő szövegnél nem jár pont erre a feladatrészre!

A GUI feladatoknál egyedi döntés, hogy ki, mikor commitol, de minimum 3 commit szükséges!

# **Konzolos feladat**

**1. commit, majd:**

Készíts egy Harcos osztályt, akinek ero és ugyesség adattagjai vannak. Az osztály terjessze ki az *absztrakt* Karakter osztályt, ahol nev és faj adattagokat tárolunk, inicializálunk. Minden karakternek vannak eszközei, amit az Eszkoz osztály ír le. Az eszközöket a nev és suly adattagok jellemeznek. Az eszközök súlya valós érték.

**2. commit, majd:**

A Karakter osztály dobjon *saját* kivételt, ha a megadott név 3 betűnél rövidebb! Tudjon felvenni eszközöket és eldobni. Eldobni egész index és túlterhelve Eszkoz típusú paraméterek alapján lehet.

Legyen egy olyan metódus, ami az eszközöket a nevük szerint rendezve adja vissza. Ezt használja a felülírt toString, plusz az összes többi adatot a kiírására. A stringek összefűzésére StringBuilder osztályt használjunk! A név szerint rendező metódus ne referenciát adjon vissza (pl jó a nem módosítható lista vagy egy másolatot)!

Lehessen iterálni azokon a példányokon, akiknek az osztály a statikus típusa *( Karakter k = new Harcos(..) 🡪 Karakter (’ős”): statikus, Harcos („utód”) dinamikus típus*). Azaz tudjuk foreach szerkezettel bejárni az osztályt, ami során az eszközöket adja vissza, egész pontosan az Eszkoz osztály legenerált toString metódusát*. (magyarul: a Karakter osztály legyen iterálható??)*

**3. commit, majd:**

A Harcos osztálynak legyen egy túlterhelt metódusa, amikor nem kell megadni a faj adattag értékét. Ebben az esetben az értéke legyen „ember”.

Az osztály példányait lehessen rendezni mind ero mind ugyesseg adattagjai alapján!

**4. commit, majd:**

A main metódusban hozz létre 4 Harcos példányt, ezek közül *1 dobjon kivételt*!

Mutasd be az egyik példányon a felvesz és eldob metódusokat, kiírással.

Szerializáld az egyik példányt, *majd töltsd is vissza* a kimentett állapotot. Kiírással mutasd be a működést. (*A feladat csak akkor ér pontot, ha a betöltés is megvalósult!)*

Mutasd be, hogy *egy példány iterálható*, azaz járd be egy foreach ciklussal!

Egy *tömbben* tárold el a 3 példányt, majd jelenítsd meg az *erő és az ügyesség alapján berendezett* *tömböt*

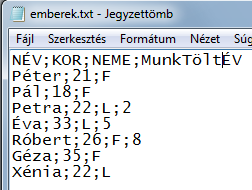
**5. commit**

# **Grafikus felület:**

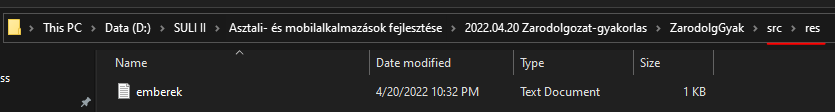
# Irányított GUI

## Specifikáció:

Egy szöveges állományt kell kezelni, ami az alábbi formátumú legyen:



*(a src/res mappába mentjük az alapfájlt)*



Mind a tartalom, mind a szerkezetnek ilyennek kell lennie! A tartalom csak a kiegészítő lehetőség (ld. köv. pont) hatására változhat!

## Kiegészítő lehetőség:

A programban legyen lehetőség új adatsort rögzíteni a fájlban, ahol a fájl nevét és helyét nem kell külön kezelni, ezek „bedrótozva” is lehetnek. A felület kinézete szabadon választott, de minden adatot tudjunk benne megadni, minél inkább törekedjünk arra, hogy ne lehessen elírni adatokat, pl. a nem esetén csak 1 betű, ’L’ vagy ’F’ kerüljön elfogadásra.

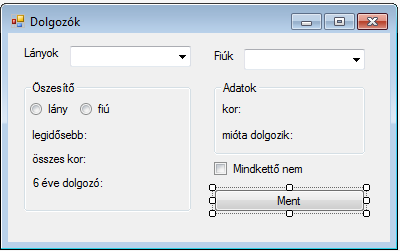
A program kezelje, ha nem található a fájl, de megléte esetén feltételezhetünk legalább 1 adatsort benne!

## A program fő funkciója (Ezt már kötelező elkészíteni):

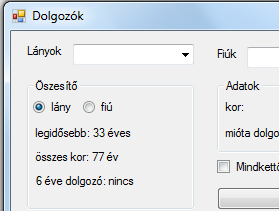
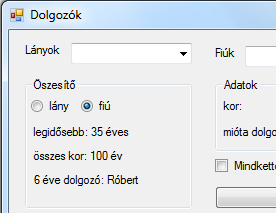
1. Készítsünk összesítést az alábbi kérdésekre és jelenítsük meg külön a lányok és külön a fiúk adataival
   1. hány éves a legidősebb?
   2. összesen hány évesek?
   3. ki az, aki legalább 6 éve dolgozik?
2. Válogassuk szét a fájlból a lányok és a fiúk nevét! Ezt jelenítsük meg 1-1 lenyíló menüben (*combobox*)
3. Ha a lenyíló menüben kiválasztunk 1 nevet, akkor a hozzá tartozó kort és azt, hogy mióta dolgozik, jelenítsük meg!
4. Legyen egy „Mentés” gomb, ami az 1. feladat eredményét fájlba írja!
5. Legyen egy választó négyzet (*checkbox*), ami alapján mind a kettő nem adatait kiírhatjuk a fájlba

**Az egyes feladatok és a hozzájuk tartozó eredmények a következő oldalon láthatóak. A felületek kinézete pontosan a megadott minta szerinti legyen!**

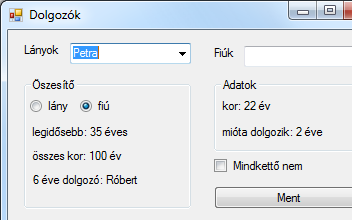
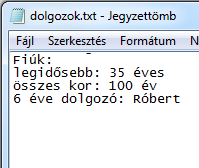
## Készítsd el –megfelelő elnevezéssel – az alábbi felületet:



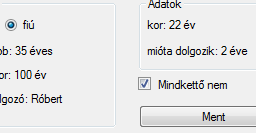
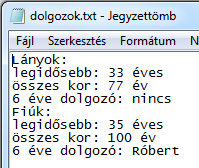
## A megfelelő választó gomb feltölti az „Összesítő” részt (1. feladat):

## A kiválasztott lenyíló feltölti az „Adatok” részt (2. és 3. feladat), illetve „Ment” gomb (4. feladat):

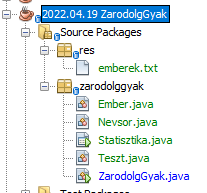
 

## Mindkettő nem adatainak kiírása (5. feladat):

Megoldás menete:

Mappastruktúra:



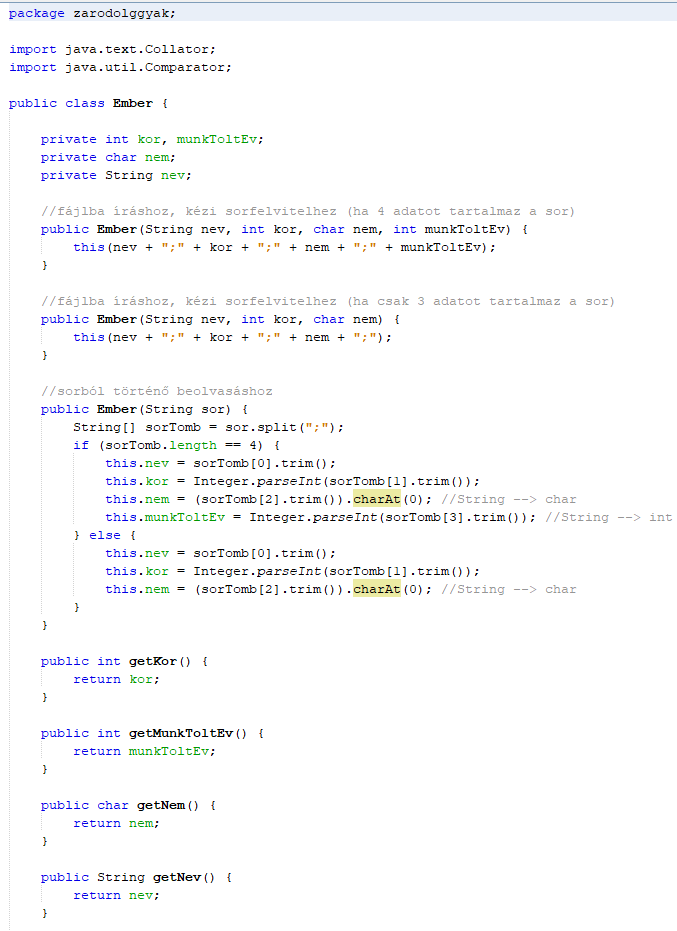
**Konzolos rész**:

Létrehozok egy ***Ember*** osztályt, 3féle konstruktorral

Létrehozok egy ***Nevsor*** osztályt, amelynek egyetlen adattagja egy Ember objektumok tárolására alkalmas lista. Az osztályhoz 2 konstruktor tartozik: egy „kézi” rögzítéshez igazított, valamint egy fájlból történő beolvasást végrehajtó konstruktor. A lista „gettere” másolatlistát ad vissza.

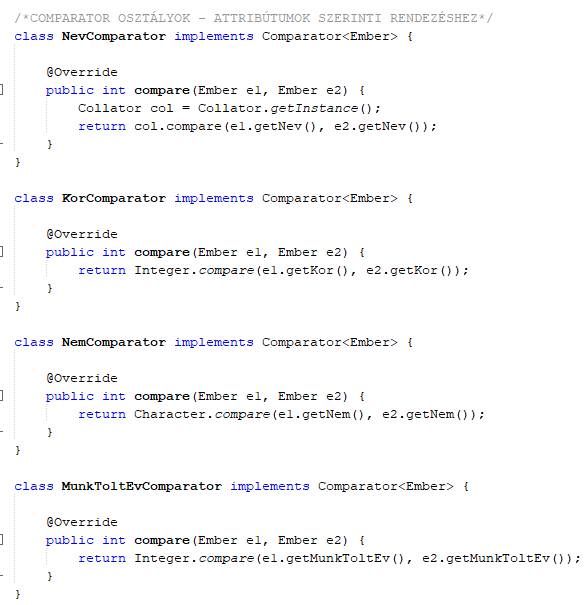
A ***Teszt*** osztály a műveletek tesztelésére lett megkreálva.

Az Ember osztály kódja:

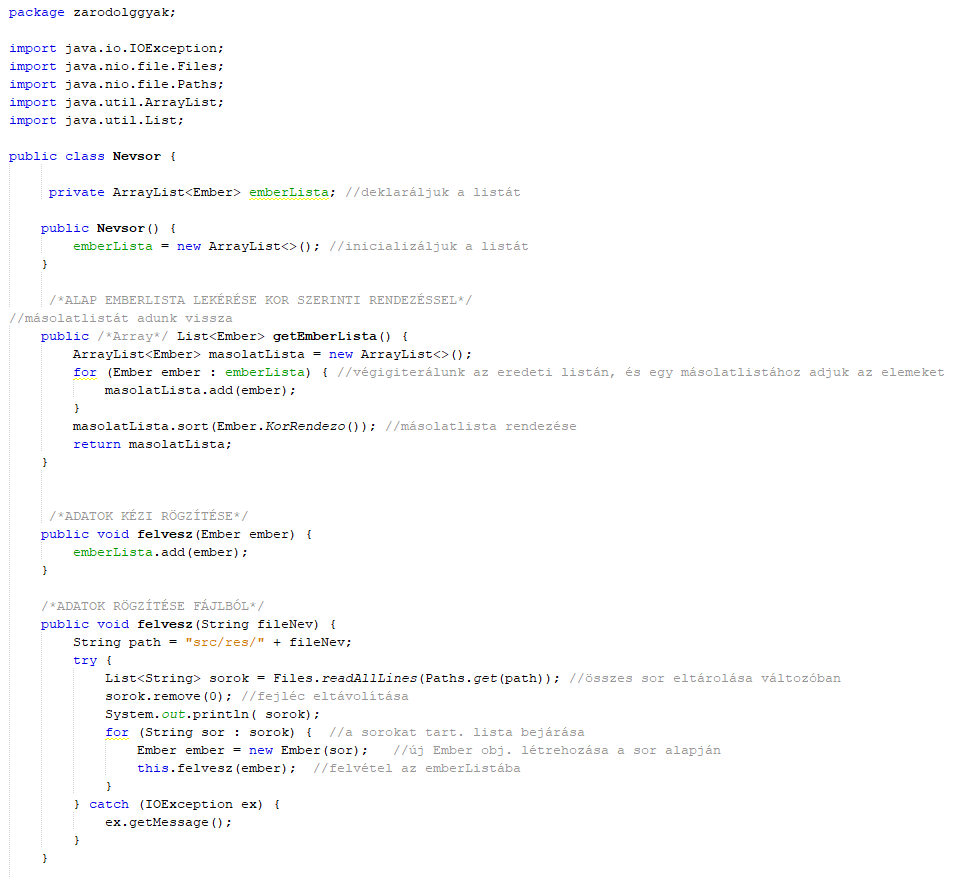


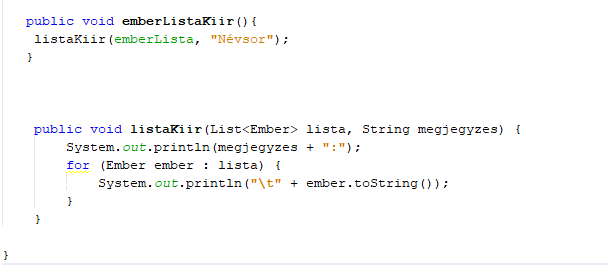


A kapcsolódó Comparator osztályok:



A Nevsor osztály kódja:

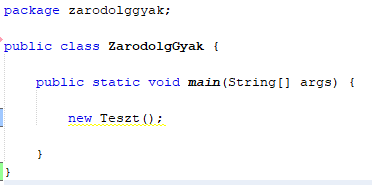




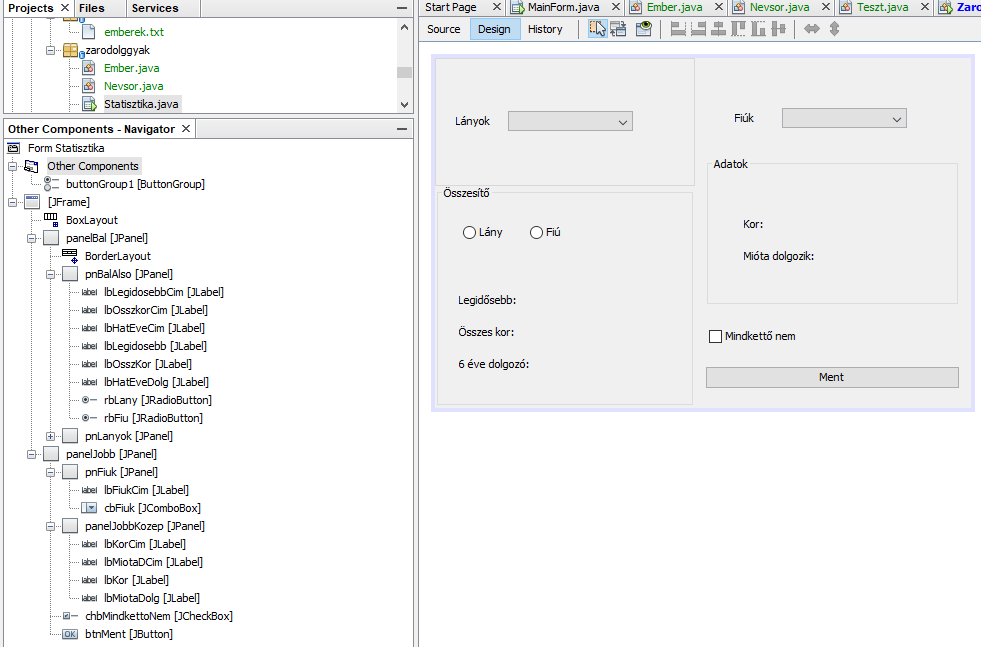
A Teszt osztály kódja:



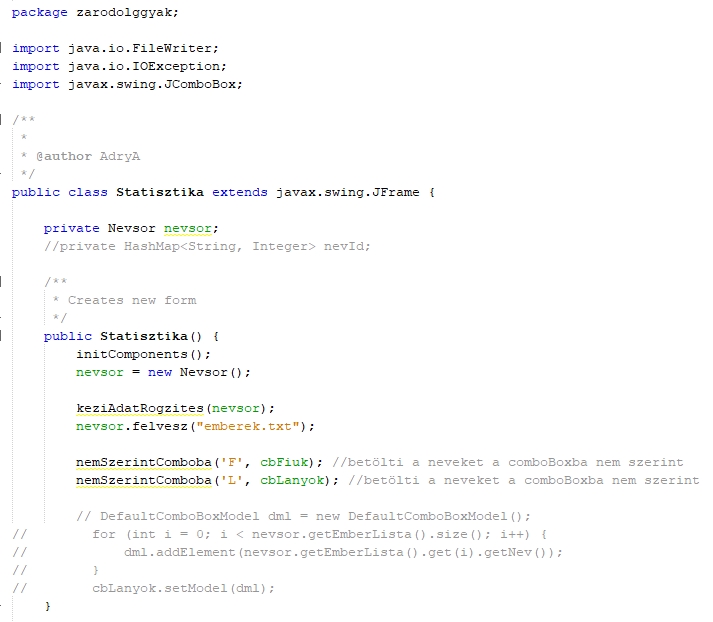
A ZarodolgGyak.java tartalmazza a főprogramot, amely a Teszt osztályt példányosítja:



**Grafikus rész**:

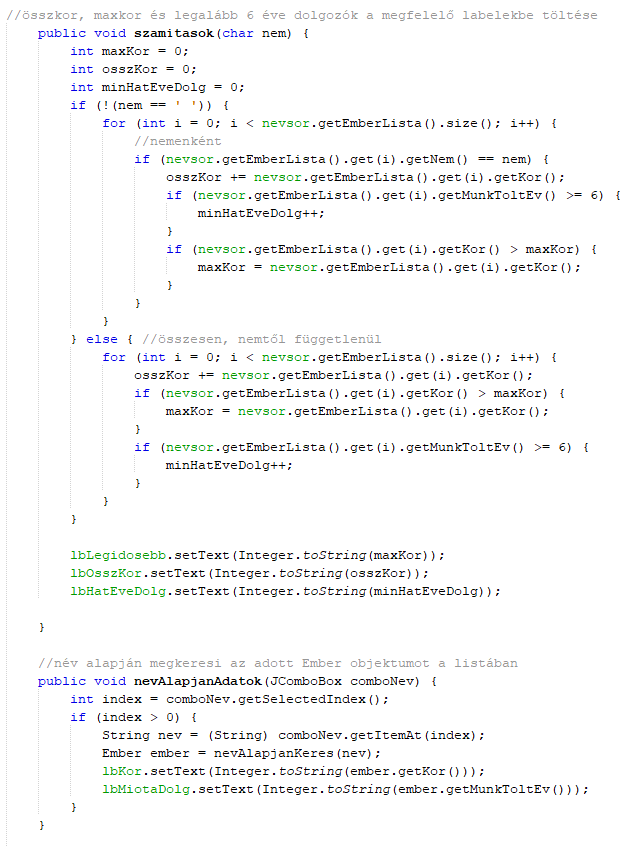


A lista felhasználásával végezzük el a különféle műveleteket (keresés, kiválogatás, stb.), és töltjük be az adatokat a megfelelő grafikus elemekbe, valamint írjuk ki fájlba:











# Egyénileg elkészített GUI és program

## Specifikáció:

Készíts szöveges állományokat használó, kérdés / válasz alapú programot. Legyen egy felületünk, ahol egy csomagot tudunk összeállítani, beírjuk a kérdést, meg a helyes választ, és elmentjük. Ezt többször ismételhetjük, így a kérdés / válasz lapokat, egy a felhasználó által elnevezett (fájl nevének és helyének a megválasztása után) csomagba (.txt fájl) mentünk.

Egy másik felületen tudjunk csomagot választani, amiben több -az előzőleg elmentett- kérdések és válaszok vannak. Sorban, a fájlban meghatározott sorrendben megjelenik a kérdés, majd egy gomb megnyomására megjelenik a válasz.

## Technikai kiegészítés

A programnak nem kell vizsgálnia a helyes választ nincs is bekérési lehetőség, csak, amikor készítjük a kártyákat. A programnak az a feladata, hogy a felhasználót segítse fogalmak megtanulásában és kikérdezésében. A felhasználó maga dönti el, hogy az általa adott válasz helyes volt e, a megjelenített jó válasz segítségével. Ebben nem segít a program.

Több csomag is készíthető, különböző témakörökben. Pl. matematika, amiben egyszerű számolásokat kérdezünk, meg irodalom, ahol költő és vers témákat dolgozunk fel. Persze lehet programozási kérdéseket is feltenni a feladat megoldása során.

Fontos, hogy a felhasználó maga választhassa meg a fájl, azaz a csomag nevét és helyét! Ezt az operációs rendszerben megszokott módon tudja megtenni!

### További kiegészítő lehetőségek:

A fájl nevének kiválasztásakor paraméterezzük a fájlválasztó ablakot, egyéni szűrők, fájlformátum, felülírási lehetőség vizsgálatával. Az is segítség, ha rögtön egy meghatározott helyet ajánlunk fel, pl a dokumentumok, vagy a projekt könyvtárát.

Segédanyagok - mit hol találok:

**Absztrakt osztály ill. öröklődés** 🡪 Bankkártyaprogram, 2021.12.16 Dolgozat, FaIdomokProgramUML

**Szerializáció, rendezések, fájlkezelés, bejárhatóság, másolhatóság, iterálhatóság** 🡪 utolsó dolgozat (HadseregProgram)

**Saját kivételosztály** ---> AK Akademia - feladatok -> kivetelkezeles57elso

**Stringbuilder használata** --> <https://stackoverflow.com/questions/1532461/stringbuilder-vs-string-concatenation-in-tostring-in-java>