Debreceni Egyetem
Informatikai Kar

# VIZSGÁZTATÁST SEGÍTŐ ALKALMAZÁS JAVA NYELVEN

Témavezető:
**Dr. Tornai Róbert**
egyetemi adjunktus

Készítette:
**Urbán Attila**
programtervező informatikus hallgató

Debrecen
2011.

# Tartalomjegyzék

# 1. Bevezetés

Mai, modern világunkban a multimédia szinte az élet minden területén jelen van. Előnyeit egyaránt élvezhetjük a munkában, otthon és természetesen az oktatásban is.

Ma már nyilvánvalónak tűnik, hogy a háztartásokban található televízió készülék, DVD lejátszó, egyre több helyen pedig számítógép is. Holott ezek az eszközök 50-60 évvel ezelőtt még csak néhány ember számára voltak elérhetőek. A rohamos fejlődés következtében nem csak az otthonokban, de a középiskolai és az egyetemi oktatásban is egyre jelentősebb szerepet kap a multimédia. Az oktatási intézményekben azonban az anyagi keretek szűkössége miatt, csak lassabb ütemben halad az elterjedése.

A multimédia egyik nagy előnye, hogy segítségével az oktatók táblázatokkal, grafikonokkal, képekkel, vagy akár videókkal is színesíthetik az előadásaikat. Így még jobban felkeltve a diákok érdeklődését egy-egy témakörrel kapcsolatban. Ráadásul több médium egyidejű használatával, a hallgatók az új ismereteket nagyobb eredményességgel tudják elsajátítatni [1].

A multimédia előnyei azonban nem csak a hagyományos oktatásban, de a távoktatásban is jól használhatóak. Az E-Learning segítségével a hallgatóknak lehetőségük van a világ bármely pontjáról a tananyagot rugalmasan, a saját időbeosztásuknak megfelelő ütemben elsajátítaniuk. A különböző tesztek, oktatási segédanyagok segítségével pedig egy valós képet kaphatnak arról, hogy az adott anyagrészen belül milyen hiányosságaik vannak, és egy jövőbeni sikeres vizsgához, milyen ismeretekre van még szükségük. Az E-Learning további előnye, hogy a hallgatók a problémáikat akár egymással, akár az oktatóval is meg tudják beszélni. Így akár néhány percen belül választ kaphatnak a kérdéseikre.

Bár a távoktatás esetében a hallgatók az interneten is kitölthetik a kurzus keretében összeállított teszteket, a tudásukról minden esetben az adott oktatási intézményben kell számot adniuk. Természetesen sokkal kényelmesebb lenne, ha a hallgatók magát a vizsgát is ki tudnák tölteni az interneten, a válaszaik beérkezése után pedig rögtön megismerhetnék az eredményüket. Ez azonban egyelőre nem megvalósítható, hiszen a legfejlettebb technikai találmányok alkalmazásával sem biztosítható, hogy a hallgatók nem használnak fel valamilyen segédeszközt a válaszaik megadásánál.

De nem csak a távoktatás esetében, a hagyományos képzésnél is több probléma felmerül. Egyetemi tanulmányaim során több olyan kollokviumon is részt vettem, amelyen egy feleletválasztós tesztet kellett kitölteni. Ennek a kiértékelése 100-150 ember esetében meglehetősen sok időt vett igénybe. Ezt akár radikálisan is le lehetne csökkenteni, ha lehetőség lenne a teszt számítógépes kitöltésére. Hiszen ilyenkor az oktatónak csak egy alkalommal kellene átnéznie a kérdéseket, illetve az arra adható válaszokat. Igaz, ebben az esetben a hallgatók böngészőt, illetve különböző alkalmazásokat is futtathatnának, amelyek nagyon nehézzé tennék a vizsga felügyeletét.

Ennek a problémának a hatására kezdtem el egy olyan megoldást keresni, amely segítségével a hallgatók minimális felügyelet mellett is meg tudnák írni a vizsgát. Habár a kipróbált alkalmazások többségében volt lehetőség egyéni kérdéssor összeállítására, kitöltésére, valamint a végén a válaszok ellenőrzésére, egyik sem biztosított megfelelő védelmet a különböző csalásokkal szemben. Ezért tűztem ki a szakdolgozatom céljául egy olyan alkalmazás elkészítését, amely az alapvető funkciókon túl, lehetőséget biztosít az elindított programok koordinálására is.

# 2. Java

*„Write once, run anywhere.”*
*James Gosling*

A Java egy majdnem tiszta objektumorientált nyelv, amely a C++ továbbfejlesztésével alakult ki. Létrehozásakor a fejlesztők kihagyták belőle azokat a nyelvi elemeket, amelyek a hibalehetőségeik miatt megnehezítették a programozók munkáját. Ilyenek a mutatók, a struktúrák, illetve a dinamikus memóriahasználat.

## 2.1. Kialakulása

Története egészen 1991-ig nyúlik vissza, amikor a Sun Microsystems „Green Team” csoportja, James Gosling vezetésével egy olyan nyelvet kezdett el fejleszteni, amely lehetővé teszi a különböző háztartási eszközök közötti kommunikációt. A nyelvvel szemben alapvető elvárás volt, hogy platform független legyen. Ennek érdekében a nyelv egy virtuális számítógépre közbenső, úgynevezett bájtkódot generál, amely a JVM (Java Virtual Machine) segítségével minden olyan operációs rendszeren futtatható lesz, amely rendelkezik a megfelelő interpreterrel.

A Java a 90-es évek közepén óriási ütemben kezdett terjedni. Ez részben annak volt köszönhető, hogy a böngészőkbe beépülő JVM életre tudta kelteni az addig statikus lapokat. Az appletek segítségével olyan alkalmazások futtatása is lehető vált, amelyekkel animációkat, képeket, illetve videókat is meg lehetett jeleníteni. Kezdetben csak a HotJava Browser támogatta ezt a funkciót. Néhány évvel később viszont bekerült a Netscapbe is. Manapság pedig már az összes elterjedt böngésző támogatja a használatukat.

Az idők során a Java a világ legszélesebb körben használt programozási nyelve lett. Világszerte több mint 9 millió fejlesztő használja a napi munkája során [2]. Kiterjedt osztálykönyvtárának hála, nem létezik olyan probléma, amelyet ne lehetne megoldani a segítségével.

## 2.2. Fejlődése

- **JDK 1.0 (1996. január 23):**
  - kezdeti verzió
  - első stabil változata a JDK 1.0.2

- **JDK 1.1 (1997. február 19.):**
  - belső osztály
  - JavaBeans
  - JDBC: Java Database Connectivity
  - reflection
  - RMI: Remote Method Invocation

- **J2SE 1.2 (1998. december 8.):**
  - Java technológia három, független részre bontása:
    - J2SE: Java 2 Platform, Standard Edition
    - J2EE: Java 2 Platform, Enterprise Edition
    - J2ME: Java 2 Platform, Micro Edition
  - IDL: Interface Definition Language
  - kollekciók: List, Map, Queue, Set
  - strictfp: lebegőpontos műveletek korlátozása (hordozhatóság érdekében)

- **J2SE 1.3 (2000. május 8.):**
  - HotSpot
  - JavaSound
  - JNDI: Java Naming and Directory Interface
  - JPDA: Java Platform Debugger Architecture

- **J2SE 1.4 (2002. február 6.):**
  - assert
  - biztonsági és kriptográfiai fejlesztések
  - IPv6 támogatása
  - Java Web Start
  - XML elemző és XSLT feldolgozó

- **J2SE 5.0 (2004. szeptember 30.):**
    - automatikus be- és kicsomagolás
    - felsorolásos típus
    - foreach ciklus
    - generikus típus
    - statikus import
    - változó paraméterszámú függvények

- **J2SE 6.0 (2006. december 11.)**
    - JDBC 4.0
    - JVM teljesítményének optimalizálása
    - különböző GUI fejlesztések
    - régebbi Windows 9x verziók támogatásának megszüntetése

- **J2SE 7.0 (2011. július 28. ?)**
    - generikus példányosításánál a jobb oldalon elegendő a <> használata
    - gyorsabb fájlműveletek a java.nio csomaggal
    - hatékonyabb szemétgyűjtögető (G1)
    - string használatának lehetősége a switch utasítás szintaktikájában
    - tömörített 64 bites mutatók

# 3. Moodle

A Moodle szó a Modular Object-Oriented Dynamic Learning Environment kifejezés rövidítése, azaz moduláris objektum-orientált dinamikus tanulási környezetet jelent.

## 3.1. Kialakulása

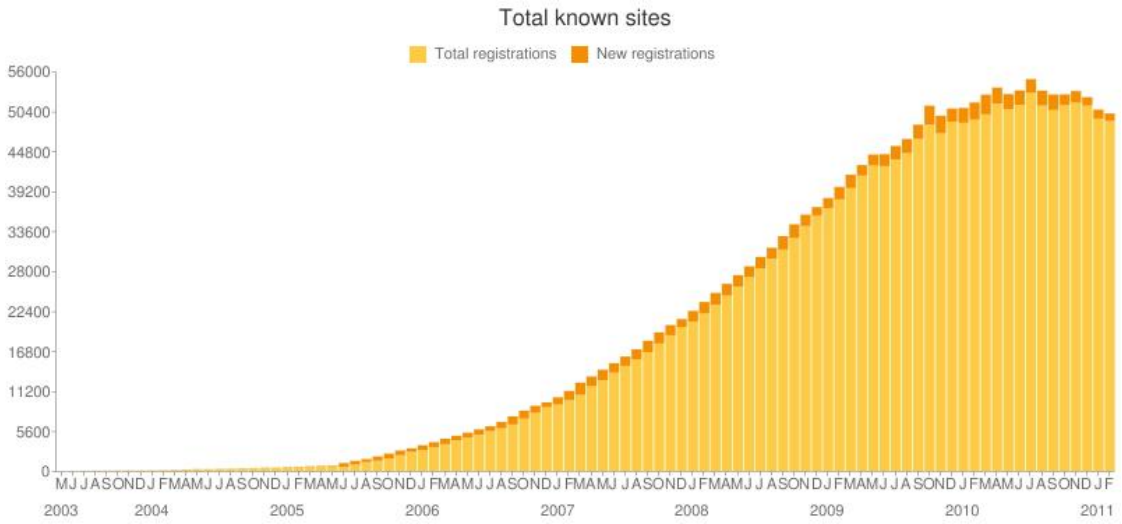
A keretrendszer első verziója 2002. augusztus 20-án jelent meg, Martin Dougiamas, ausztráliai fejlesztő munkájának eredményeként. Tervezése során elsődleges szempont volt a konstruktivista pedagógia alapelveinek megvalósítása. Hibásnak érezte azt a felfogást, hogy a tananyag kialakítása kizárólag az oktató feladata legyen. Lehetővé kívánta tenni a hallgatók és az oktatók közötti folyamatos kommunikációt. Ennek érdekében elérhető egy fórum modul, ahol a hallgatók nem csak egymással, de az oktatóval is meg tudják beszélni a felmerülő problémákat.

A Moodle azonban számos más, hasznos eszközt is biztosít a felhasználók számára. Az oktatóknak például lehetőségük van a hallgatók munkájának állandó nyomon követésére, illetve értékelésére. Egy kurzus végén így nem csupán a megírt esszé, vagy a kitöltött teszt eredményére támaszkodhatnak, hanem sokkal részletesebb képet kaphatnak az egyes hallgatók teljesítményéről.

Részben tudásának, részben nyílt forráskódjának köszönhetően, a Moodle hatalmas fejlődésen ment keresztül az elmúlt években. 2010. november 24-én jelent meg a 2.0-s verzió, amelyben szinte a teljes kódot újraírták, illetve optimalizálták. A három legaktívabb, önkéntes fejlesztő az új verzió sikeréhez közel 1.4 millió kódsorral járult hozzá [3]. Néhány jelentős újdonság [4]:

- beépített WYSIWYG szerkesztő
- feltételes tevékenységek
- integrált plágiumkereső
- megújult wiki
- továbbfejlesztett blog motor
- új navigáció és design
- új szerepkör: manager

A Moodle-t jelenleg 211 ország, több mint 50000 különböző intézményében (1. ábra), illetve oktatási portálján használják [5]. A regisztrált felhasználók száma meghaladja a 41 milliót.




*1. ábra: Regisztrált portálok*

*Forrás: http://moodle.org/stats/index.php*

## 3.2. Szerepkörök

A Moodle egy LMS (Learning Management System) alkalmazás. Feladata, hogy azonosítsa a felhasználókat, majd a jogosultságuknak megfelelő tananyagokkal, kurzusokkal rendelje össze őket. A Moodle-ben hét különböző szerepkör létezik [6]:

- **adminisztrátor-rendszergazda**: A legmagasabb rangú felhasználó. Ő felel a szerver karbantartásáért, valamint a keretrendszer szabályos működéséért.

- **kurzuskészítő**: A legmagasabb rangú oktató. A kurzuskészítők rendelkeznek kurzusdefiniálási joggal. Minden általuk létrehozott kurzust látnak, és képesek azokat szerkeszteni. Lehetőségük van oktatókat hozzárendelni a kurzushoz, illetve meghatározni a sorrendjüket. Továbbá az oktatóktól megvonhatják a szerkesztési jogukat, illetve el is távolíthatják őket.

- **menedzser**: Hozzáférhet a különböző kurzusokhoz, illetve módosíthatja is őket. Általában nem vesz részt a kurzusokon.

- **oktató**: Oktató bárki lehet, akit egy adminisztrátor vagy kurzuskészítő hozzárendel egy kurzushoz. Akár egy hallgató is. Az oktatók egy kurzus menedzselését, karbantartását végzik. Tananyagokat tölthetnek fel, szabályozhatják azok hozzáférhetőségét, illetve különböző feladatokat adhatnak a hallgatók számára.

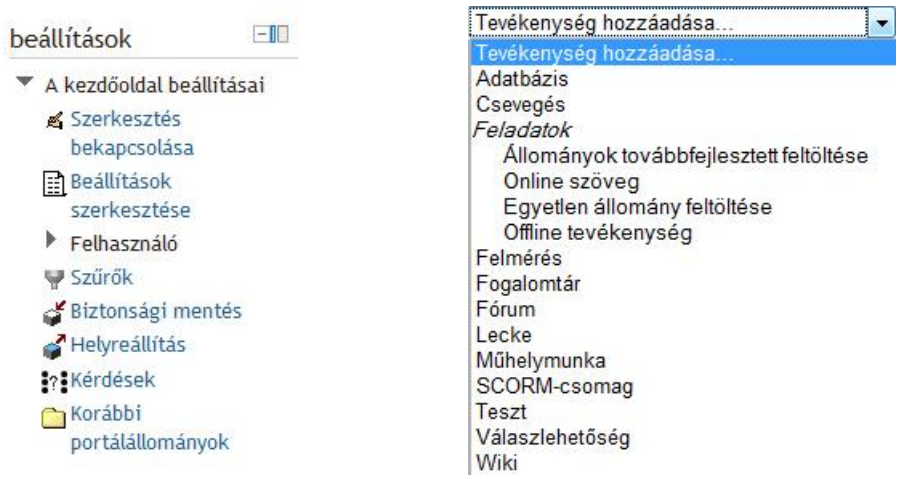
- **nem szerkesztő oktató**: Olyan oktató, aki nem rendelkezik szerkesztési joggal a kurzusban. Nem tudnak tananyagot feltölteni, illetve újabb feladatot kiírni a hallgatók számára. Viszont értékelhetik a hallgatók tevékenységeit, feladatait, illetve megtekinthetik az osztályzataikat is.

- **hallgató**: Olyan személy, aki hallgatói tevékenységet folytat az intézményben. A hallgatók hozzáférhetnek a számukra biztosított tananyagokhoz, illetve teljesíthetik az oktatók által feltöltött feladatokat is.

- **vendég**: Olyan személy, aki nincs regisztrálva a rendszerben. Vendégként lehetősége lehet bejelentkezni, illetve tanulmányozni az egyes funkciókat. A vendégek bejelentkezéséről az adminisztrátor vagy a kurzuskészítő dönthet.

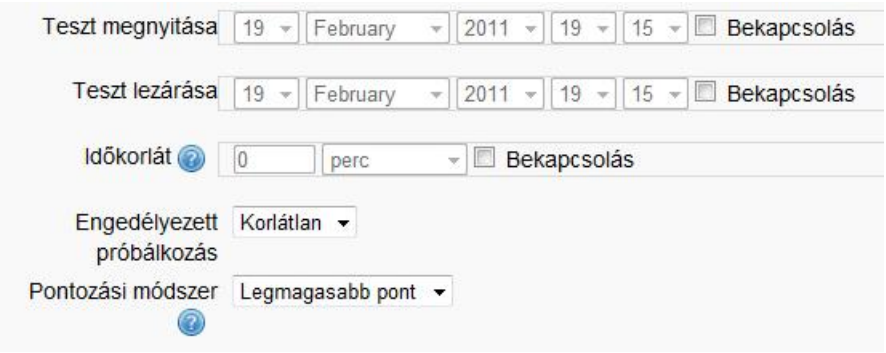
## 3.3. Teszt létrehozása

Oktatóként történő bejelentkezés után, lehetőség van teszt létrehozására (2. ábra). Ehhez a Beállításokon belül, a Szerkesztés bekapcsolása szükséges. Majd a Tevékenység hozzáadásánál ki kell választani a Teszt opciót.




*2. ábra: Teszt létrehozása*

Egy teszt elkészítésénél számtalan beállítási lehetőség választható (3. ábra). Meg lehet adni a megnyitás és lezárás időpontját, valamint egy időkorlátot is, a teszt kitöltéséhez. Lehetőség van továbbá a kérdések összekeverésére, illetve jelszó létrehozására. Így a hallgatók csak ennek birtokában tudnak majd hozzáférni a teszthez. Nem utolsó sorban pedig a próbálkozások száma is korlátozható.




*3. ábra: Beállítási lehetőségek*

A teszt létrehozása után, különböző típusú kérdések adhatóak meg (4. ábra). Az egyik leggyakrabban használt típus a feleletválasztós kérdés. Ezt a hallgatók is kedvelik, hiszen lehetőséget biztosít a tippelésre. Éppen ezért az oktatók sokszor több helyes választ is megadnak, illetve a hibás válaszoknál pontlevonást alkalmaznak. Szintén gyakran használt típus a kiegészítendő kérdés. Ilyenkor a hallgatóknak egy jól definiált fogalom nevét kell megadniuk. Illetve lehetőség van esszé létrehozására is.


*4. ábra: Kérdéstípusok*

# 4. Fejlesztés

Bár a Moodle számtalan lehetőséget biztosít a hallgatók tudásának ellenőrzésére, a vizsga során a tanulók tetszőleges programokhoz hozzáférhetnek. Így a számonkérés során lehetőségük van például böngészőprogram, csevegőprogram, vagy bármilyen más alkalmazás elindítására is, amivel könnyen válaszolni tudnak a feltett kérdésekre. Még abban az esetben is, ha a vizsga intézményen belül, több oktató felügyelete mellett zajlik, hiszen az alkalmazások többségét észrevétlenül, néhány másodpercen belül el lehet indítani, illetve be lehet zárni.

A Moodle másik hátrányát a bonyolult felhasználói felületben látom. Bár a legtöbb oldalon elérhető a súgó, amely segítséget nyújt az egyes funkciók használatával kapcsolatban, sokszor ezek elolvasása után sem egyértelmű egy-egy beállítási lehetőség helyes alkalmazása. Ez pedig sokakat elriaszthat a használatától.

Ezen problémákból kiindulva, a célom egy olyan alkalmazás elkészítése, amely az alapvető funkciókon túl, lehetőséget biztosít az elindított programok ellenőrzésére, valamint könnyen áttekinthető funkciókkal rendelkezik. Így az alkalmazás használata azon oktatók, illetve hallgatók számára is átlátható és egyértelmű lesz, akik csupán első alkalommal találkoznak vele.

Az alkalmazást egy felhasználónév és jelszó segítségével lehet majd elérni. Mind hallgatóként, mind oktatóként lehetőség lesz a bejelentkezésre. Mindkét szerepkör számára lehetővé kívánom tenni, hogy megtekinthessék a korábbi vizsgák eredményeit, illetve az aktuális vizsgák listáját. Az oktatók továbbá létre tudnak majd hozni új vizsgákat. Ezek lehetnek esszé, illetve teszt jellegűek, de lehetőségük lesz a kettő kombinálására is. Ebben az esetben a hallgatók csak akkor férhetnek hozzá az esszé kérdéseihez, ha a teszt végén jobb eredményt értek el, mint az oktató által megadott minimum. A tesztek tetszőleges számú kérdésekből állhatnak, a kérdésekre pedig akár több helyes válasz is megjelölhető lesz. Az oktatók számára lehetőség nyílik továbbá a vizsgán résztvevő hallgatók kiválasztására, a vizsgák szerkesztésére, illetve azok törlésére is. Valamint az egyes eredményeket is el tudják majd távolítani.

Az alkalmazás elkészítése során azon programok számára, amelyek futtatása mindenképp szükséges, létre kívánom hozni az engedélyezett programok listáját. Tipikusan ilyenek a különböző vírusirtók, tűzfalak, illetve Windows szolgáltatások. Az alkalmazás mind a vizsga előtt, mind a vizsga közben, folyamatosan figyelni fogja az elindított programokat, a nem engedélyezetteket pedig azonnal bezárja.

A vizsga során továbbá figyelemmel fogom kísérni azt is, hogy a hallgató elhagyja-e az alkalmazás ablakát. Feltételezve, hogy ekkor valamilyen programot szeretne elindítani annak érdekében, hogy megadhassa a helyes választ, az alkalmazás bezárja a vizsgát, és a hallgató eredménye elégtelen lesz.

Ezen megoldások azonban nem feltétlenül nyújtanak elegendő védelmet a csalásokkal szemben. Bár a felhasználóknak általában nincs arra lehetőségük, hogy a telepített alkalmazások fájljait módosítsák, az adminisztrátori jelszó esetleges megszerzésével erre alkalmuk nyílna. Így a vizsga megkezdése előtt, az engedélyezett programok közül tetszőleges alkalmazás állományát felülírva, akár egy böngésző elindítására is lehetőségük lenne.

Bár sok alkalmazás, így az Internet Explorer is leáll, ha a futtaható állománya átnevezésre kerül, a Chrome, illetve a Mozilla Firefox is elindítható ezzel a módszerrel. Ennek kivédése érdekében, az alkalmazás minden engedélyezett programhoz létrehoz egy MD5 ellenőrzőösszeget. Ennek segítségével egyértelműen meg lehet majd állapítani, hogy a felhasználó valóban azt az alkalmazást szeretné-e elindítani, ami számára korábban engedélyezve lett.

Azt gondolom, hogy a három funkció együttesen már megfelelő védelmet fog jelenteni a csalásokkal szemben. Így megakadályozható válik, hogy a felhasználók tetszőleges alkalmazáshoz hozzáférhessenek a vizsga során.

## 4.1. Hallgatók, oktatók kezelése

A program elkészítése során a kliens-szerver modellt fogom alkalmazni, ahol a szerver az adminisztrátort, a kliens pedig a hallgatót, illetve az oktatót testesíti meg.

Új felhasználót kizárólag az adminisztrátor tud majd létrehozni. Feltételezve, hogy a szerverhez egyedül ő férhet hozzá, a bejelentkezés mindössze a kliens esetében lesz szükséges.

A felhasználói adatok, a vizsgák, valamint az eredmények mentésére a szerializációt fogom használni [7]. Ennek segítségével ugyanis egyszerűen és gyorsan elmenthetőek az egyes objektumok állapotinformációi. Mivel azonban a hallgatók adatait és vizsgáit nem igazán szerencsés ugyanabban a könyvtárban tárolni, ezért a fejlesztés legelső lépésében szükség van a megfelelő könyvtárszerkezet kialakítására. Ennek érdekében két könyvtárat hoztam létre:

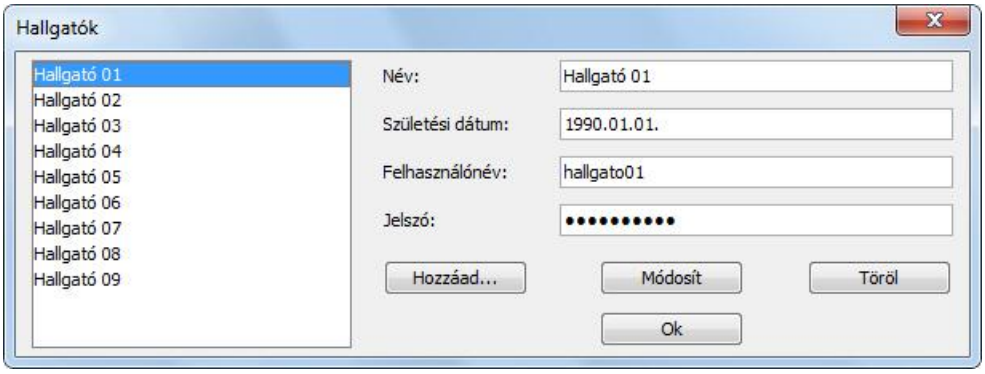
- **adatok**: A hallgatók, az oktatók, valamint az engedélyezett programok adatainak mentéséhez.

- **vizsgak**: Az oktatók által létrehozott számonkérések, valamint a hallgatók által megírt vizsgák is itt kerülnek tárolásra, a saját felhasználónevükkel megegyező könyvtárban.

Mivel az alkalmazásnak el kell tudnia tárolni a kitöltött vizsgákat, valamint azok eredményeit, a programba kizárólag regisztrált felhasználók tudnak majd bejelentkezni. Vendégek számára nem lesz elérhető. A hallgatók, illetve az oktatók egyértelmű azonosításához mindenképp szükség lesz névre, felhasználónévre, illetve jelszóra. Egy oktatási intézményben azonban könnyen elképzelhető, hogy több azonos nevű személy számára is lehetővé kell tenni az alkalmazás használatát. Ezért a fenti adatokon túl, a születési dátum megadására is szükség lesz.

A hallgatók, illetve az oktatók is a Felhasznalo osztály példányaiként kerülnek elmentésre. A tárolás során azonban szét kell választani a két szerepkört. Ez azért is szükséges, mert bejelentkezésnél egyértelműen meg kell határozni, hogy az adott felhasználó hallgatóként vagy oktatóként van-e nyilvántartva a programban. Ellenkező esetben egy hallgató akár a saját eredményeit is szabadon módosíthatná. Továbbá a felhasználói felületen is külön menüpontban lesz lehetőség hallgatók, illetve oktatók hozzáadására, valamint az adataik módosítására.

A szerver elindítása után, a Hallgatók, illetve az Oktatók menüpontban van lehetőség felhasználók hozzáadására, módosítására és törlésére (5. ábra).




*5. ábra: Hallgatók kezelése*

A Hozzáad gombra kattintva, lehetőség van új felhasználó létrehozására. Az adatok megadását követően a program ellenőrzi, hogy minden mező ki van-e töltve, a jelszó erőssége megfelelő-e, valamint megvizsgálja, hogy az adott felhasználónév szerepel-e az adatbázisban. A jelszó akkor megfelelő, ha legalább nyolc karakter hosszú, valamint tartalmaz betűket és számokat is. Természetesen a nagyobb biztonság érdekében célszerű lenne, ha kis- és nagybetűket vegyesen is tartalmazna, ezt azonban már túl erős megszorításnak érezem, ezért a jelszót e nélkül is elfogadja az alkalmazás.

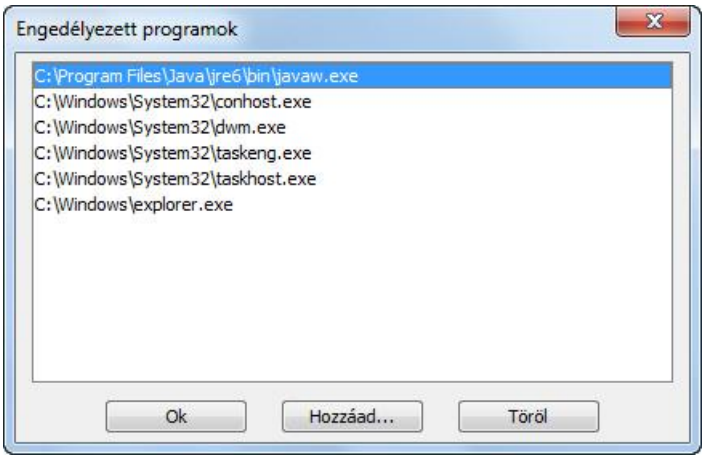
## 4.2. Engedélyezett programok

A felhasználói adatbázis létrehozásán túl, rendkívül fontos az engedélyezett programok listájának elkészítése is. Az alkalmazás ugyanis ez alapján fogja eldönteni, hogy a vizsga közben milyen programok futhatnak, és melyek azok, amelyeket be kell zárni. A lista összeállításánál érdemes megnézni, hogy az operációs rendszeren milyen telepített alkalmazások találhatóak, és ezek közül melyek azok, amelyek minden rendszerindításnál elindulnak. Ilyenek például a különböző vírusirtók, amelyeket a program nem fog tudni bezárni. Így ezeket mindenképp ajánlott hozzáadni a listához.

Az automatikusan elinduló programokat az msconfig parancs kiadása után, az Automatikus indítás fülön lehet megtekinteni. Az itt szereplő elemek közül érdemes letiltani azokat, amelyek futtatása nem feltétlenül szükséges.

De nem csak a telepített programokra, a Windows egyes folyamataira is érdemes figyelmet fordítani. Ilyen például az msfeedssync.exe, amely meghatározott időnként frissíti az RSS hírcsatornákat. Ha ez az időpont egybeesik a vizsga idejével, az alkalmazás ezt minden figyelmeztetés nélkül be fogja zárni. Érdemes továbbá átnézni a különböző frissítési szolgáltatásokat is. Különös tekintettel a beépített Windows Update-re. Automatikus frissítés esetén ugyanis ez különböző kérdéseket jeleníthet meg, amikre ha a felhasználó reagálni szeretne, akkor el kell hagynia az alkalmazás ablakát, aminek hatására a vizsgája elégtelen lesz.

Az Engedélyezett programok megnyitása után (6. ábra), a Töröl gomb segítségével tetszőleges alkalmazás eltávolítható a listából. A javaw.exe törlésére azonban nincs lehetőség. Ez ugyanis feltétlenül szükséges a program futtatásához. Ha valamilyen okból mégis hiányozna, vagy helytelenül szerepelne az elérési útja, az alkalmazás megnézi a JAVA_HOME környezeti változó értékét, ezt követően pedig felveszi a fájlt a listába.

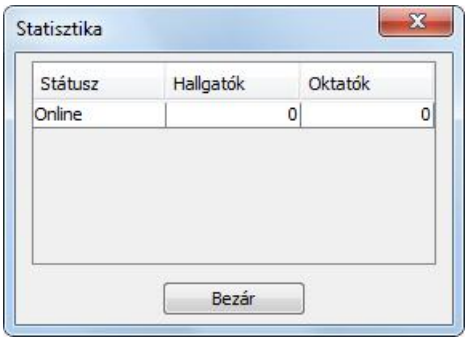



*6. ábra: Engedélyezett programok*

Természetesen új alkalmazás felvételére is van lehetőség, a Hozzáad gombra kattintva. Ekkor megjelenik egy új ablak, ahol a Tallózás gomb segítségével, kényelmesen meg lehet keresni a kívánt fájlt. Mivel azonban nem biztos, hogy a szervert futtató számítógépen is megtalálhatóak ugyanazok a programok, amiket a vizsga során engedélyezni szeretnénk, a teljes elérési utat kézzel is meg lehet adni. Ebben az esetben az Ok gombra kattintva, a program ellenőrizni fogja, hogy létezik-e az adott fájl. Amennyiben nem található, egy figyelmeztető üzenetet jelenít meg. Az elérési út hozzáadására csak ezt követően lesz lehetőség.

Bár kézzel is meg lehet adni az engedélyezni kívánt programot, ez azonban nem javasolt. Az alkalmazás ugyanis egy ellenőrzőösszeget is elment minden egyes programhoz, hogy a hallgatók ne tudjanak csalni az állományok átnevezésével [8]. Ekkor azonban ez az összeg -1 lesz, és az alkalmazás nem fogja tudni ellenőrizni az állomány valódiságát. Így az engedélyezett program helyett, akár egy böngészőt is el lehet majd indítani.

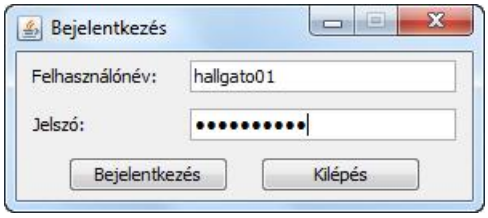
Habár a szerver elindításánál, illetve leállításánál is megjelenik egy információs üzenet, a szerver státuszát a Statisztika menüpontban is meg lehet tekinteni, ahol lehetőség van a belépett hallgatók és oktatók számának nyomon követésére is (7. ábra).




*7. ábra: Statisztika*

## 4.3. Bejelentkezés

A kliens elindítását követően, hallgatóként és oktatóként is lehetőség van a bejelentkezésre (8. ábra). A felhasználónév és a jelszó mezőt mindkét esetben kötelező kitölteni, ellenkező esetben egy hibaüzenetet jelenít meg az alkalmazás.




*8. ábra: Bejelentkezés*

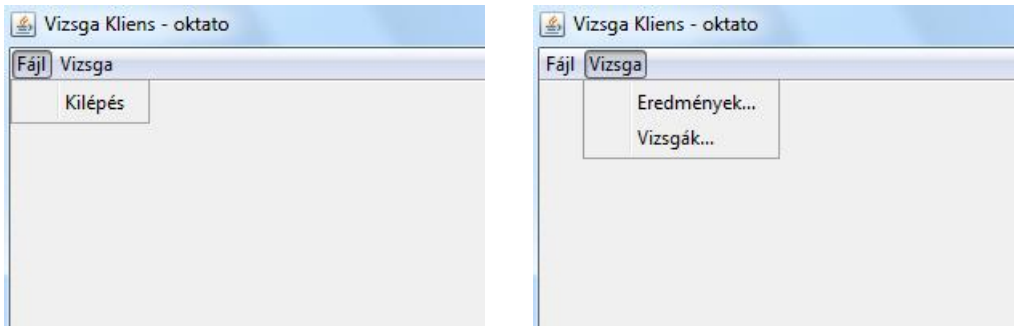
A felhasználói adatok megadása után, a Bejelentkezés gombra kattintva, az adatok elküldésre kerülnek a szervernek, amely megvizsgálja, hogy a megadott információk helyesek voltak-e, majd ennek eredményéről visszajelzést küld a kliensnek.

- **hallgato**: Ha a felhasználónév egy hallgatóhoz tartozik.
- **oktato**: Ha a felhasználónév egy oktatóhoz tartozik.
- **hiba**: Hibás felhasználónév vagy jelszó esetén.

Elképzelhető, hogy a bejelentkezés pillanatában a szerver valamilyen ok miatt nem elérhető, ekkor az alkalmazás egy hibaüzenet jelenít meg.

A bejelentkezés folyamatát igyekeztem a leginkább felhasználóbaráttá tenni. Elsődleges célom az volt, hogy a felhasználóknak lehetősége legyen egér használata nélkül, csupán az ENTER billentyű lenyomásával bejelentkezni. Ennek érdekében implementáltam a KeyListener interfészt, amely figyeli a leütött, illetve a felengedett billentyűket. Ha pedig az aktuálisan leütött billentyű megegyezik az ENTER-rel, a bejelentkezési adatok azonnal elküldésre kerülnek a szervernek. Ha valamelyik adat hibás volt, mindkét mező tartalma törlődik, a fókusz pedig a felhasználónév mezőre kerül. Így elérhető, hogy a bejelentkezés során egyáltalán ne legyen szükség egér használatára.

Ha a bejelentkezési adatok helyesek voltak, megjelenik a program felülete (9. ábra), amely a hallgatók és az oktatók esetében is azonos menüelemeket tartalmaz. De természetesen ezeken belül mindkét felhasználói csoport más-más funkciókhoz férhet hozzá.

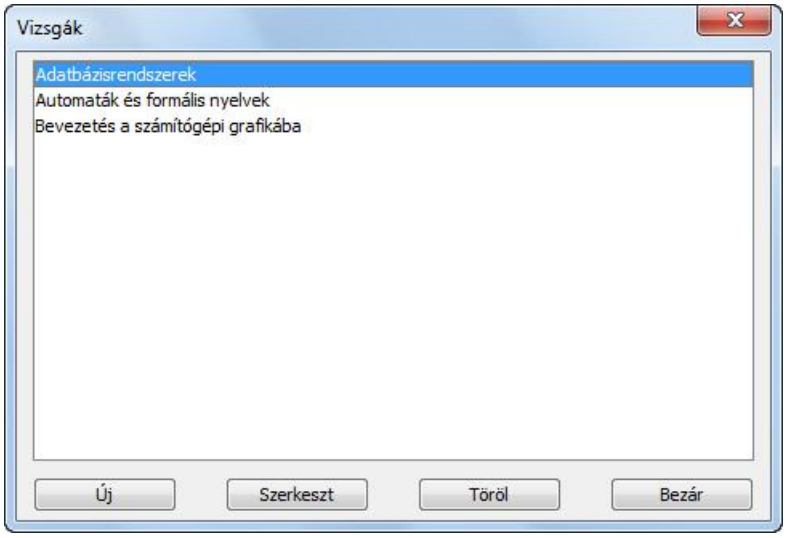



*9. ábra: Alkalmazás menüje*

- **kilépés**: A program bezárása.
- **eredmények**: A vizsgák eredményeinek megtekintése. Illetve az oktatók itt értékelhetik a hallgatók által megírt esszéket.
- **vizsgák**: A hallgatók itt tekinthetik meg a számukra elérhető vizsgákat. Az oktatók továbbá létrehozhatnak új vizsgákat, valamint szerkeszthetik, illetve törölhetik is azokat.

## 4.4. Vizsga létrehozása

A Vizsgák menüpontra kattintva, megjelenik az elérhető vizsgák listája (10. ábra). A hallgatók csak azon vizsgákat látják, amelyeket az oktatók kijelöltek a számukra, és amelyeket még nem töltöttek ki. Az oktatók ezzel szemben az összes, általuk létrehozott vizsgát meg tudják tekinteni.

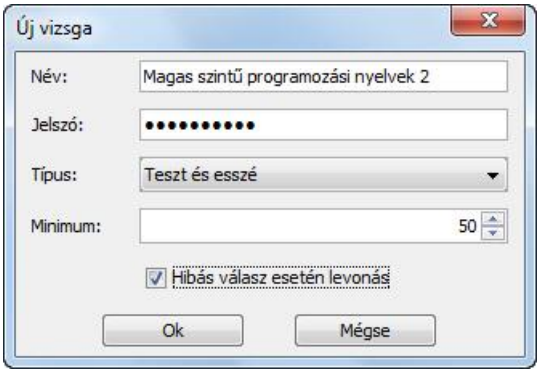



*10. ábra: Vizsgák listája (oktató)*

Amennyiben a vizsgák listája még üres, a program a Szerkeszt, a Töröl és a Megnyit gombokra kattintva, egy hibaüzenetet jelenít meg, az adott funkciónak megfelelően.

Biztonsági okokból, ha az oktatók egyszerre több vizsgát is törölni szeretnének, azt csak egyesével tehetik meg. Nem tudnak egyszerre több vizsgát kijelölni. Törlésnél a program egy dialógusablakot is megjelenít, ahol az Igen gombra kattintva távolítható el az adott elem a listából.

Vizsgát kizárólag az oktató tud létrehozni, az Új gombra kattintva. Ekkor egy új ablak jelenik meg, ahol a vizsgára vonatkozó legfontosabb adatokat lehet megadni (11. ábra). Az ablak megtervezésénél törekedtem arra, hogy az oktatók által használt legfontosabb opciók megtalálhatóak legyenek rajta, ugyanakkor könnyen áttekinthető legyen, és ne váljon túlzsúfolttá a felülete. Elsődleges célom az volt, hogy az egyes beállítási lehetőségek azon oktatók számára is egyértelműek legyenek, akik csupán első alkalommal használják az alkalmazást.




*11. ábra: Vizsga létrehozása*

- **név**: A vizsga neve. Kötelező megadni, a későbbiekben ugyanis a hallgatók, illetve az oktatók csak ez alapján tudják majd azonosítani a különböző vizsgákat.

- **jelszó**: Megadása opcionális. Amennyiben az oktató létrehoz egy jelszót, a hallgatók kizárólag ennek birtokában tudják majd megnyitni a vizsgát. Annak érdekében, hogy a jelszót ne lehessen könnyen kitalálni, a program ellenőrzi az erősségét. Csak olyan jelszót lehet megadni, ami legalább nyolc karakter hosszú, valamint tartalmaz betűket és számokat is.

- **típus**: Az oktatók három típusú vizsga közül választhatnak:

    - teszt és esszé: A hallgatóknak először egy tesztet kell kitölteniük. Az esszé megírására csak ezt követően lesz lehetőségük. Ennél a típusnál az oktatók megadhatnak egy minimum értéket, amelyet a hallgatóknak el kell érniük a teszt során, különben a vizsgájuk elégtelen lesz.

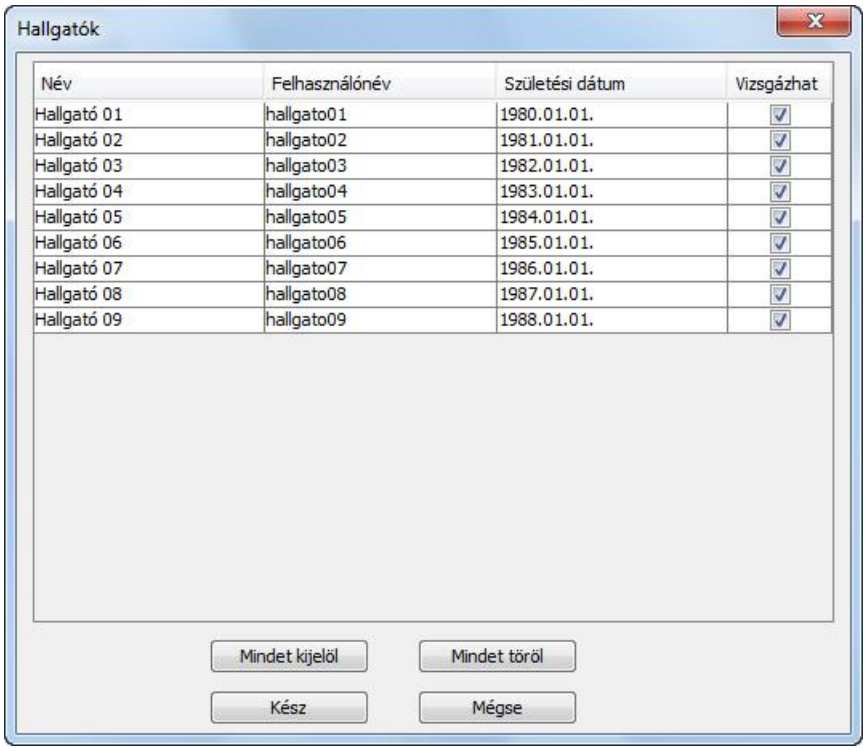
    - esszé: Az oktatóknak csupán a kérdést kell megadniuk. A hallgatók erre tetszőleges terjedelemben válaszolhatnak. Értékelése a vizsga után kézzel, az Eredmények menüpontban történik.

    - teszt: Az oktatók tetszőleges számú kérdést létrehozhatnak, amelyeknél akár több helyes választ is megjelölhetnek. Javítása automatikusak történik, a megadott helyes válaszoknak megfelelően.

- **minimum**: Kizárólag Teszt és esszé típusú vizsgánál használható. A teszt során, a hallgató által elérendő minimum meghatározását szolgálja.

- **hibás válasz esetén levonás**: Teszt és esszé, valamint Teszt típusoknál is használható. Kiválasztása esetén, ha egy kérdésnél a hallgató nem a helyes választ, esetleg helyes válaszokat jelöli meg, a kérdésre -1 pontot kap. Ezzel elkerülhető, hogy a hallgatók megtippeljék azokat a kérdéseket, amelyek válaszaiban nem teljesen biztosak.

A vizsga adatainak megadása után, az oktatók kiválaszthatják, hogy mely hallgatók számára szeretnék elérhetővé tenni a vizsgát (12. ábra). Mivel elképzelhető, hogy egyelőre nem szeretnék, hogy rajtuk kívül bárki is hozzáférjen, nem kötelező kiválasztaniuk valakit.

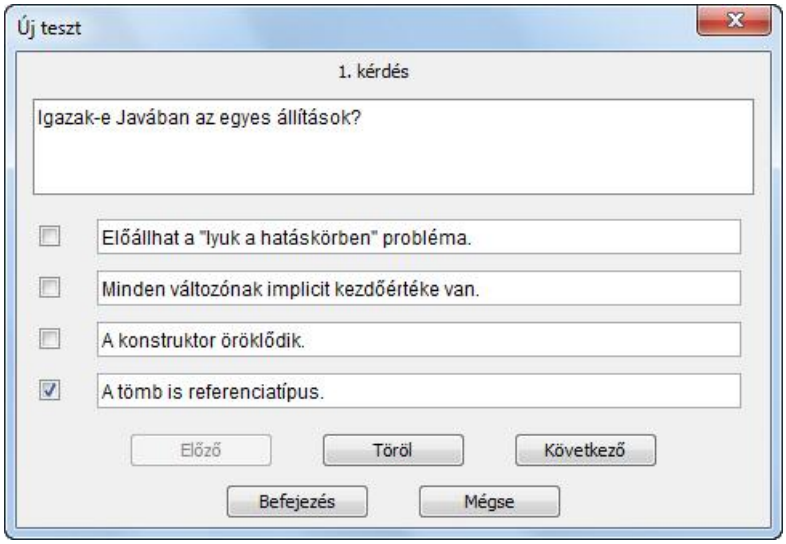



*12. ábra: Hallgatók listája*

Mivel könnyen lehetséges, hogy több azonos nevű hallgató is van egy intézményen belül, az egyértelmű megkülönböztethetőség érdekében, a név mellett a felhasználónév és a születési dátum is látható a táblázatban.

Bár a példában csak néhány hallgató adatai szerepelnek, elképzelhető, hogy akár több száz felhasználó is regisztrálva van, akik közül már lényegesen nehezebb a megfelelő hallgatók kiválasztása. Ennek elősegítése érdekében létrehoztam egy Mindet kijelöl, illetve Mindet töröl gombot, amelyek segítségével egyszerűen módosíthatóak a kijelölések.

A vizsga adatainak megadása után, amennyiben a számonkérés típusánál Teszt és esszé vagy Teszt lett kiválasztva, az oktató megadhatja a vizsga során megjelenítendő kérdéseket (13. ábra).




*13. ábra: Tesztkérdés létrehozása*

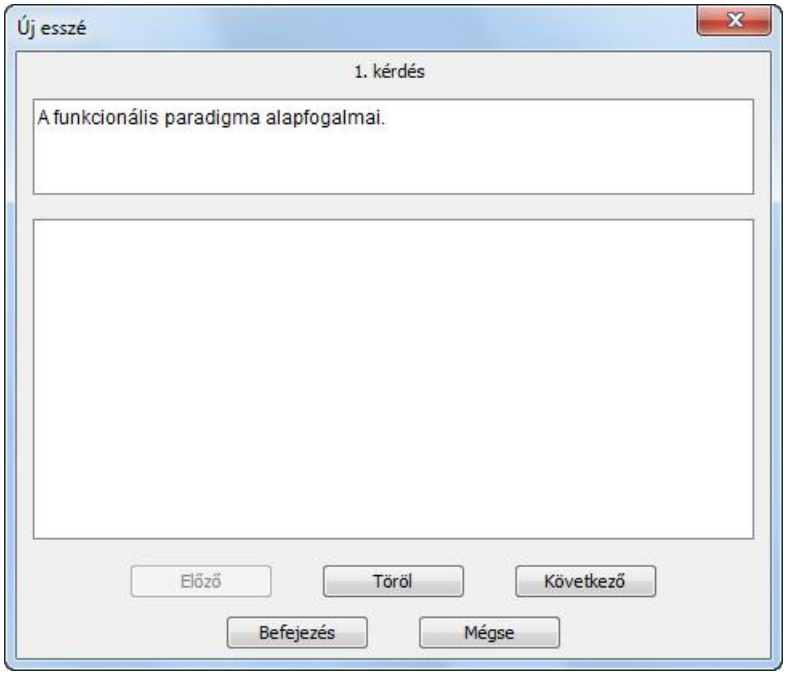
Mivel egy teszt 50, 100, vagy akár még több kérdésből is állhat, ezek létrehozását igyekeztem a lehető legkényelmesebbé tenni az oktatók számára. Ennek érdekében az ablak felső részében elhelyeztem egy számlálót, amelynek segítségével mindig nyomon lehet követni, hogy eddig hány kérdés lett létrehozva. Továbbá készítettem egy Előző, illetve egy Következő gombot, amelyekkel könnyedén lehet váltani a kérdések között. Az első kérdéshez érve, az Előző gomb inaktívvá válik. Ezzel jelezve, hogy a listában nem található az aktuálisat megelőző elem.

A Következő gomb hatására, amennyiben már nem található több elem a listában, a program törli a szövegmezőket, a fókusz pedig a kérdés mezőjére kerül. Így az oktatónak nem kell az egér segítségével odakattintaniuk, hanem egyből be tudják gépelni a következő kérdés szövegét.

Bár az oktatók az összes kérdést szabadon módosíthatják, elképzelhető, hogy egy kérdés tévedésből került elmentésre, amelyet szeretnének törölni. Ennek érdekében létrehoztam egy Töröl gombot. Használatakor először egy megerősítést kér a program, majd igen válasz esetén, az adott elemet eltávolítja a listából.

Új kérdés létrehozásánál, ha az aktuális kérdés már szerepel a listában, a program egy hibaüzenetben tájékoztatja erről az oktatót. Szintén figyelmeztetést jelenít meg az alkalmazás, ha egy kérdésen belül azonos válaszok lettek megadva. Ezek a hibák ugyanis a teszt kiértékelésénél fals eredményhez vezethetnek, hiszen az alkalmazás nem tudja eldönteni, hogy melyik válasz pontosan melyik kérdéshez tartozik.

A programban az oktatóknak lehetőségük van esszét is létrehozni (14. ábra). Az esszé felülete nagyon hasonló a teszt felületéhez, azonban itt a válaszlehetőségek helyett egy szövegmező található, amelyben a hallgatók tetszőleges terjedelemben válaszolhatnak a program által feltett kérdésekre.




*14. ábra: Esszékérdés létrehozása*

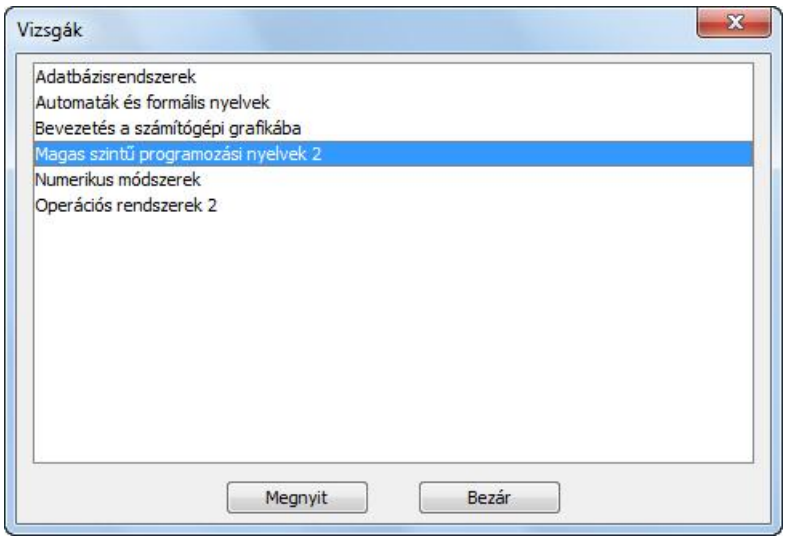
Bár általában az esszé kérdéssor lényegesen kevesebb részből áll egy teszthez képest, a kérdések közötti jobb eligazodás érdekében itt is létrehoztam egy számlálót, valamint az Előző, illetve a Következő gombokat.

Esszé esetében az oktatóknak csak a kérdést kell megadniuk, a lehetséges választ természetesen nem. Erre a program nem is nyújt lehetőséget. A hallgatók által megadott válaszokat később, az Eredmények menüpontban tudják majd megtekinteni, ahol lehetőségük van azok értékelésére is.

Az esszé kérdéseinek megadása után, az oktatók a Befejezés gombra kattintva tudják elmenteni a vizsgát. A mentés sikerességéről a program egy információs üzenetet is megjelenít. Mivel a korábban megadott adatok mentése is ekkor történik, a Mégse gomb hatására az alkalmazás az összes addig végrehajtott változást elveti. Beleértve az esetleges tesztkérdéseket is.

## 4.5. Vizsga megnyitása

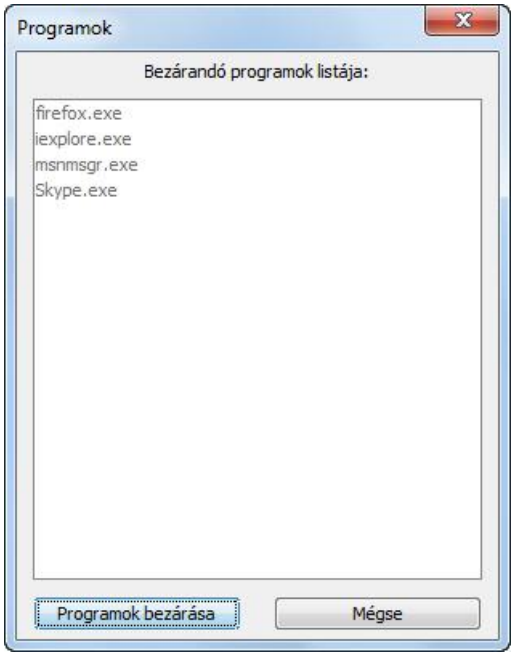
Hallgatóként bejelentkezve az alkalmazásba, a Vizsgák menüpontban lehetőség van az oktatók által feltöltött vizsgák megnyitására (15. ábra).




*15. ábra: Vizsgák listája (hallgató)*

Ekkor a program először ellenőrzi, hogy futnak-e olyan alkalmazások, amelyek fájlneve szerepel az engedélyezett programok listájában. Amennyiben igen, megnézi, hogy a teljes elérési út megegyezik-e a listában lévővel. Könnyen előfordulhat ugyanis, hogy a hallgatók megszerzik a futtatható programok listáját, és az eredeti fájlnevet módosítva, megpróbálnak elindítani egy böngészőt vagy más alkalmazást. Ehhez ugyanis elegendő a vizsga előtt futtatniuk egy feladatkezelőt és megvizsgálni, hogy mely programok esetében nem szükséges a bezárás.

Ha vannak olyan futó alkalmazások, amelyek elérési útja nem egyezik meg a listában lévővel, vagy amelyek egyáltalán nem is szerepelnek abban, a program egy új ablakban megjeleníti őket, és engedélyt kér a bezárásukra (16. ábra).




*16. ábra: Bezárandó programok listája*

A képen látható, hogy az alkalmazás többek között a Firefox, illetve az Internet Explorer bezárását kéri, amelyek futtatása természetesen nem javasolt vizsga közben. Amennyiben a felhasználó a Programok bezárására kattint, ezek automatikusan bezáródnak. Ehhez az alkalmazás a **taskkill /f /im <folyamatkép>** parancsot használja:

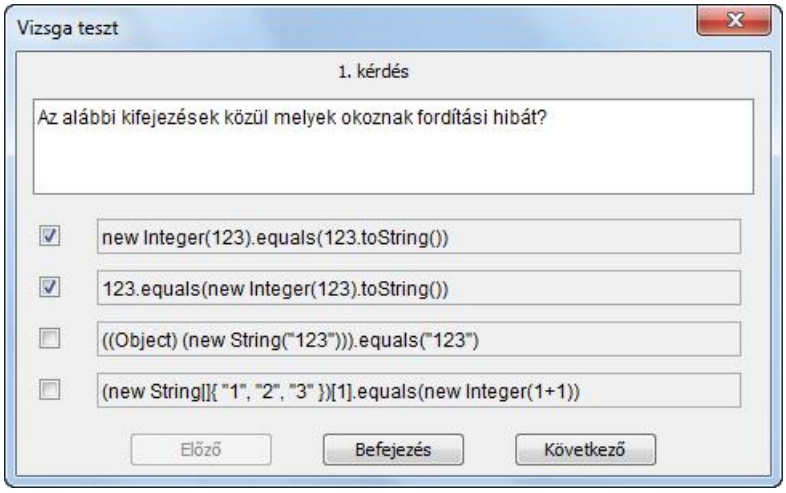
- f: A folyamat lezárásának kikényszerítése.
- im: A lezárandó folyamat képének megadása.

Az f kapcsoló használata azért szükséges, mert az alapértelmezett beállítások szerint, mindkét böngésző megerősítést kér a felhasználótól több lap, egyszerre történő bezárása előtt. Bár az alkalmazás addig nem lép tovább, amíg nem sikerült az összes, nem engedélyezett programot bezárni, a vizsga során is problémák merülhetnek fel, ha a felhasználó egy böngészőt próbál elindítani. De más programoknál is jelentkezhetnek hibák, ezért a kényszerbezárás használata feltétlenül szükséges.

Amennyiben az alkalmazásnak sikerült minden, nem engedélyezett programot bezárnia, megvizsgálja, hogy az oktató létrehozott-e jelszót a vizsga megnyitásához. Ha igen, akkor megjeleníti azt az ablakot, ahol a hallgatónak lehetősége van ezt megadnia. Ha pedig nem tartozik hozzá jelszó, akkor az alkalmazás, a vizsga típusának megfelelően megjeleníti az első teszt, vagy az első esszé kérdést.

Bár a jelszó beállításánál az oktatóknak egy olyan jelszót kellett megadniuk, ami egyaránt tartalmaz számokat és betűket is, ez akár az oktató keresztnevével és születési dátumával is megegyező lehet, ami természetesen nem egy megfelelő erősségű jelszó. Annak érdekében, hogy a hallgatók ne kísérletezzenek a jelszavak megfejtésével, a program legfeljebb két alkalommal engedélyezi hibás jelszó megadását. Harmadik alkalommal a hallgató vizsgája elégtelen lesz.

Ha a hallgató által megadott jelszó helyes volt, az alkalmazás megnyitja a hozzá tartozó teszt- vagy esszé kérdéssort (17. ábra), ezt követően pedig folyamatosan ellenőrzi a hallgató tevékenységét.




*17. ábra: Első tesztkérdés*

Mivel az elindított programok állandó ellenőrzése igen nagy processzorterhelést jelentene, az alkalmazás csupán 3 másodpercenként vizsgálja meg a futattott programokat. Ha új alkalmazást talál, azt természetesen azonnal bezárja. Mivel a rendszer a vizsga során több olyan folyamatot is elindíthat, amelyek nem szerepelnek az engedélyezett programok listájában, a hallgató számára mégsem járnak semmilyen előnnyel, nem tartottam volna szerencsés döntésnek a bezárások számának korlátozását. Így azonban akár az is elképzelhető, hogy a hallgató elhelyez egy szöveges dokumentumot az asztalra, amelyben a válaszok találhatóak, majd ezt időről-időre megnyitja. Ennek érdekében az alkalmazás nem csak az elindított programokat figyeli, hanem azt is, hogy a vizsga alkalmazás ablaka elveszti-e a fókuszt. Azaz a hallgató megpróbál-e ablakot váltani a vizsga során. Ha igen, a program minden előzetes figyelmeztetés nélkül, elégtelennek nyilvánítja a vizsgát.

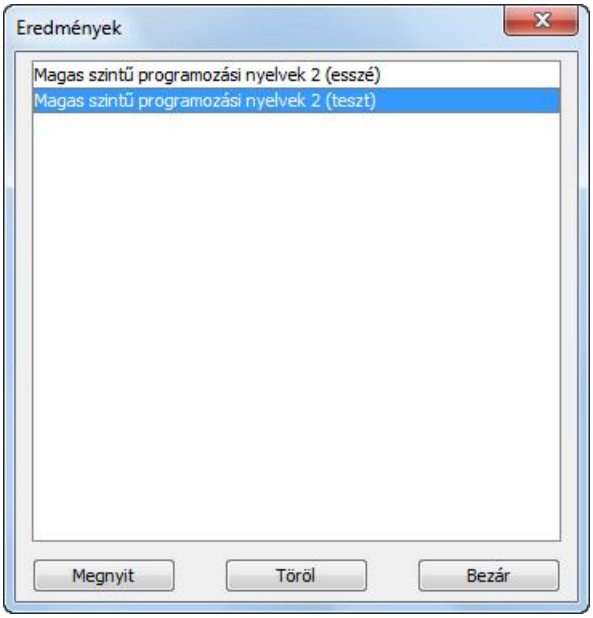
Bár a fókuszfigyelés miatt kockázatos, de elméletben a hallgatók létrehozhatnak az asztalon egy batch állományt, ami a taskkill /f /im javaw.exe parancs futtatásával, bezárja a vizsga programot. Ez számukra különösen hasznos lehet, ha néhány kérdésre nem tudják a választ, a bezárás után ugyanis szabadon hozzáférhetnek bármelyik alkalmazáshoz. Ennek érdekében, a vizsga megnyitásakor a program egy üres feladatsort ment el a hallgató könyvtárába, a vizsga eredményéhez pedig 0 % kerül eltárolásra. Továbbá a program felülírja a vizsga adatait tartalmazó állományt is, így a hallgatónak nem lesz lehetősége újból megnyitni a vizsgát. Természetesen amennyiben a hallgató szabályosan, a Befejezés gomb megnyomásával fejezi be a vizsgát, az eredményei felülírásra kerülnek, így az Eredmények menüpontban már a valós értékeket tekintheti meg.

Amennyiben a vizsga esszé és teszt kérdéseket is tartalmaz, és az oktató a vizsga létrehozásakor megadott egy minimum értéket, a teszt megírását követően a program ellenőrzi, hogy a hallgatónak sikerült-e teljesítenie ezt az eredményt. Ha nem sikerült, akkor az alkalmazás az esszé kérdéseket már nem jeleníti meg, a hallgató a vizsgát nem tudja folytatni.

## 4.6. Eredmények megtekintése

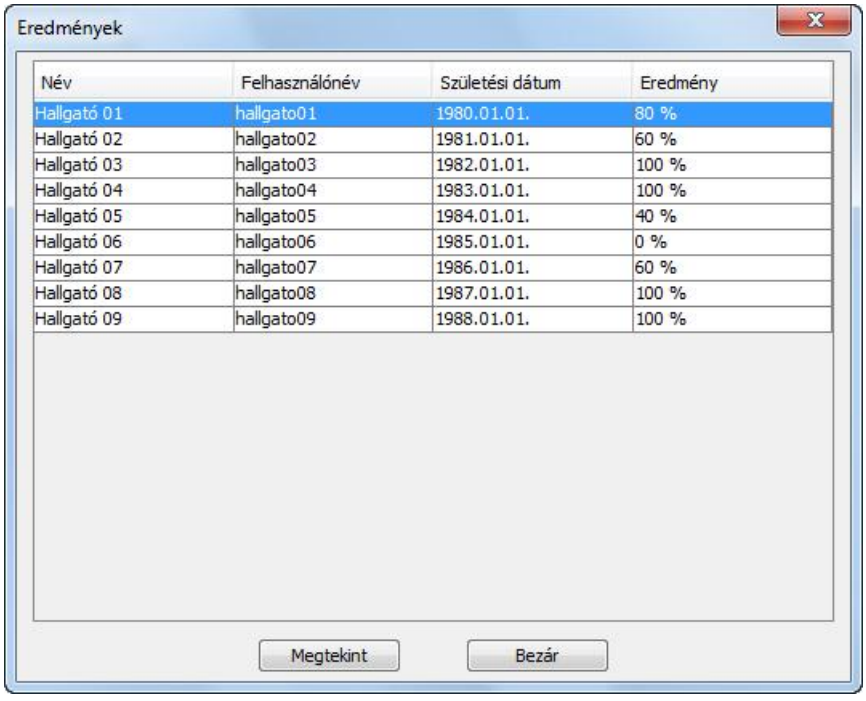
Az Eredmények menüpontban az oktatóknak először ki kell választaniuk, hogy pontosan melyik típusú vizsga eredményeit szeretnék megtekinteni (18. ábra). Erre azért van szükség, mert az oktatókhoz több vizsga, a vizsgákhoz pedig akár több száz eredmény is tartozhat. Ha pedig ezeket ömlesztve, egyetlen táblázatban jelenítené meg a program, akkor a táblázat akár több ezer soros is lehetne, amiből rendkívül nehézzé válna az egyes eredmények kikeresése. Még akkor is, ha az eredményeket a hallgatók vagy a vizsgák nevei szerint rendezve jelenítené meg az alkalmazás.

A hallgatók esetében nincs szükség a vizsga előzetes kiválasztására, ők közvetlenül is meg tudják majd tekinteni az eredményeiket (19. ábra). Hozzájuk ugyanis általában csak néhány vizsga tartozik, amelyek eredményei közül is csupán egy-egy vonatkozik rájuk. Ezek pedig még akkor is áttekinthetőek lesznek, ha a program egyetlen táblázatban jeleníti meg azokat.




*18. ábra: Eredmények listája*

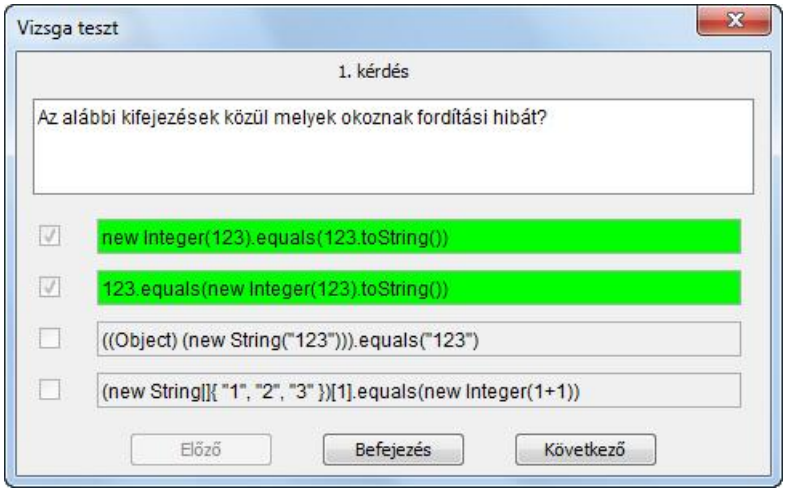
Természetesen a hallgatók és az oktatók esetében a táblázatnak más-más oszlopokat kell tartalmaznia. Mivel az oktatók előzetesen már kiválasztották, hogy melyik vizsga eredményeit szeretnék megtekinteni, ezért náluk a vizsga nevét nem szükséges feltüntetni. Viszont a hallgatók egyértelmű beazonosításához szükség van az egyes hallgatók nevére, felhasználónevére és születési dátumára.




*19. ábra: Teszteredmények*

Mivel a hallgatóknak nem kell előzetesen kiválasztaniuk, hogy melyik vizsga eredményét szeretnék megtekinteni, az esetükben szükség van a vizsga nevének és típusának a feltüntetésére is. Azonban náluk az egyértelmű beazonosításra nincs szükség, ezért a táblázatban nem kell szerepelni a személyes adataiknak.

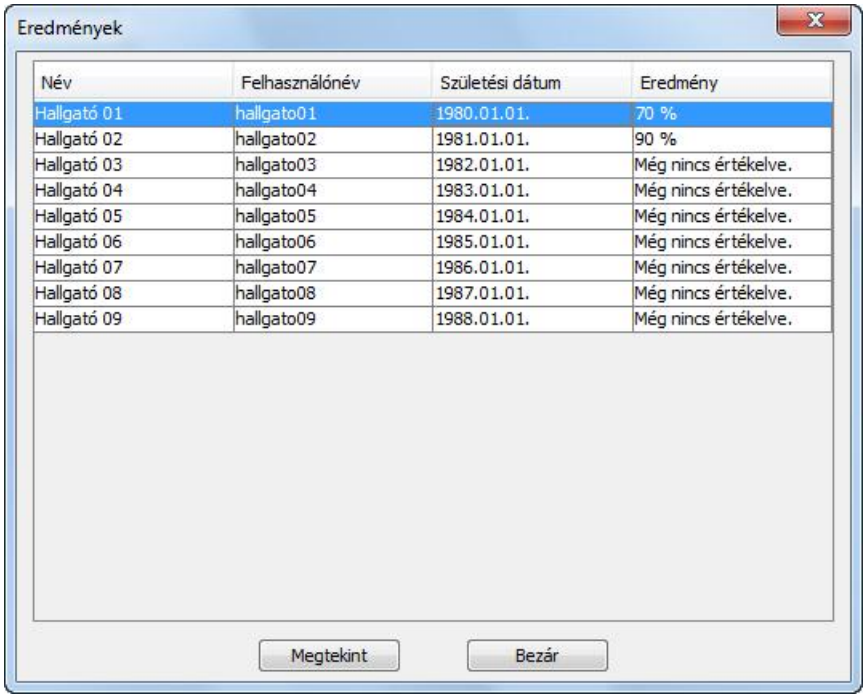
Az Eredmények ablakban a hallgatók, illetve az oktatók számára is elérhető a Megtekint gomb, amellyel megjeleníthetik az esszékre, valamint a tesztekre megadott válaszokat (20. ábra).




*20. ábra: Teszt válaszainak megtekintése*

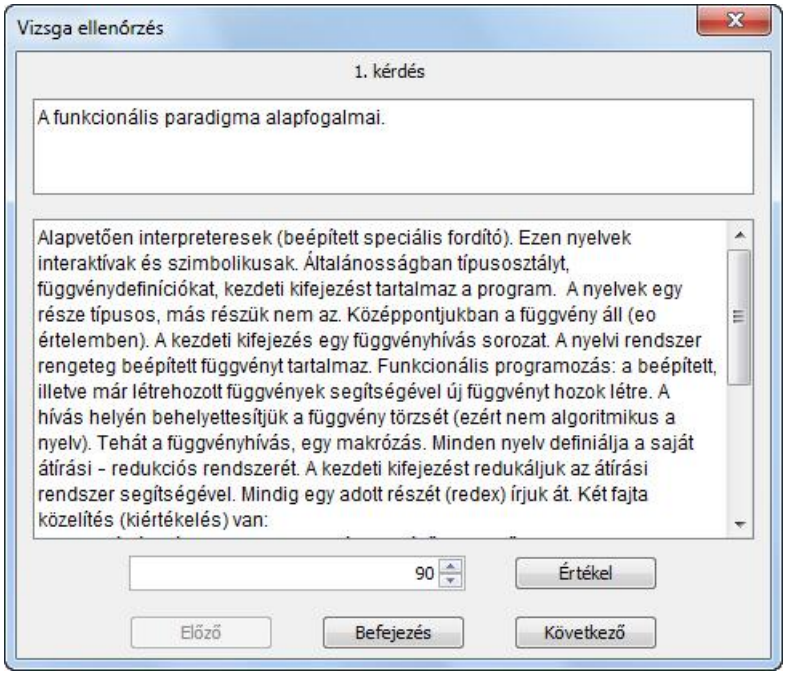
A tesztek esetében a program először betölti a szerverről az oktató, illetve a hallgató könyvtárában lévő **<vizsga neve>-teszt.obj** állományt, majd megvizsgálja, hogy az oktató által megadott válaszok, megegyeznek-e a hallgató válaszaival, illetve, hogy a hallgató jelölt-e meg további válaszokat. Ha a hallgató válasza megfelelő, akkor a program zöld háttérszínnel megjelöli azt, ellenkező esetben a háttérszín piros lesz. Ennek segítségével a hallgatók és az oktatók is könnyen nyomon tudják követni, hogy melyik kérdésnél, pontosan mi volt a hiba.

Bár a tesztet a vizsga befejezését követően azonnal kijavítja az alkalmazás, sajnos az esszé esetében erre nincs lehetőség. A hallgatók ugyanis szabadon kifejthetik a gondolataikat az adott kérdésről, amit a program nem képes megfelelően értékelni. Ezeket a válaszokat az oktatóknak kell minősíteniük. Azon vizsgáknál, amelyek már értékelve lettek, megjelenik az eredmény. Egyébként a „Még nincs értékelve.” felirat látható a táblázatban (21. ábra).




*21. ábra: Esszéeredmények*

Az oktatók a Megtekint gomb segítségével nyithatják meg az egyes esszéket. Ekkor egy új ablakban megjelennek a kérdések, illetve a hallgatók által adott válaszok (22. ábra). Az ablak felépítése nagyon hasonló ahhoz, ami az esszé létrehozásánál volt látható, csupán az értékeléshez szükséges objektumokkal egészült ki.




*22. ábra: Esszé válaszainak megtekintése*

Az értékelés az Értékel gomb segítségével történik. Ekkor az oktató által megadott eredmény elmentésre kerül a hallgató könyvtárába, majd a táblázatban is frissül a hozzá tartozó érték. Az oktatók tetszőlegesen sokszor értékelhetnek egy vizsgát, így amennyiben egy hallgatóhoz téves eredmény került elmentésre, utólag lehetőségük van azt javítani.

# 5. Felmerülő problémák

A program fejlesztése során az első megoldandó probléma a megfelelő kommunikáció megteremtése volt a kliens és a szerver között. Bár az egyetemi tanulmányaim során a Hálózati architektúrák és protokollok tárgy gyakorlatának teljesítéséhez szükség volt egy hálózati alkalmazás elkészítésére, ezt nem Java, hanem C nyelven kellett implementálni. Így első lépésben a Java hálózatkezelést támogató osztályait, illetve azok metódusait kellett tanulmányoznom.

A hálózatkezelés alapjainak áttekintésében nagy segítségemre volt a Java 2 útikalauz programozóknak 5.0 [9] című könyv, melyet a fejlesztés során többször is segítségül hívtam. Az ebben található részletes elméleti magyarázatok és példakódok segítségével, könnyen megoldhatóvá váltak a felmerülő problémák.

Bár a könyvben egy rendkívül részletes leírás található a hálózatkezelésről, a példában szereplő szerveralkalmazás egy időben csak egy kliens kiszolgálására alkalmas. Ez természetesen számomra nem megfelelő, hiszen akár több tucat hallgatónak is be kell tudnia jelentkeznie egy időben a programba. Ennek megoldása érdekében, több hálózati alkalmazásokkal kapcsolatos könyvet is elolvastam [10], illetve felkerestem a különböző, Java-hoz kapcsolódó oldalakat [11].

Ezek áttanulmányozása után, végül úgy oldottam meg a problémát, hogy definiáltam egy boolean típusú változót, ami a szerver elindításától, illetve leállításától függően, igaz vagy hamis értéket vesz fel. Ezt követően létrehoztam egy új szálat [12], amelyben egy while ciklust futtattam egészen addig, amíg a változó értéke igaz volt. Ezután a cikluson belül, az ss.accept() metódus segítségével figyeltem a beérkező kapcsolatokat, majd az új kapcsolatot átadtam a SzerverKapcsolat osztály számára. Mivel ekkor még mindig csak egy kliens kiszolgálását tudta volna elvégezni az alkalmazás, a SzerverKapcsolat számára egy új szálat hoztam létre. Ezzel lehetővé vált, hogy egy időben tetszőleges számú hallgató is bejelentkezhessen a programba (1. kód).

Ezt követően már csak a kliens és a szerver közötti információátvitelt kellett megvalósítanom, melyhez a BufferedReader és a PrintWriter osztályokat használtam.

```
Thread thread = new Thread(new Runnable() {

    public void run() {
        while (fut) {
            try {
                s = ss.accept();
            } catch (IOException ex) {
                System.out.println(ex.getMessage());
            }
            Thread t = new Thread(new Runnable() {

                public void run() {
                    SzerverKapcsolat szerverKapcsolat =
                              new SzerverKapcsolat(s, szerver);
                }
            });
            t.start();
        }
    }
});
thread.start();
```

*1. kód*

Bár a szerver most már megfelelően kezeli a klienseket, további probléma merülhet fel, ha a felhasználó nem a Fájl menü Kilépés menüpontját használja a kijelentkezésre, hanem a bezárás gombot, esetleg az ALT + F4 billentyűket veszi igénybe. Ekkor ugyanis a Socket nem kerül lezárásra, és a bejelentkezett hallgatók, illetve oktatók számát is hibásan jeleníti meg a program. Ennek elkerülése érdekében, implementáltam a WindowListener interfészt, amellyel többek között megadható, hogy a program milyen utasításokat hajtson végre az ablak bezárásakor (2. kód).

```
public void windowClosing(WindowEvent e) {
    if (oktato) {
        kliens.kilep("oktato");
    } else {
        kliens.kilep("hallgato");
    }
    System.exit(0);
}
```

*2. kód*

Ennek hatására az alkalmazás először megvizsgálja, hogy a bejelentkezett felhasználó oktatóként van-e nyilvántartva a programban. Ezt követően meghívja a kilep metódust, amely eggyel csökkenti a bejelentkezett hallgatók vagy oktatók számát, majd lezárja a Socket-et.

A fejlesztés során a következő jelentős problémával akkor szembesültem, amikor a vizsga megnyitása előtt szerettem volna megvizsgálni, hogy a különböző, aktív alkalmazásokhoz milyen elérési út tartozik. Bár a tasklist paranccsal ki lehet listázni a futó folyamatokat, a hozzájuk tartozó elérési utat egyik kapcsolója segítségével sem lehet meghatározni.

Több különböző fórumot végigolvasva találtam rá az ingyenes, Process Viewer alkalmazásra, amely képes a folyamatokhoz tartozó elérési utakat is megjeleníteni. Mivel azonban nem szerettem volna egy ismeretlen programot felhasználni, ezt végül elvetettem. Részben azért is, mert a Feladatkezelőben sem láthatóak alapértelmezés szerint az elérési utak, de a beállítások módosításával lehetőség van azok megjelenítésére. Így biztos voltam benne, hogy van olyan utasítás a Windows-ban, amellyel meg lehet határozni ezeket.

Hosszas keresés után találtam rá a wmic parancsra, amellyel a rendszer számos más információi mellett, az aktív folyamatok elérési útját is meg lehet tekinteni: **wmic process where name="<fájlnév>" get executablepath**. Ezt követően elsőként megnéztem, hogy az egyes folyamatok közül melyek azok, amelyek neve szerepel az engedélyezett programok listájában. Ezen alkalmazások esetében ugyanis meg kell vizsgálni, hogy az elérési utak megegyeznek-e a listában szereplővel (3. kód). Ha nem egyezik meg, vagy az alkalmazás egyáltalán nem szerepel a listában, akkor be kell zárni.

```
Process process = Runtime.getRuntime().exec("wmic process
          where name=\"" + fajl + "\" get executablepath");
br = new BufferedReader(new InputStreamReader(
          process.getInputStream()));
String sor;
while ((sor = br.readLine()) != null) {
    if (sor.contains("ExecutablePath") || sor.isEmpty()) {
        continue;
    }
    String md5 = md5EllenorzoOsszeg(sor);
    if ((programok.get(program).equals("-1") ||
              programok.get(program).equals(md5)) &&
              sor.toLowerCase().startsWith(program.toLowerCase())) {
        return true;
    } else {
        return false;
    }
}
```

*3. kód*

Ezt követően a fejlesztés során már nem ütköztem jelentős problémába. Természetesen apróbb hibák jelentkeztek, de ezeket viszonylag gyorsan ki tudtam javítani. Az egyik különlegesebb hiba, a vizsga megírását követően, a fókuszfigyelésből adódóan jelentkezett. A program ugyanis a vizsga közben folyamatosan ellenőrzi, hogy a hallgató elhagyta-e azt az ablakot, amelyben a válaszokat adhatja meg az egyes kérdésekre. Ehhez elsőként megvizsgálja, hogy futnak-e olyan alkalmazások, amelyek nem szerepelnek az engedélyezett programok listájában, majd megnézi, hogy a megfelelő ablakon van-e a fókusz. Azonban ha a hallgató pontosan abban a néhány tizedmásodpercben fejezte be a vizsgáját, amikor a program az aktív alkalmazások ellenőrzését végezte, a fókusz már átkerült az eredményt megjelenítő információs üzenetre, így a program tévesen, „A vizsgája elégtelen lett.” üzenetet is megjelenítette. Ez nagyjából minden 40-50. vizsga esetében jelentkezett. Természetesen a két ellenőrzés megcserélésével a hiba könnyen javítható volt, és ezt követően már nem jelent meg egy sikeres vizsga után sem ez a hibaüzenet.

# 6. Tesztelés

Bár az alkalmazás megírása során nagy hangsúlyt fektettem az egyes funkciók tökéletesítésére, illetve magam is rendkívül sokáig teszteltem a programot, az esetlegesen továbbra is fennálló hibák kiderítéséhez feltétlenül szükség volt egy szélesebb körű tesztelésére is.

Sajnos az egyetemen nem volt arra lehetőség, hogy a hallgatók az egyik gyakorlaton, számítógépen írjanak meg egy próbavizsgát, ezért néhány barátomat kértem meg, hogy segítsenek az alkalmazás tesztelésében.

A program tesztelésére a K/4-es teremben, laptok segítségével került sor. Legelőször meg kellett vizsgálni, hogy milyen folyamatok futnak az egyes gépeken, majd ezeket hozzá kellett adni az engedélyezett programok listájához. Mivel a hozzáféréseket és a próbavizsgát még otthon elkészítettem, ezért eközben már mindenki sikeresen be tudott jelentkezni az alkalmazásba.

Első lépésben egy fekete doboz tesztelést akartam megvalósítani. Ezért csupán azt közöltem a tesztelőkkel, hogy az alkalmazás igyekszik megakadályozni a különböző csalásokat. Ezek pontos leírását azonban nem ismertettem. A vizsga megnyitása után, a legtöbben valamilyen böngészőt szerettek volna elindítani. Mivel azonban ezek nem szerepeltek az engedélyezett programok között, illetve az alkalmazás ablakáról is elkerült a fókusz, mindenkinek azonnal elégtelen lett az eredménye.

Ezt követően egy fehér doboz tesztelést is végre kívántam hajtani. Ekkor már részletesen elmagyaráztam, hogy az alkalmazás pontosan hogyan próbálja kiszűrni a csalásokat. Nagyon kíváncsi voltam, hogy ezen információk birtokában sikerül–e valamilyen módszerrel kijátszani a programot. A legtöbben az asztalról próbáltak elindítani különböző alkalmazásokat, de ezeket a program néhány másodpercen belül mindig bezárta. Többen próbálták még a feladatkezelő segítségével leállítani az alkalmazást, hogy utána az interneten rákeresve a kérdésekre, sikeresen meg tudják azokat válaszolni. Ez azonban a legtöbb esetben, a fókuszfigyelés miatt nem sikerült. Aki azonban elég gyors volt, az sem tudta újból megnyitni a vizsgát.

Az egyik érdekes próbálkozás volt, hogy mivel a javaw.exe szerepelt az engedélyezett programok között, ezért néhányan megpróbáltak különböző Java alkalmazásokat elindítani. Mivel azonban a program észlelte, hogy ilyen folyamat már fut, hasonlóan a többi alkalmazáshoz, ezeket is bezárta. Természetesen a kliens kivételével, így lehetőség volt a vizsga folytatására.

A tesztelés során senkinek sem sikerült olyan alkalmazást elindítania, illetve olyan módszert találnia, amivel ki tudta volna játszani a vizsga ellenőrzését. A csalások miatt pedig többeknek elégtelen is lett a vizsgája. Ezt követően mindenki meg tudta tekinteni az eredményét, a teszteknél pedig azt is megnézhették, hogy melyik kérdésnél pontosan milyen hibáik voltak.

A vizsga során a szerveralkalmazással sem voltak tapasztalhatók problémák. A bejelentkezett felhasználók számát, valamint az eredményeiket is folyamatosan nyomon tudtam követni. Ezért az alkalmazásban a rendkívül alapos, előzetes tesztelésnek köszönhetően, nem volt szükség hibajavításra.

# 7. Fejlesztési lehetőségek

Bár az alkalmazás a kezdeti elképzeléseimhez képest számos funkcióval bővült, még mindig több ponton tartalmaz fejlesztési lehetőséget.

A program talán legnagyobb hiányossága, hogy jelenleg csupán egyszerű szöveggel lehet válaszolni az egyes kérdésekre. Ez pedig sok esetben nem elegendő. Különösen a matematikához kapcsolódó számonkéréseknél, ahol a képletek megadásához több speciális karakterre is szükség lenne. Bár a hallgatók ezekre a neveik segítségével is hivatkozhatnak, ez rendkívül nehézzé teszi a képletek megadását, illetve olvasását.

De nem csak speciális karakterekkel, hanem különböző ábrák létrehozásával is lehetne bővíteni a programot. A hallgatóknak ugyanis gyakran kell valamilyen rajzot készíteniük a vizsga során. Erre azonban jelenleg nincs lehetőségük.

Az oktatók munkájának megkönnyítése érdekében hasznos lenne továbbá, ha nem csak a számítógépen tudnák megtekinteni a vizsgákhoz tartozó dokumentumokat, de ki is tudnák azokat nyomtatni. Ez különösen az esszék esetében lenne előnyös, ahol a hallgatók válaszai gyakran több oldalasak is lehetnek. Ezek áttekintése pedig sokkal kényelmesebb lenne nyomtatott formában.

Bár az alkalmazás jelen pillanatban is számos olyan funkcióval rendelkezik, amellyel megkönnyíti a hallgatók, illetve az oktatók munkáját, e funkciók implementálása egy esetleg, jövőbeni fejlesztés során azt gondolom, mindenképpen előnyös lenne.

# 8. Összegzés

A szakdolgozatom célja egy olyan alkalmazás elkészítése volt, amellyel megakadályozható, hogy a hallgatók a vizsga során tetszőleges programhoz hozzáférjenek a számítógépen.

Az alkalmazás fejlesztése során nagy hangsúlyt fektettem arra, hogy lehetőleg az összes, a hallgatók által használható trükköt ismerje, és megfelelő védelmet biztosítson ellenük. Ennek érdekében az alkalmazás nem csak azt figyeli, hogy az egyes programok engedélyezve vannak-e, de megvizsgálja azok elérési útját, illetve ellenőrző összeget is számol hozzájuk. De nem csupán az alkalmazásokat tartja számon, hanem azt is, hogy a megfelelő ablakon van-e a fókusz. Így ha a hallgató bármilyen okból elhagyja az ablakot, a vizsgája azonnal elégtelen lesz.

Azt gondolom, hogy ezen funkciók megfelelő védelmet jelentenek azon hallgatókkal szemben, akik valamilyen tiltott segédeszközt szeretnének használni a vizsga során. Természetesen a papíralapú puskákat az alkalmazás nem tudja kiszűrni, ezért felügyelőtanárokra mindenképp szükség van, de a hallgatók felügyeletét biztosan megkönnyíti számukra a program.

Bár a bevezetőben kitűzött célokat sikerült megvalósítanom, a mesterképzés keretében szándékom áll a fejlesztési lehetőségeknél ismertetett funkciók, illetve az időközben felmerülő új elképzelések implementálása. Lehetővé téve, hogy a hallgatók szabadon, megszorítások nélkül kifejthessék a tudásukat egy-egy kérdésről. Ezzel az alkalmazás valódi alternatívát jelentene bármilyen, hagyományos számonkéréssel szemben.

# 9. Irodalomjegyzék


[1] Tanuljunk tanulni!
http://www.tanuljunktanulni.hu/?o=memoria

[2] Oracle Java Developer Day
http://eventreg.oracle.com/webapps/events/ns/EventsDetail.jsp?p_eventId=116172&src=7038703&src=7038703&Act=49

[3] Moodle development traffic
http://blog.mudrak.name/2010/10/moodle-development-traffic-382010/

[4] Moodle 2.0 – újdonságok és érdekességek
http://moodle.jgypk.hu/file.php/23/Szeged_Moodle2_VCs_.pdf

[5] Moodle.org - Registered sites
http://moodle.org/sites/index.php?country=HU

[6] Moodle - Hallgatói kézikönyv
http://elearning.ttmk.nyme.hu/file.php/1/Segedletek/hallgatoi_kezikonyv.pdf

[7] Vég Csaba: Instant Java/Java EE/SOA I-II
Logos 2000, Debrecen, 2007., 311. oldal

[8] Create a checksum
http://www.rgagnon.com/javadetails/java-0416.html

[9] Nyékyné Gaizler Judit: Java 2 útikalauz programozóknak 5.0
ELTE TTK Hallgatói Alapítvány, Budapest, 2008., 591. oldal

[10] Csizmazia Balázs: Hálózati alkalmazások készítése
Kalibán, Budapest, 1998.

[11] Java Network Programming
http://www.ibiblio.org/java/books/jnp/javanetexamples/index.html

[12] Nyékyné Gaizler Judit: Java 2 útikalauz programozóknak 5.0
ELTE TTK Hallgatói Alapítvány, Budapest, 2008., 472. oldal