Debreceni Egyetem

Informatikai Kar

Java programozási nyelvet oktató program

Témavezető:
Nyakóné dr. Juhász Katalin
tudományos főmunkatárs

Készítette:
Ströbik László
informatikus-könyvtáros BSc.

Debrecen
2010.

Tartalomjegyzék

Bevezetés

Egyetemi tanulmányaim során sokféle témakörrel foglalkozó tantárggyal találkoztam, melyek közül legjobban azok érdekeltek, melyek a programozással kapcsolatosak voltak.

Ezen órákon ismerkedtem meg a Java programozási nyelvvel, mely rögtön megfogott egyszerűségével, ugyanis a legtöbb olyan nyelv, melynek segítségével komolyabb alkalmazásokat hozhatunk létre, bonyolultak, a Java viszont ötvözi az egyszerűséget és a nagyszerűséget, ugyanis Javaban fejleszthetünk bármilyen alkalmazást, a különböző osztálykönyvtárak adta lehetőségek bősége miatt szinte nem is létezik olyan feladat, amit a Javaban ne tudnánk megoldani.

A Java mellett szólnak olyan tulajdonságok, melyek a nyelvet a programozási nyelvek győztesévé teszi. Ezek a következők:

Egyszerű : a nyelv a C++ leegyszerűsített változata, eltűntek a mutatók, helyette az ún. referencia típus szerepel, illetve a memória felszabadítása automatikus lett, egyszerűbbé téve ezzel a programozó munkáját.

Objektum-orientált : a Java tiszta objektumorientált nyelv.

Elosztott : egy Java alkalmazás képes elérni az Interneten bárhol megtalálható URL-lel azonosított objektumot elérni.

Robosztus : vagyis hibatűrő, azaz a Java nyelven írt program a futásnál bekövetkezett nem normális esetekben is stabilan, normálisan fut, ezt a fejlesztők úgy érték el, hogy sok munkát fektettek a hibák fordításnál való kiszűrésére.

Biztonságos : a Java nyelven előállított internetes közegben felhasznált programoknál megkötések vannak, nem érhetnek el védett eszközöket vagy állományokat ezek a programok.

Semleges architektúrájú : a futtatás feltétele csupán a Java Virtuális Gép, mely a fordító által lefordított közbenső bájtkódot natív kóddá fordítja, ez sok fejlesztő szerint

kicsit lassú,ezért találták ki a Just In Time fordítókat, mely a lefordított kódokat megjegyzik és a legközelebbi hivatkozáskor már nem fordítják újra azokat.

Hordozható : mivel a futtatás feltétele csupán a Java Virtuális Gép, ezért a közbenső bájtkódot hordozhatjuk és akármilyen gépen futtathatjuk, amelyiken van JVM, az úgyis gondoskodik a natív kóddá való átalakításról.

Interpretált : a bájtkód értelmezésével az értelmező (JVM) utasításonként hozza létre a natív kódot.

Többszálú : ugyanabban az időben több programrész futhat külön szálon

Dinamikus : a Java nyelv tervezői úgy alkották meg a nyelvet, hogy az könnyen bővíthető legyen, tovább lehessen fejleszteni.

A Java nem csak az én, hanem sokan mások szívébe is belopta magát, több olyan weboldal kimutatásai alapján, mely a programozással foglalkozik, a Java szinte mindenhol az első helyen szerepel a népszerűség tekintetében, a felsőoktatásban is szinte mindenhol a C++ mellett - mely legfőbb riválisa – lehetőségünk van megismerkedni a Javaval.

Szakdolgozatomnak egy olyan oktatóprogramot elkészítését vállaltam, mely a Java-t oktatja. Két indok vezetett ezen témaválasztáshoz. A legfőbb indok az volt, hogy annyira megszeretettem a programozást a Java kapcsán, hogy késztetést éreztem arra, hogy azon emberekkel megkedveltessem, akik érdeklődnek a programozás iránt. Közrejátszott még az is, hogy azon könyvek nagy százaléka, melyek a Java-val foglalkoznak, túlságosan bonyolultan tálalják az abban való programozást.

Összegezve tehát feladatomnak egy olyan oktatóprogram elkészítését tűztem ki célul, mely kerüli az elméleties megközelítést és a gyakorlatra helyezi a hangsúlyt, folyamatosan fejleszti a felhasználó tudását, nem céloz meg célcsoportot kor vagy oktatási intézmény szerint, mindenkinek szól, aki szeretne megismerkedni a Javaban való programozással.

Bevezetés a programozásba

Az ember történelme során mindig törekedett élete megkönnyítésére, így alkotott meg nagyon sok olyan dolgot, ami időt és fáradtságot takarított meg számára. A számítógép megszületése is tulajdonképpen az ember lustaságának és leleményességének tudható be. Szeretettek volna egy olyan eszközt, amely tud adatokat rögzíteni és ezeken különböző műveleteket végrehajtani. Az, hogy a számítógépet milyen feladatok megoldására használjuk, az az emberi fantáziától és a számítógépre írt programoktól függ.

Egy számítógépes program egy olyan, a számítógép számára érthető instrukciósorozat, amely megmondja, hogy az mit csináljon, tehát, hogy az adatokkal milyen műveleteket hajtson végre ahhoz, hogy a feladat megoldásához jussunk. A programokat készítőknek viszont nemcsak arra kell törekedni, hogy a program szimplán feladatokat hajtson végre, hanem hogy azt, aki majd fogja használni - tehát a felhasználót – minél felhasználóbarátabban kiszolgálja, mert a felhasználó nem érti a gép világát.

Hasonló eset a programozók és a programozási nyelvek esetében, itt is igyekeztek a számítógép programozásának emberibbé tételével. A processzor, ami a számítógép központi egysége (CPU - central processing unit), a kettes számrendszert használja, szemben az ember által használt tízes számrendszerrel, és csak a kettes számrendszerben megfogalmazott utasításokat érti meg. A gépi kód, vagy más néven natív vagy tárgykód ilyen utasításokból áll. A programozás kezdeti szakaszában ezt használták. Mivel megértése az ember számára nehézkes, ennek kiküszöbölése érdekében alkották meg az assembly nyelvet, mely úgy működik, hogy minden utasításnak létrehoztak egy emlékeztetőt, mely az utasításra jellemző rövidítés, ezeket nevezzük mnemonikoknak. Ezeket az emlékeztetőket használva írják a programokat, viszont ha futtatni akarják, akkor egy fordítóprogramra van szükség, mert a gép nem érti az emlékeztetőket, ezt a fordítóprogramot hívjuk assemblernek. Az assembler lefordítja gépi kódra az assembly nyelven írt programot, így azt a számítógép végre tudja hajtani.

Az assembly-t alacsony szintű nyelvnek is szokták nevezni, mert a számítógéphez közel áll. Előnye, hogy az ilyen nyelven létrehozott program gyors és kis helyfoglalású, illetve, hogy bizonyos feladatok csak ilyen nyelven írt programok segítségével oldhatóak meg. Hátránya azonban, hogy a programozónak sokat kell dolgoznia és még mindig nehezen érthető. Hátránya még az is, hogy komplexebb feladatok megoldására nem alkalmas, a nehezen érthetőség mellett főként ez vezetett a magas szintű programozási nyelvek létrehozásához.

A magas szintű nyelvek közelebb állnak az ember gondolkozásához, megjelennek a változók, típusok, ciklusok, feltételek, elágazások, eljárások, függvények. Azért mondjuk, hogy közelebb állnak az emberhez, mert a nyelv elemei emberi nyelven íródtak, az egyes utasítások értelmezésére és gépi nyelvre való lefordítására a fordítóprogram hivatott. Egyes programozási nyelveknél a fordítás nem különül el a végrehajtástól, ezeknél az értelmező – amely egyfajta fordítóprogram – értelmezi és végre is hajtja azonnal az utasításokat. Minden egyes nyelvnek megvan a saját szintaktikája, azaz nyelvi szabálya, ha ezeket nem tartjuk be, akkor a fordítóprogram futása során hibával fog visszatérni, és a program nem kerül lefordításra.

Egy program szintaktikai hiba mellett rendelkezhet szemantikai hibával, ez a program logikai helyességét jelenti, tehát ha a programunk nem a kívánt feladatot hajtja végre, de a nyelv szabályainak megfelel, az szintaktikailag helyes, viszont szemantikailag helytelen. A szemantikai hibákat a fordítóprogramok nem jelzik, ezeket a programozónak kell kiszűrni. A gépi kódú programozást első generációs, az assembly nyelvet második generációs, a magas szintű nyelveket pedig harmadik generációs programnyelveknek nevezzük. A Java a harmadik generációs nyelvek közé a magas szintű nyelvekhez tartozik.

A Java története

1970-ben Ken Thomson létrehozta a B nyelvet, amely az első Unix operációs rendszer nyelve volt, de nem bizonyult elég hatékonynak, ezért Dennis Ritchie 1971-ben kifejlesztette a C nyelvet, mely nagyon sikeres lett hatékonysága és hordozhatósága miatt. 1989-ben megjelent végleges szabványosítása, majd C++ változata is, mely a C nyelvnek az objektumorientáltság tulajdonságait adta.

A Java nyelv a C-re épít, annak egy egyszerűsített változata. Története egészen 1991-ig nyúlik vissza, amikor is a Sun Microsystem egy csoportja James Gosling vezetésével egy olyan nyelvet akartak megalkotni, amely nagyon kicsi, lefordított kódja pedig nagyon hatékony, illetve hordozható tulajdonsággal rendelkezik. Ez utóbbi ötletükből jött létre a Java Virtual Machine (rövidítése JVM), azaz Java virtuális gép, amely a Java fordító által előállított közbenső, ún. bájtkódot futtatja úgy, hogy utasításonként értelmezi és alakítja át natív kóddá. Tehát mindazon számítógépen futtatható Java program, amelyen a JVM megtalálható, ezzel tették a Javat platform függetlenné, azaz géptől és operációs rendszertől függetlenné. A nyelvet Oak-nak, azaz tölgynek akarták elnevezni, mert James Gosling ablaka előtt egy gyönyörű szép tölgyfa állt, de miután rájöttek, hogy Oak nevű programozási nyelv már létezik, ezért a kedvenc kávéjuk származási helyéről, a Java szigetről nevezték el, és a nyelv szimbóluma is egy kávécsésze lett.

A Java nagyon jóra sikeredett, viszont legnagyobb sikerét az internet és a Web hozta meg, amely óriási fejlődésnek indult abban az időben, viszont a honlapok csak egyszerű információkat tartalmaztak.

1994-ben kiadták a HotJava böngészőt, amibe beleépítették a JVM-et, így a honlapokon megjelentek az animációk, videófilmek, a Java kisalkalmazások, más néven appletek futottak a felhasználó gépén.

1995-ben bemutatták ezt a fejlesztésüket, 1996-ban már a Netscape is képes volt appletek futtatására, ezután pedig minden böngészőbe belekerült ez a technika, ma már nincs olyan, amelyik ne lenne képes appletek futtatására. A nyelv dinamikus mivoltából kifolyólag

azóta nagyon sokat fejlődött, folyamatosan jelentek és jelennek meg hozzá olyan osztálykönyvtárak, melyek a technika fejlődésével a Javat alkalmassá tették és teszik a mindenkori feladatok megoldására.

Objektum-orientált paradigma

Az objektum-orientált paradigma a programozási módszerek között a harmadik, elődei a moduláris és a strukturált programozás.

A moduláris programozásban a különböző feladatok úgynevezett modulokra különülnek el, melyeket külön kell megírni, majd ezeket tesztelni, és egy egységbe olvasztani, ahol ellenőrizni kell a modulok közötti információcserét.

A strukturált programozás során a feladatokat olyan tovább nem bontható részfeladatokra bontjuk, melyekhez kidolgozzuk a megfelelő algoritmust, így a tesztelés során hamar kiderül, hogy melyik programrészben van a hiba, így könnyen javítható.

Azonban ezek a módszerek nem bizonyultak elég hatékonynak, az idő haladtával új szemléletre volt szükség, ekkor alkották meg az objektum-orientált paradigmát.

Az objektum-orientált programozás során a valós világot sokkal jobban utánozzuk, az objektumok ugyanis a valós világ lemodellezett elemei, melynek vannak tulajdonságai és viselkedései. Az objektumokat közös tulajdonságaik alapján osztályba soroljuk. Az így létrezott nyelv sokkal modulárisabb és strukturáltabb.

Az objektum-orientált paradigma nagy sikernek örvendett és örvend mai napig is, pedig keletkezését kirobbantó szoftverkrízis az 1968-as NATO konferencián fogalmazódott meg.

Minőségi szoftver jellemzői

Ha egy program fejlesztésébe belekezdünk, akkor ügyelnünk kell arra, hogy az elkészítés során olyan programot hozzunk létre, melynek végeredménye a következő tulajdonságokkal rendelkezik:

Helyesség : egy szoftver akkor helyes, ha az azt megrendelő felhasználói igényének megfelelő, tehát azt csinálja, amit elvárnak tőle.

Hibatűrés : hibatűrés alatt azt értjük, ha a szoftver nem normális esetekben, különböző hibák esetén is normálisan működik.

Karbantarthatóság : egy szoftver akkor ideális, ha az könnyen javítható.

Bővíthető : akkor beszélhetünk erről, ha a szoftver könnyen bővíthető különböző funkciókkal a felhasználó igénye szerint.

Újrafelhasználhatóság : a programozás során gyakran előfordul, hogy egy olyan feladatot kapunk, melyet már egyszer megoldottunk egy hasonló feladat kapcsán. Ilyen esetben azt a részt át lehet emelni az új feladathoz, így azzal már nem kell újból foglalkoznunk, a programok egyes részei tehát újra felhasználhatók.

Kompatibilitás : erről akkor beszélhetünk, ha a szoftverek között együttműködés van.

Felhasználóbarátság : akkor mondhatjuk ezt, ha a szoftver használata egyszerű, logikus, tájékoztatja a felhasználót a megfelelő időben, tehát maximális módon kiszolgálja a felhasználó által támasztott igényeket.

Hordozhatóság : nem függ hardver- és szoftverkörnyezettől, szabadon átvihető.

Hatékonyság : legjobb mértékben használja ki az erőforrásokat, a futási időt ezzel csökkentve.

Ellenőrizhetőség : tesztelés során az adatok és eljárások könnyedén összeállíthatóak.

Sérthetetlenség : a program akkor tesz ennek eleget, ha a különböző programhibák nem okoznak sérülést a számítógépben, programokban és az adatokban.

Szabványosság : egy programról akkor mondhatjuk, hogy szabványos, ha az működésében, külalakjában és dokumentálásában megfelel a szabványnak.

Szoftverfejlesztés

A programfejlesztés folyamata sok összetett lépésből áll. Minden lépésnek megvan a maga szerepe, mindegyik nagyon fontos és kihagyhatatlan ahhoz, hogy egy minőségi szoftvert hozzunk létre.

Feladatspecifikáció : összegyűjtjük a feladattal kapcsolatos követelményeket, minden olyan dolgot, amit meg akarunk valósítani.

Analízis : meg kell vizsgálnunk, hogy a feladat megoldható-e, ha igen, akkor a feladat nagyságát, amennyiben nagyobb lélegzetű projektről van szó, akkor a feladatok részekre kell bontani, illetve meg kell vizsgálni a feladat által követelt szükséges erőforrásokat.

Tervezés : az analízis során kialakult terv részletesebbé tétele, vagyis a programterv elkészítése.

Implementálás : vagy más néven kódolás, vagyis a programterv kódolása valamilyen programozási nyelven.

Tesztelés : az implementálás során elkészített forrásprogram tesztelése, elsőnek szintaktikailag, azaz, hogy nem követtük a programozási nyelv által előírt szabályokat és formai hibát ejtettünk. Ha ez sikeres, tehát nincs szintaktikai hiba, akkor megvizsgáljuk a programot szemantikai szemszögből is, tehát hogy a program azt

csinálja-e, ahogy az a programtervben meg van határozva, amennyiben nem, akkor javítanunk kell a szemantikai hibá(ka)t.

Dokumentálás : a dokumentáció kétféle lehet, fejlesztői és felhasználói. A fejlesztői dokumentáció a fejlesztőnek szól, saját magát vagy más fejlesztőket tájékoztat a program működéséről a programban elhelyezett megjegyzésekkel. A felhasználói dokumentálás során pedig a felhasználót kell tájékoztatnunk a program telepítésének, indításának esetleges követelményeivel és a program működésével kapcsolatban.

Javasion – Java oktatóprogram

A program bemutatása során nem a kódot szeretném elemezni, hanem a programról való elméleti tudnivalók után, a gyakorlati részen a program látható futását, illetve az egyes fontosabb háttérben zajló algoritmusokat is tárgyalom.

Javasion – a név, mely beszél

A Java az appletek megjelenésétől kezdve nagyon sikeressé és népszerűvé vált, és ahogy fejlődött a nyelv, úgy árasztotta el a világot inváziószerűen. Ez a folyamat napjainkban is zajlik, ez ihlette a programom nevét, ami a Javasion nevet kapta, mely a Java szó és az invasion, magyarul invázió egybeolvasztása, ezzel is jelezve azt, hogy a Java-t szeretné „terjeszteni”, megkedveltetni.

A program igényei

A program futtatásához szükséges a JRE, azaz a Java Runtime Environment, ez általában már minden gépen megtalálható, de a CD tartalmaz egy olyan komplett csomagot (JDK – Java Development Kit), melyben a JRE mellett megtalálható az oktatás során levő mintaprogramok és saját programok fejlesztéséhez szükséges fejlesztőkörnyezet, a NetBeans,

mely ugyancsak Javaban íródott. A program az adatokat adatbázisból nyeri ki, így szükségünk lesz egy adatbázisszerverre is, a CD-n található egy MySQL szerver telepítő is.

A program telepítése

Első lépés a MySQL szerver telepítése, mely a CD gyökérkönyvtárában levő mysql-essential-5.1.46-win32.msi fájl elindításával érhetünk el. A telepítés során válasszuk a standard telepítést és kövessük a telepítő instrukcióit. A telepítés végeztével válasszuk a konfigurálást, majd ott is az alapbeállításokat. Miután elindult az adatbázisszerver, állítsuk le, és a CD database könyvtárában levő állományokat másoljuk arra a meghajtóra, ahol az operációs rendszer található, ott is a Documents and Settings\All Users\Application Data\MySQL\MySQL Server 5.1\data\ mappába, ha ezt a telepítő során nem változtattuk meg. Ezek után indítsuk újra az adatbázisszervert, és futtassuk a programot a CD gyökérkönyvtárában levő Javasion.jar fájl elindításával.

A program által tanulható ismeretek

1.Lecke – Bevezetés a programozás világába

A felhasználó ebben a leckében azokat az alapismereteket, melyek általánosan a programozással kapcsolatosak, illetve a Java történetét, tulajdonságait és felhasználási lehetőségeit tárgyalja.

2.Lecke – Első programunk, ismerkedés a változókkal

Ez a lecke a programozást oktató anyagoknál klasszikusnak számító Hello World! feliratot kiíró programmal kezd, bemutatva általa a nyelv szintaktikáját, megtanítja a felhasználót arra, hogy hogyan használja a CD-n megtalálható fejlesztőkörnyezetet, majd rátér a változókkal kapcsolatos ismeretekre.

3. Lecke – Kifejezések, operátorok, típuskonverzió

Ebben a leckében elsőnek a kifejezésekkel foglalkoztatjuk a felhasználót, megismerheti az operátorokat, tárgyalja azok felhasználási módját, precedenciáját, majd a már jól ismert változókkal alkalmazhatja a típuskényszerítést.

4. Lecke – Vezérlési szerkezetek

A lecke során a felhasználó megtanulhatja a Javaban használt egy- és többágú szelekciókat, majd a ciklusok bemutatása következik, tárgyalva az elöltesztelős, hátultesztelős és az ún. előírt, vagy rögzített lépésszámú ciklust.

5. Lecke – Osztályok, objektumok, metódusok

A felhasználó ebben a leckében az objektum-orientált Javaval ismerkedhet meg, tárgyalva az osztályok és azok mezőinek deklarálását, osztály- és példányszintűvé tevését, konstruktorok fogalmát, annak létrehozását, és jelentőségét. Metódusok kapcsán megtanulhatja a fogalmát, mi az osztályszintű és példányszintű metódus, mi a különbség eljárás és függvény között. Ismerteti még az objektumok fogalmát, azok létrehozását, objektumok által példánymetódusok meghívását, több konstruktor létrehozását, illetve bemutatja az öröklődést is.

6. Lecke – Tömbök, keresési és rendezési algoritmusok

A lecke elmagyarázza, mik azok a tömbök, többféleképpen megmutatja, hogyan hozzuk létre azokat, illetve a benne tárolt adatokat, hogyan tudjuk lerendezni a különböző algoritmusok segítségével.

7. Lecke – String és StringBuffer osztály

A String a Javaban egy olyan osztály, melynek példányaiban olyan karakterláncokat tárolhatunk el, melyeket később már nem akarunk megváltoztatni, a lecke bemutatja hogyan hozzuk létre őket, illetve a String osztály fontosabb metódusait bemutatja. A StringBuffer osztály egy példányában hasonlóan a Stringhez, karakterláncokat tárolunk, de ezt már akkor használjuk, amikor annak értékét már meg akarjuk változtatni a későbbiekben. A lecke ezen osztály példányainak létrehozási módját is bemutatja, illetve itt is bemutatja az osztály fontosabb metódusait.

8. Lecke – Állomány- és kivételkezelés

A Javaban az adatok bekérése egy programba nem alapfeladat, ugyanis a Java io csomagja kell hozzá, ezen csomag pedig az állománykezeléssel kapcsolatos osztályokat tartalmazza. Ebben a leckében az io csomag azon osztályaival ismerkedhet meg a felhasználó, melyek segítségével be tud kérni adatokat, tud fájlból olvasni, abba írni, illetve az állománykezelés kapcsán felmerülő kivételek kezelését is megmutatja.

9. Lecke – Grafikus felület – 1. rész

A Javaban a grafikus felület létrehozására két csomag van, az egyik az AWT(Abstract Window Toolkit), mely az operációs rendszer komponenseit használja fel. Emiatt a Javat sokan nem is tartották platformfüggetlennek, mert amit megírtak adott esetben Windows operációs rendszer alatt, az egy Linuxos környezetben másképp nézett ki az AWT ezen tulajdonsága miatt. A fő ok ez volt, amiért megalkották a Swinget, mely elemeit saját maga rajzolja meg, ami igaz kicsit lassabb, mint az AWT, de ezt a mai gépeken már nem vesszük észre. A Swingben megmaradtak mindazon elemek, mint az AWT-ben, ezeken kívül tartalmaz még sok új elemet is. Éppen ezért a grafikus felület programozása során a leckék a Swing-et fogják bemutatni. Ez a lecke végigveszi a Swing fontosabb elemeit, azok által gyakran használt metódusokat.

10. Lecke – Grafikus felület – 2. rész

Ebben a leckében a felhasználó megismerkedhet az elrendezésmenedzserekkel, megtanulhatja, hogyan kell olyan programot létrehozni, melyek már eseményeket is tudnak fogadni.

11. Lecke – Appletek

Mivel a Java az Appletekkel futott be, ezért ez sem lehetett kihagyni a leckék sorából. Ez a lecke bemutatja az appletek felhasználási lehetőségeit és hogy teljes legyen a grafikus paletta, a programok elkészítése során az AWT-t használja a grafikus felülethez, hogy azt is megismerje a felhasználó.

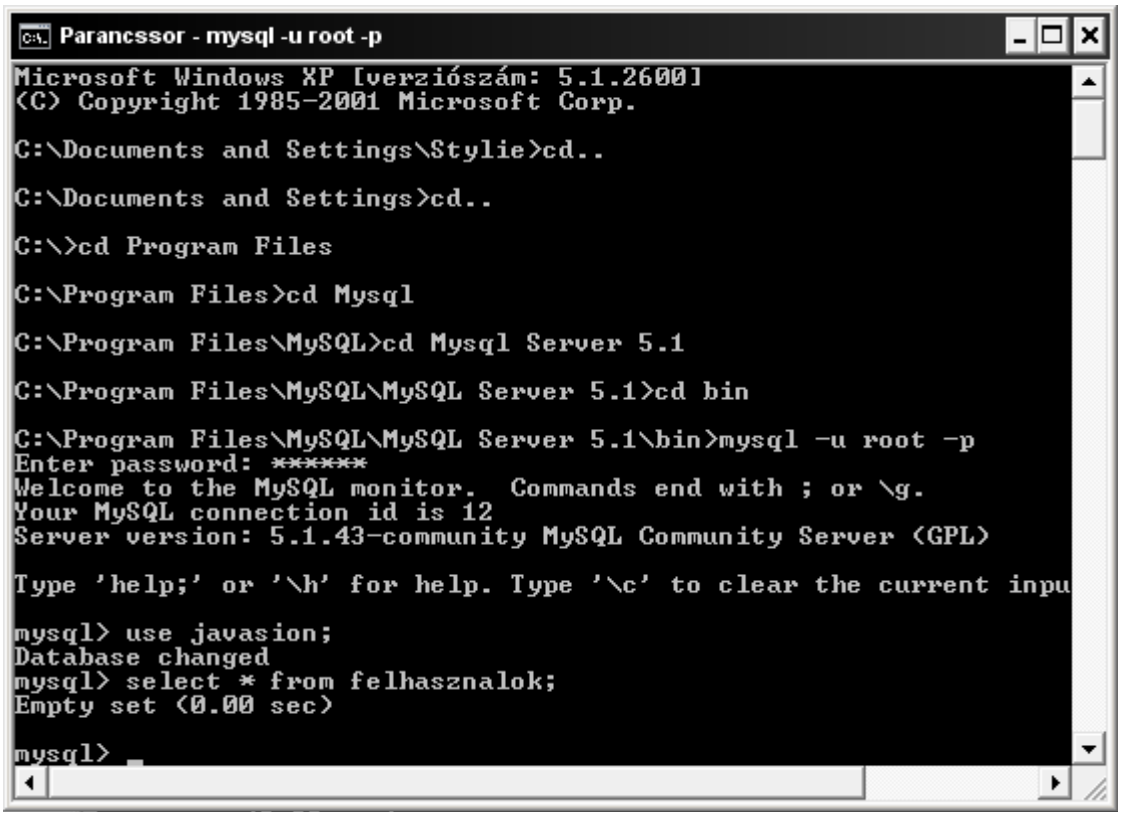
12. Lecke – Komplett alkalmazás készítése

Ez a záró lecke, ebben az eddigi megtanított ismeretek alapján egy komplett alkalmazáson elkészítését tanulhatja meg a felhasználó, ez az alkalmazás adatbázis kezelést is tartalmaz, így az is bemutatásra kerül.

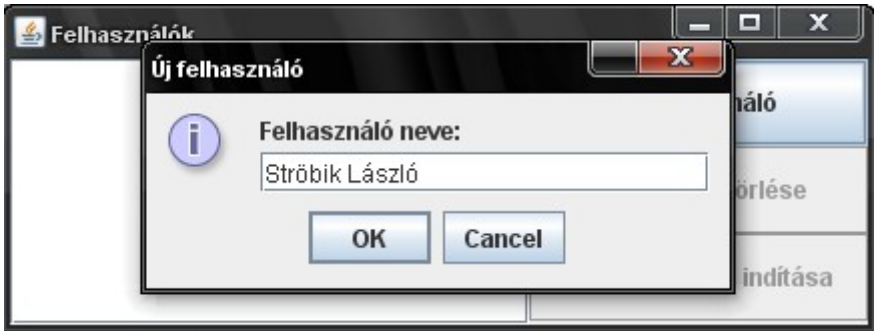
A program használata

Kezdőképernyő

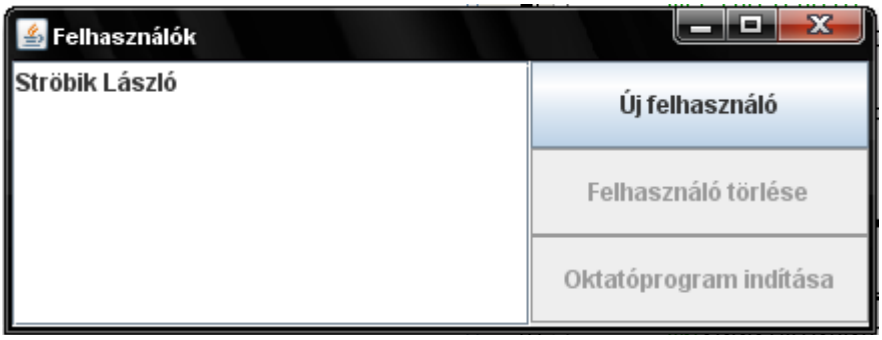
A programot úgy hoztam létre, hogy több felhasználó is dolgozhasson benne, egymástól függetlenül, egy felhasználó létrehozásakor mindössze egy nevet kell megadnunk. A felületen bal oldalt egy lista található, melybe a program az indításkor az adatbázisból betölti a felhasználókat. Jobb oldalt pedig 3 gomb, melyek közül csak az Új felhasználó aktív, ha nincs kijelölve a listából felhasználó. A program első indításakor még nem szerepel felhasználó a rendszerben. Ezt megerősíti az adatbázisból való lekérdezés is, mivel a felhasználó adatait is ott tároljuk.



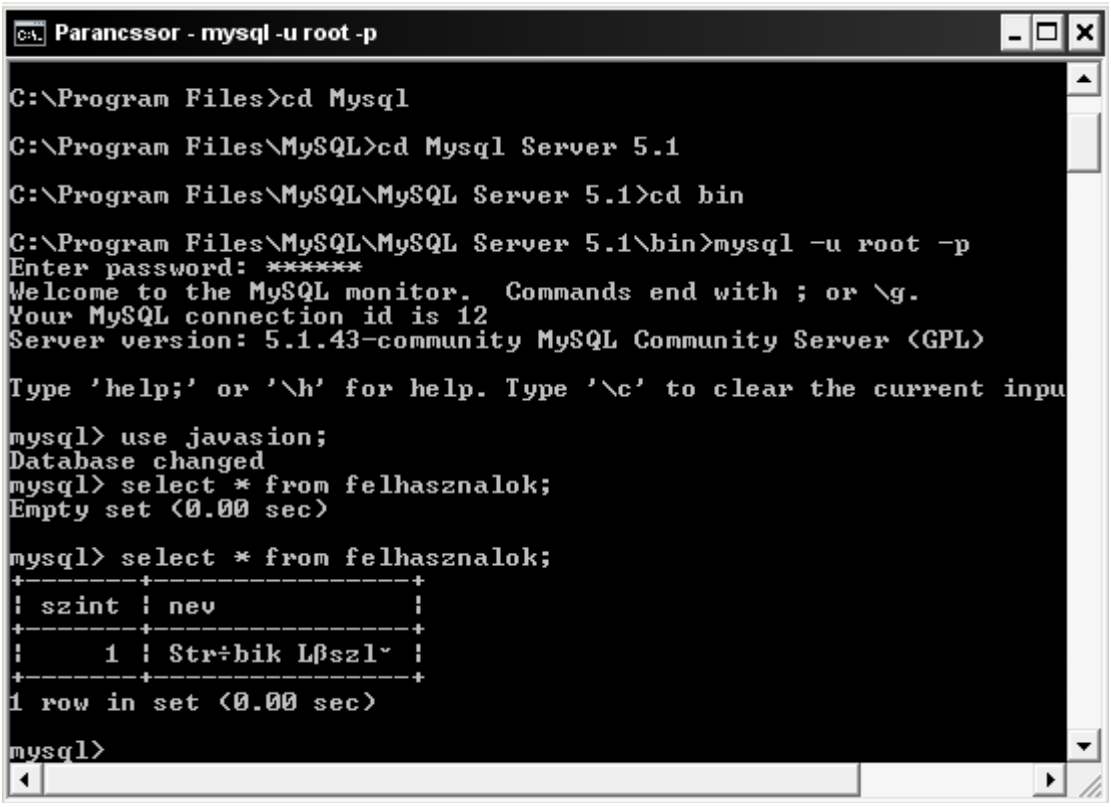
Új felhasználó gombra kattintva meg kell adnunk a nevünket:



Az Ok gombra kattintva a program máris betölti mind a listába, mind az adatbázisba a felhasználót, a listában csak a neve, az adatbázisban a szintje is fog szerepelni. A szint az, ami alapján majd a főprogramban a felhasználót az egyes leckékhez „engedjük”. Egy felhasználó szintje a „regisztráció” után automatikusan egyes.



A parancssorban leellenőrizve az adatbázisból máris visszakapjuk a megadott értéket.



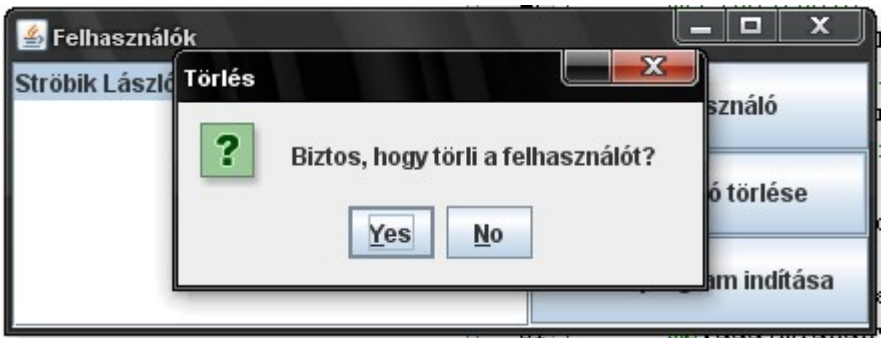
Az adatbázisban az ékezetes karakterek nem jelennek meg „normálisan”, a Java grafikus felületén viszont ez nem okoz problémát.

Új felhasználó megadásánál két gyakran előforduló eset lett kezelve, az egyik, ha nem ad meg a felhasználó semmit, akkor figyelmeztetjük erre:

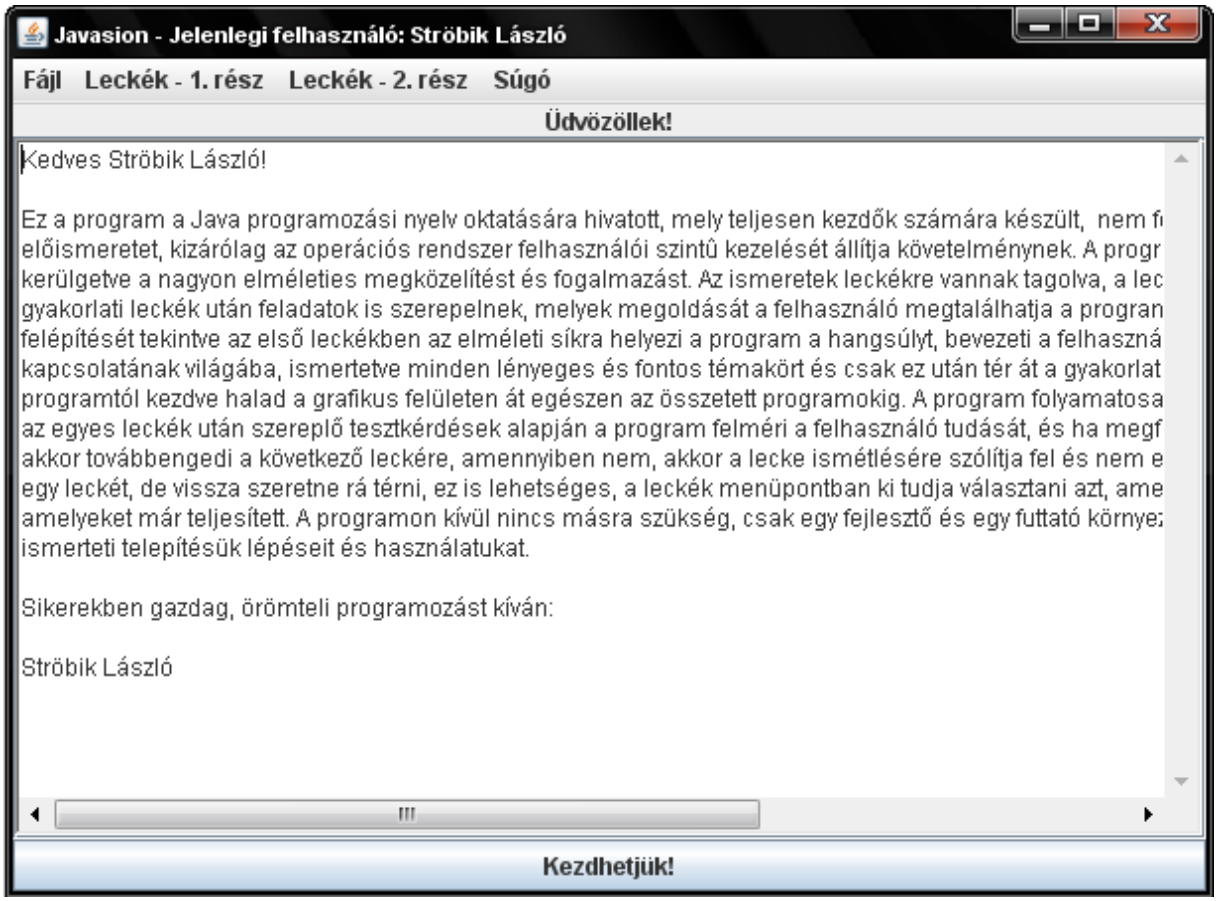
Másik eset pedig az, mikor egy olyan felhasználói nevet akar megadni, mely már létezik. Ez azért fontos, mert amikor majd a programban sikeres teljesítés esetén léptetni akarjuk a felhasználó szintjét, hogy a következő leckére lépjen, akkor mindkét felhasználó szintjét növelné. Ezt úgy oldottam meg, hogy egy ellenőrzés fut végig a felhasználók táblán, ha van már olyan nevű felhasználó, amit megadtak, akkor igaz értéket ad vissza, ha nincs, akkor hamisat. Ebben az esetben is figyelmeztetjük a felhasználót:

Mindkét esetben a felhasználó megadása dialógushoz irányítjuk vissza a felhasználót.

Ha kijelölünk a listában egy felhasználót, akkor aktívvá válik a Törlés és az Oktatóprogram indítása című gomb. A törlésre kattintva egy megerősítést váró dialógusablak jelenik meg, melyben az Ok gombra kattintva kitörölhetjük a felhasználót, a Mégsem megnyomása esetén pedig visszakerülünk az alapképernyőre.



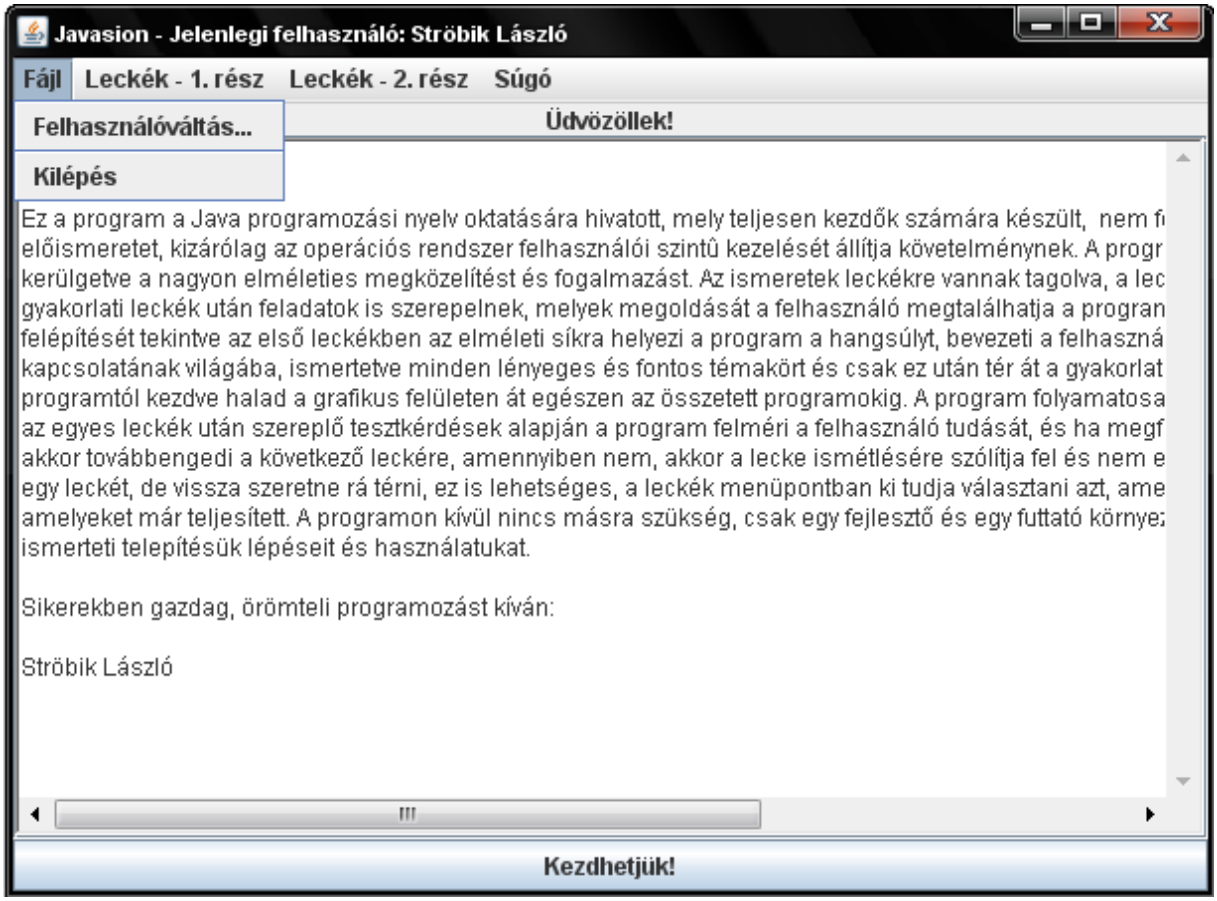
A felhasználót kijelölve a listában, ha az Oktatóprogram indítása nevű gombra rákattintunk, akkor bejön maga a főprogram, melynek címsorában feltüntetjük a program nevét, illetve a programot jelenleg használó nevét. Ezen kívül a felhasználót nevét szólítva üdvözli, és egy rövid pár sorban összefoglalja a program működését. A kezdhetjük gombra kattintva rögtön az első lecke elolvasását tölti be.



Menük:

A program egyes funkciói csoportosítva vannak menükre, négy menü található: Fájl menü, melyben felhasználót válthatunk és kiléphetünk, egy menü a leckék első részére, egy menü a leckék másik részére, egy súgó menü, melyben a Használat és a Névjegy menüpontot találjuk.

Fájl menü:

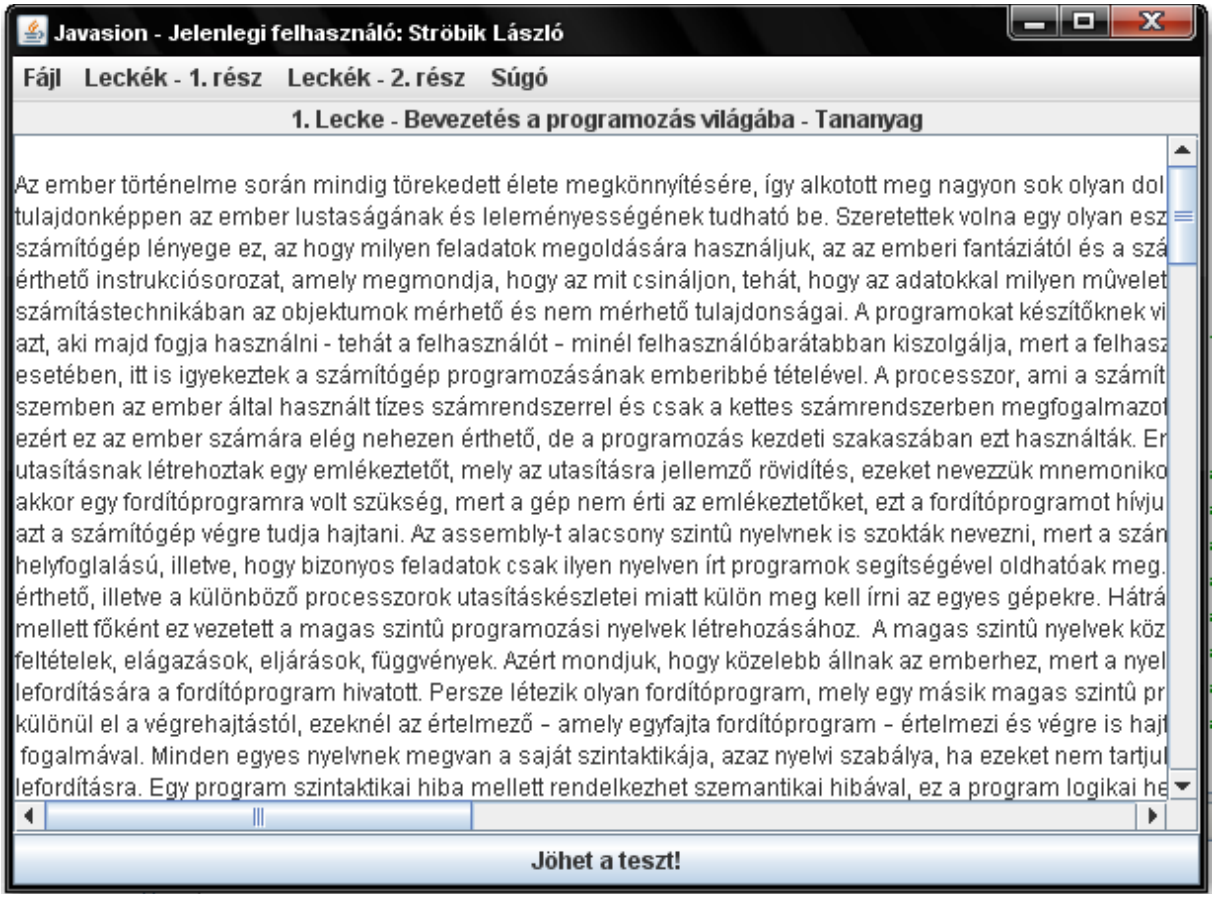


A fájl menüben két menüpont található, az első a Felhasználóváltás, melyre rákattintva visszalépünk a felhasználóválasztó képernyőhöz. A kilépés menüpontra rákattintva pedig egy dialógusablak jön be, mely megerősítést kér a kilépésről, ugyanezen folyamat megy végbe az ablak bezáró gombjára való rákattintás esetén is.

Leckék menü:

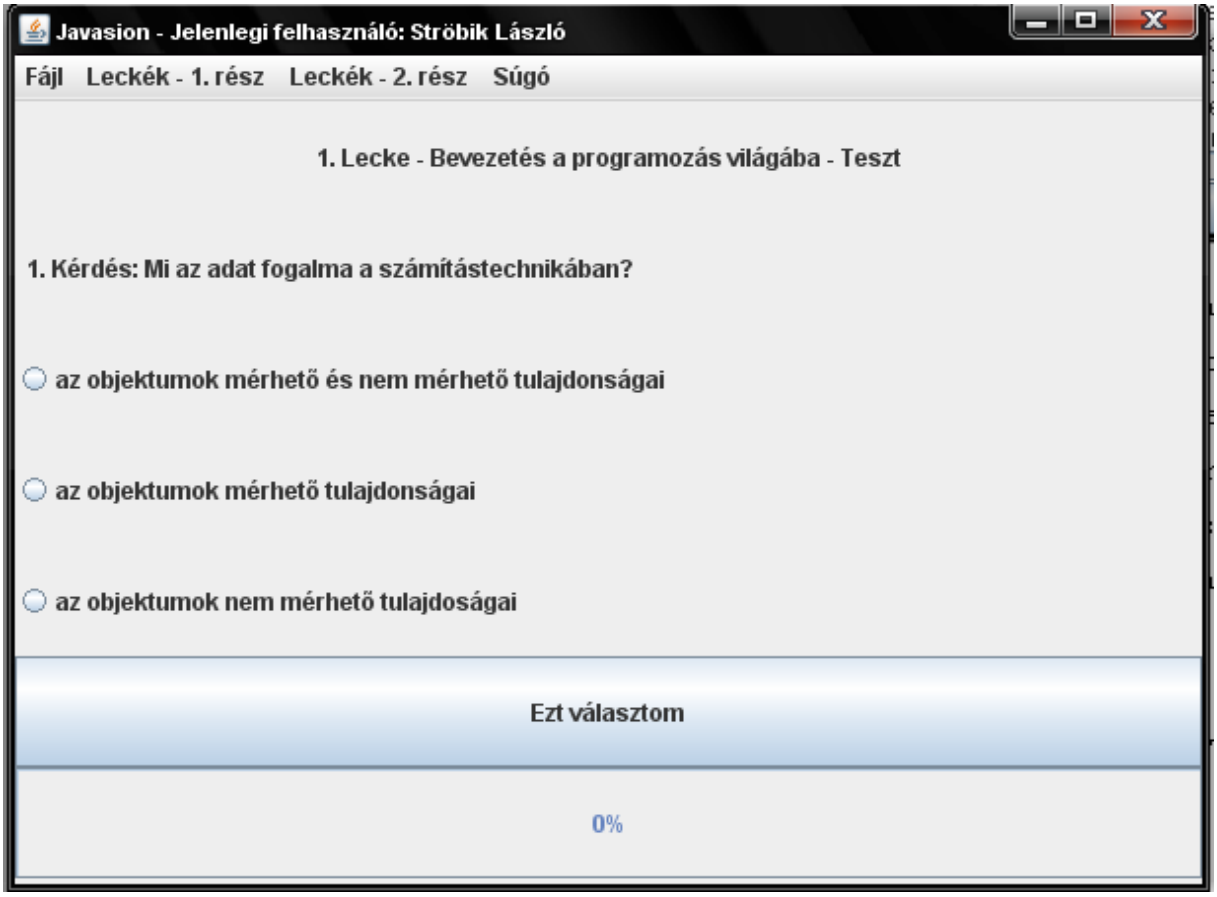
Azért, hogy a menübe való belépéskor ne legyen túlságosan hosszú a lista, a leckék részt két menüre szedtem, mind a kettő hat-hat leckét tartalmaz, melyek maguk is menük, tehát további menüpontokat tartalmaznak, ezek a Lecke elolvasása, a Teszt, és a Feladatok. A lecke elolvasása menüpontot kiválasztva a lecke szöveges magyarázatát olvashatjuk el. A Teszt kiválasztása során a felhasználó elméleti és gyakorlati tudását tesztelem. A gyakorlati részben a leckéhez kapcsolódó feladatok megoldására ösztönzöm a felhasználót. Az első lecke elméleti mivoltából kifolyólag Feladatok menüpontot nem tartalmaz, illetve kivételt képez még az utolsó lecke is, mely egy komplett alkalmazás fejlesztését követi nyomon, így abból nem kér számon.

Olvasás menüpont:



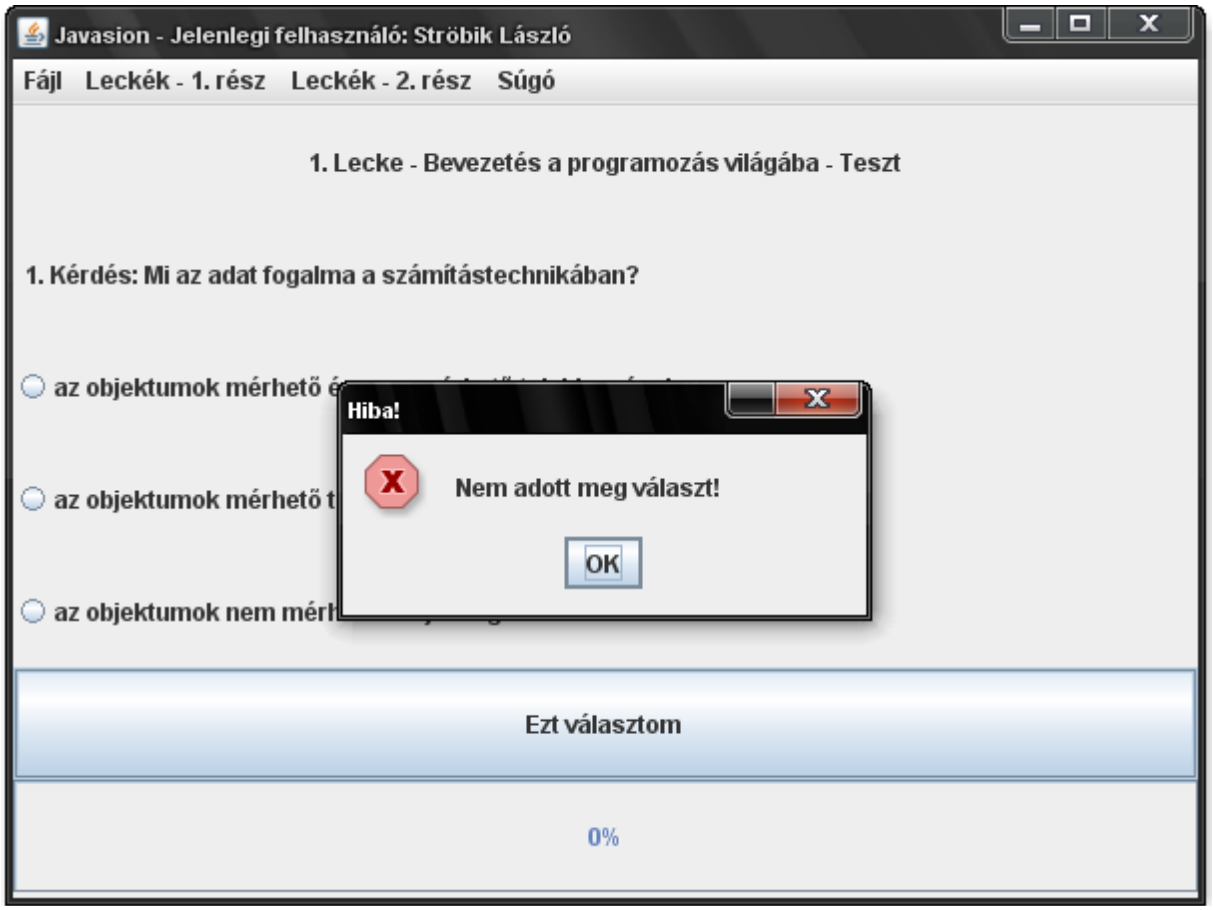
Minden lecke során találkozhatunk olvasás menüponttal, mely az aktuális lecke szöveges részét tölti be. A programban az *Elolvasas* osztály felel azért, hogy betöltse azt a szöveget az adatbázisból, melyet a felhasználó kiválasztott. Ez az osztály a swing csomagból a JPanel osztály leszármazottja. A JPanel egy összefogó konténer, melyet egy keretbe lehet belerakni, és ami összetartja a benne levő komponenseket. Ennek az osztálynak az elrendezésmenedzsere BorderLayout, azaz határ menti elrendezést biztosít, kirakhatunk komponenseket észak, déli, keleti és nyugati irányba, illetve középre. Északi irányban egy szöveges tartalmat hordozó JLabel komponens található, mely mindig az aktuális lecke számát, címét, és a tananyag szöveget fogja tartalmazni. Középen egy JTextArea komponens található, melybe az adatbázisból beemeljük a leckéhez tartozó szöveget. Déli irányban pedig egy gomb található, melynek segítségével az aktuális leckéhez tartozó tesztre ugorhatunk.

Teszt menüpont:

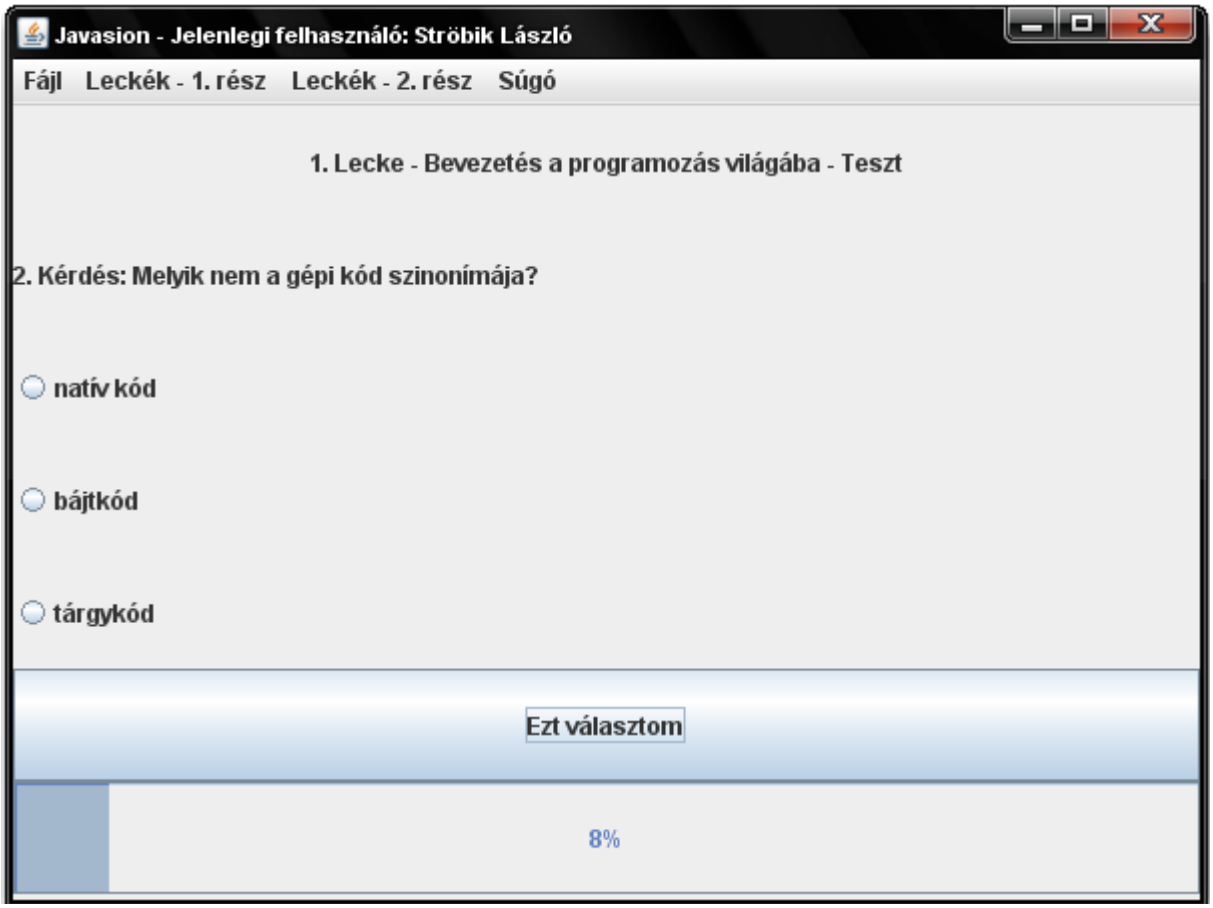


A *Teszt* osztály elrendezése egy kicsit eltérő, mint az eddigiek, ugyanis a GridLayout elrendezésmenedzsert használja, melynek segítségével rácsos kialakítást hozhatunk létre a megadott sorok és oszlopok száma alapján. Soroknak 7, oszlopoknak 0 lett megadva, így jött ki az alábbi felosztás. Az első sorban egy már ismerős JLabel komponens van, ami az *Elolvasas* osztálybeli társához hasonlóan az aktuális lecke számát, nevét, de most nem a Tananyag szót pluszban, hanem a Teszt szót tartalmazza. Alatta ismét egy JLabel, mely az aktuális kérdés sorszámát, illetve az aktuális kérdést fogja tartalmazni. Ezek után következik egy három JRadioButton, azaz rádiógombból álló ButtonGroup, azaz gombcsoport, melyeket azért foglaltunk egy gombcsoportba, hogy csak egyet lehessen kiválasztani. Ezeknek a rádiógomboknak az értéke mindig változni fog, az aktuális válaszokat tölti majd be ide a

program az adatbázisból. Az Ezt választom feliratú gomb segítségével lehet elküldeni választ. Amennyiben nem választották ki semelyik rádiógombot, akkor a program felbukkanó dialógusablakban figyelmezteti a felhasználót arról.

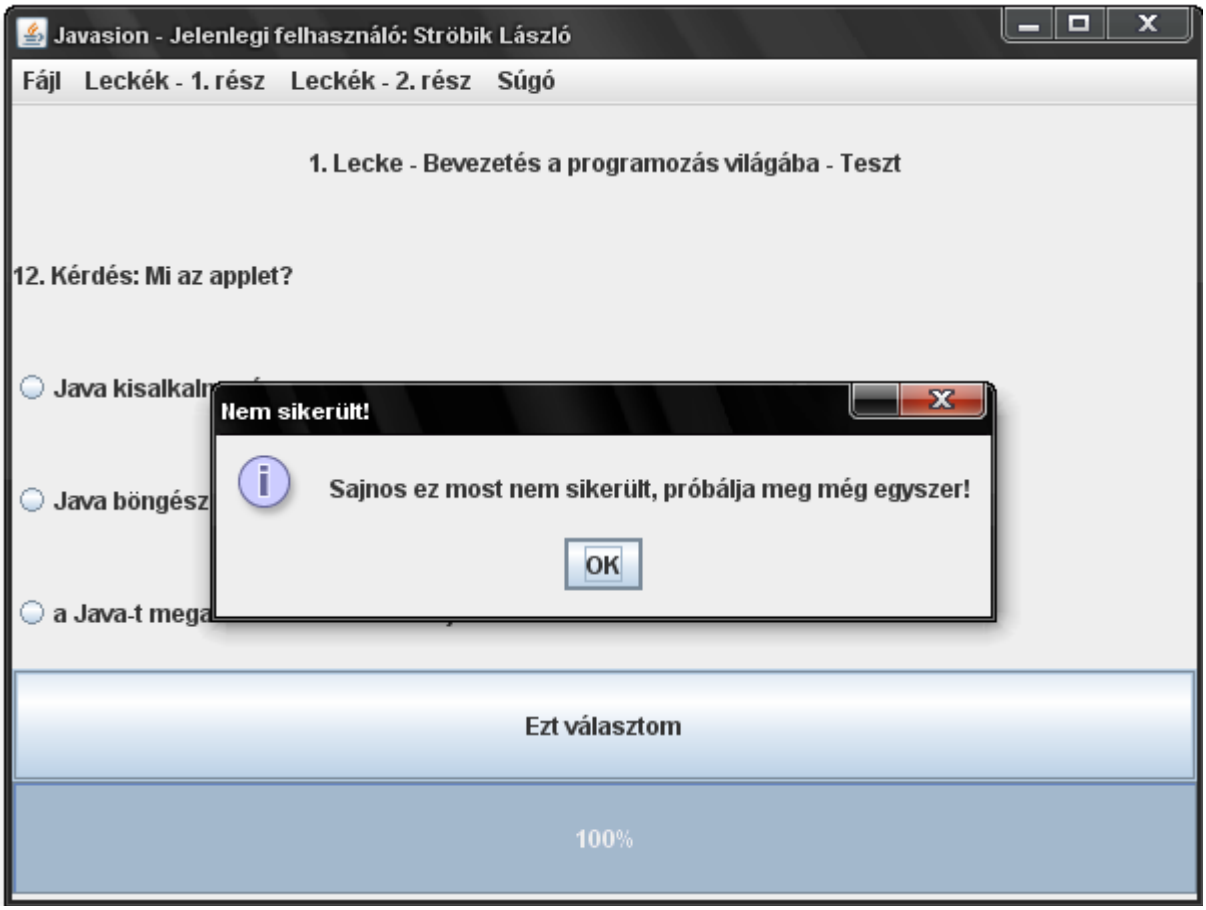


Erre azért van szükség, ugyanis ezen osztály példányának meghívásakor a helyes válaszokat egy tömbbe tölti bele, illetve a felhasználó válaszainak is létrehoz egy akkora tömböt, mint a helyes válaszoké, ebbe pedig majd az elküldés során kiválasztott válaszokat rakja bele folyamatosan. A teszt végén egy kiértékelés fut le, ahol a két tömböt összehasonlítja a program, és ha valahol nem definiált értéket talál, akkor kivétel fog keletkezni, és a program leáll.



A gomb alatt egy folyamatmutató, melynek Swingbeli komponensneve JProgressBar, a feladata az lesz, hogy mutassa, hogy hány százaléknál jár a felhasználó a teszt teljesítésében. Ha sikeres volt a válasz elküldése, akkor nem kapunk hibaüzenetet, betöltődik a következő kérdés a válaszokkal együtt, és a folyamatszámláló értéke is változik.

Ha végighaladt a felhasználó a teszten, akkor a végén a program kiértékeli a teljesítményét, ha ez 60%-on alul maradt, akkor nem engedi a felhasználót tovább a következő leckéről, erről értesítést is kap.



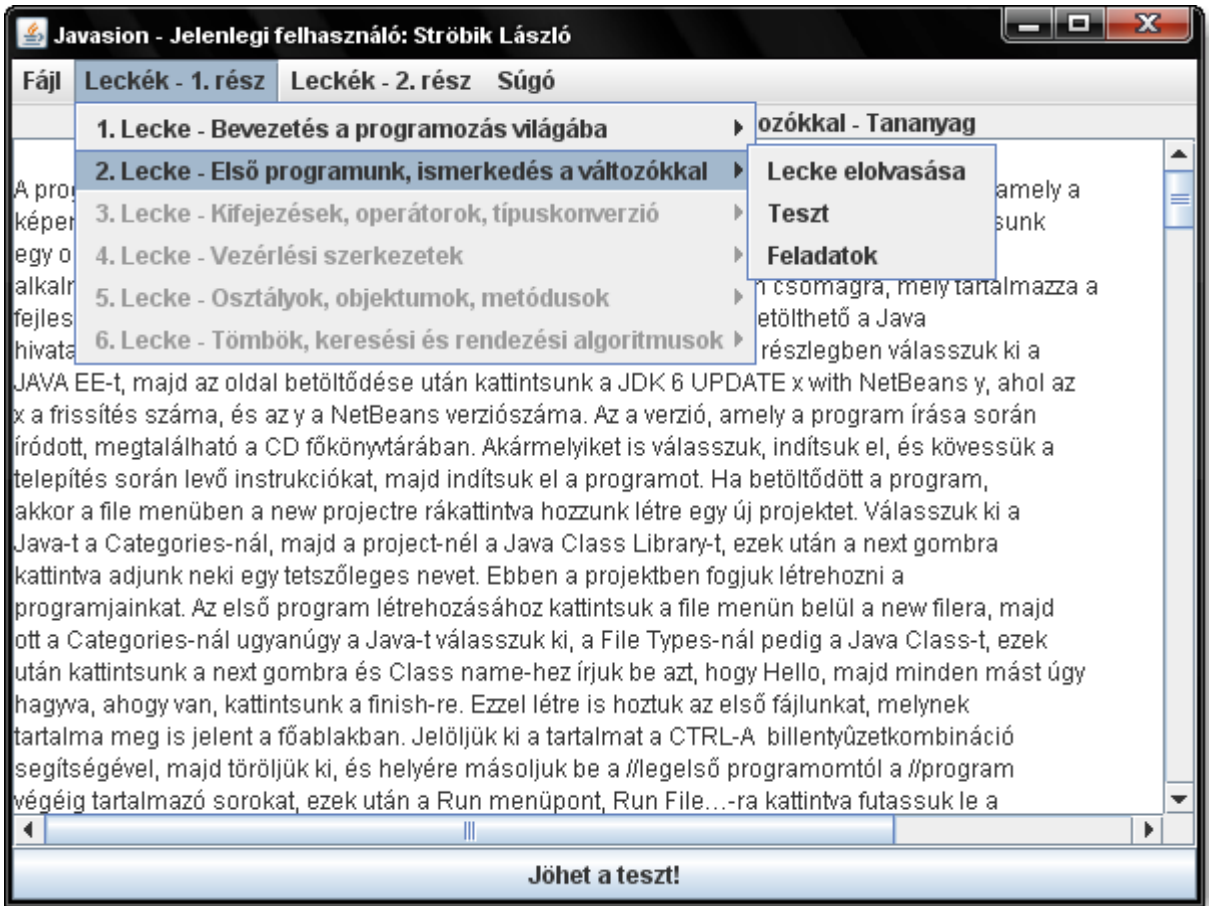
Sikeres lecketeljesítés esetén, a felhasználónak gratulál, és amennyiben a szintje akkora, mint az aktuális szint, akkor elérhetővé teszi a következő leckét, de előtte felszólítja a gyakorlati részben megadott feladatok megoldására és ahhoz a részhez továbbítja, kivéve az első leckét, ahol nincs gyakorlati rész. A programban azonban van lehetőség szorgalomból való ismétlésre is, amikor is bármelyik lecke menüből be lehet tölteni az elérhető leckéknek az elolvasás, teszt vagy feladat részét. Ha a tesztet a felhasználó betölti és sikeresen teljesíti azt, akkor ahhoz a leckéhez továbbítja a program, ahol eredetileg járt.

A program abban az esetben teszi lehetővé a következő szintet, ha a 60%-ot eléri a felhasználó és a szintje az megegyezik a lecke számával. Ugyanis ha a felhasználó már teljesített egy leckét és visszalép egy másik tesztre sikeresen teljesítve azt, akkor a program a

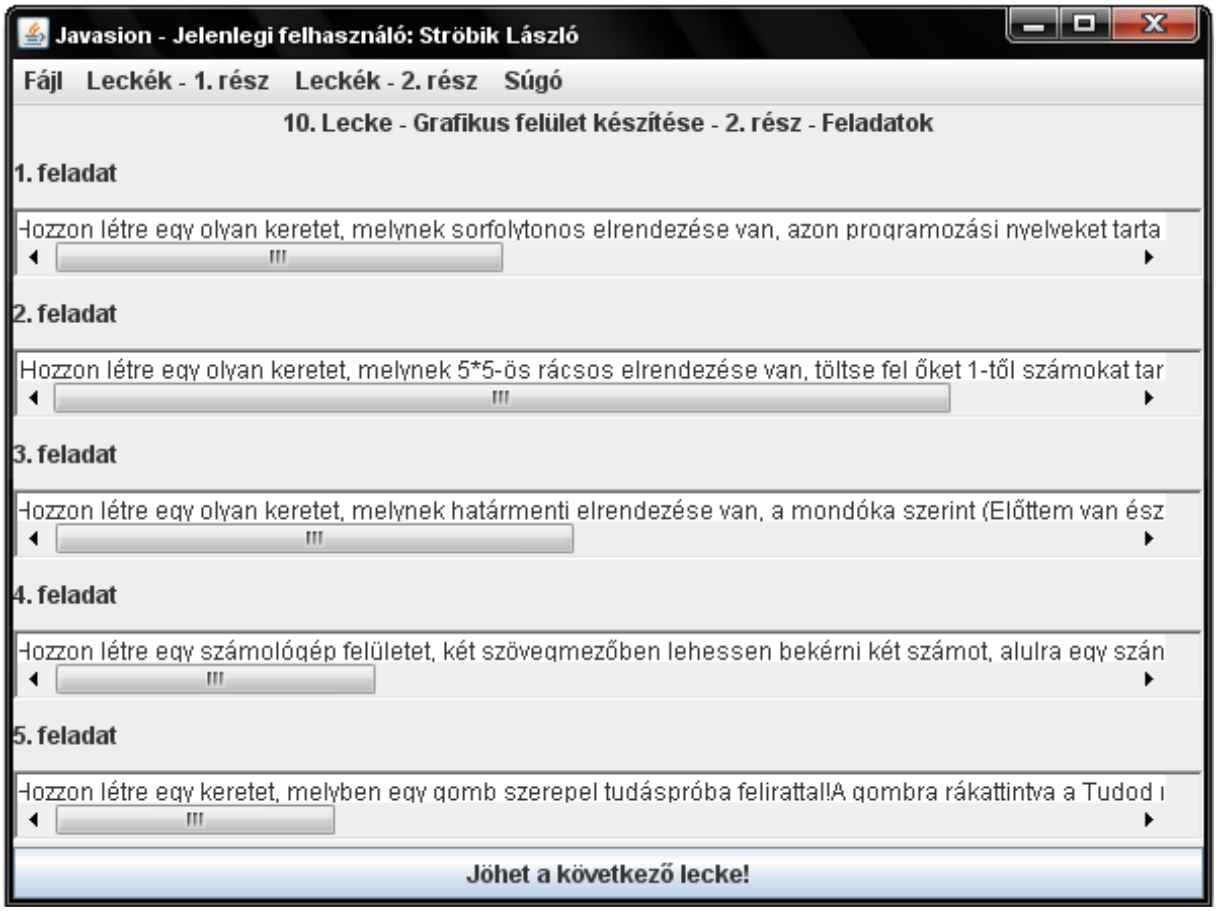
feltétel nélkül úgy futna le, hogy ha a felhasználó 5.-ös szinten van, és megcsinálja az 1.-es tesztet újra, akkor a 6. szintre lépne és ez nem megengedhető.

Maga az engedélyezés úgy működik, hogy mindkét lecke menü elemei egy tömbben vannak eltárolva, ahol alap esetben le vannak tiltva. A program indításakor a felhasználó neve alapján a program beolvassa az adatbázisból a szintet és addig az elemig feloldja a menü elemeit, melynek száma megegyezik a felhasználó szintjével. Sikeres teszt esetén is ugyanezen algoritmust megvalósító metódust hívjuk meg, csak előtte megváltoztatjuk az adatbázisban a szint értékét.

Így az első lecke sikeres teljesítése esetén már elérhetővé válik a második lecke, és annak menüpontjai:

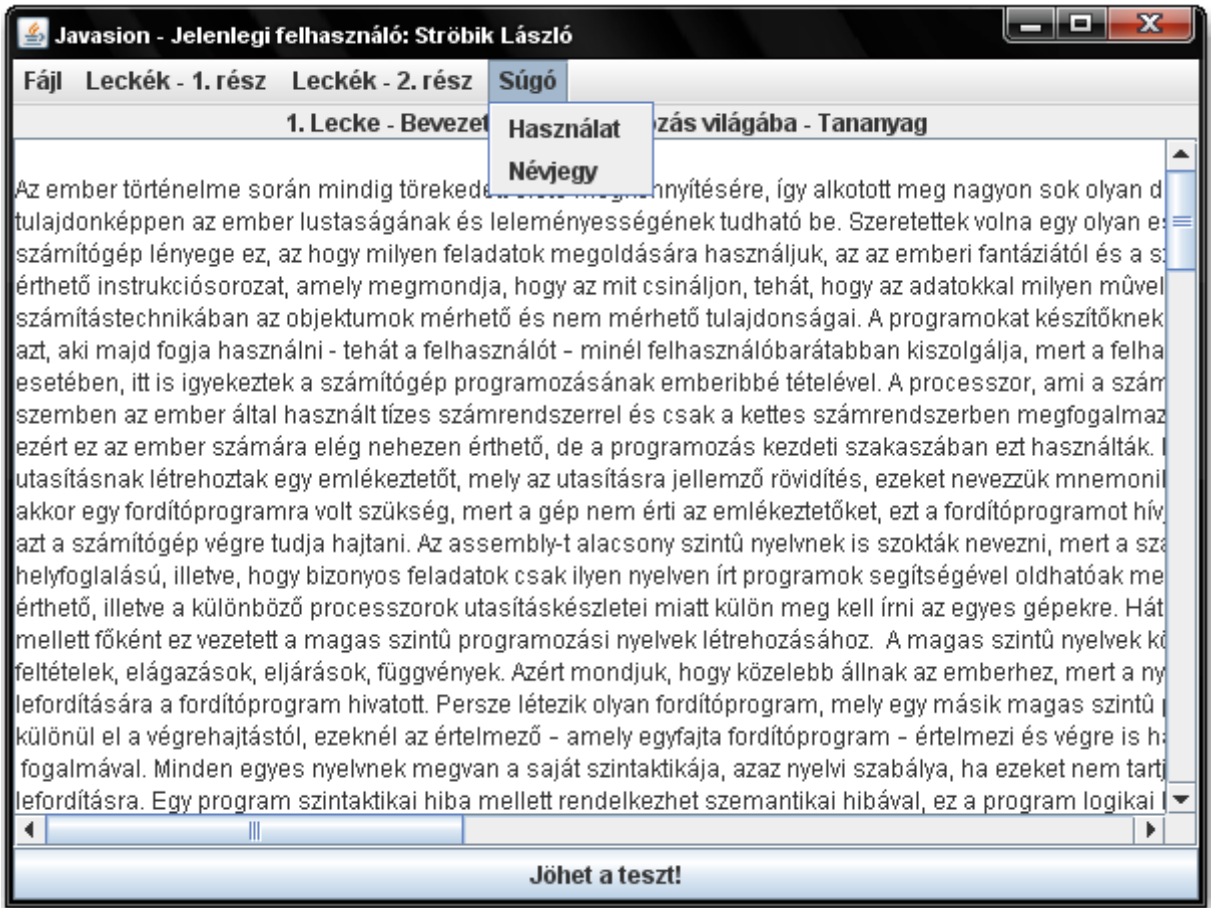


A második és további leckék során találunk gyakorlati részeket is, ahol a felhasználót az addig tanultak alapján házi feladat elkészítésére ösztönzi, melyet megtalál a CD-n, amennyiben elolvasta a legelején a program tájékoztatást. Ezen rész során hasonlóan a többi részhez egy olyan Jlabel jelenik meg, mely a lecke aktuális számát, nevét és a Feladatok szöveget. Ez a BorderLayout elrendezésmenedzser által felkínált északi helyen helyezkedik el. Középen egy panel, melybe a program annyiszor generál egy a feladat számának megjelenítésére használható Jlabel-t, illetve alatta egy TextArea objektumban a feladat szövegét, ahány feladat szerepel az adatbázisban.

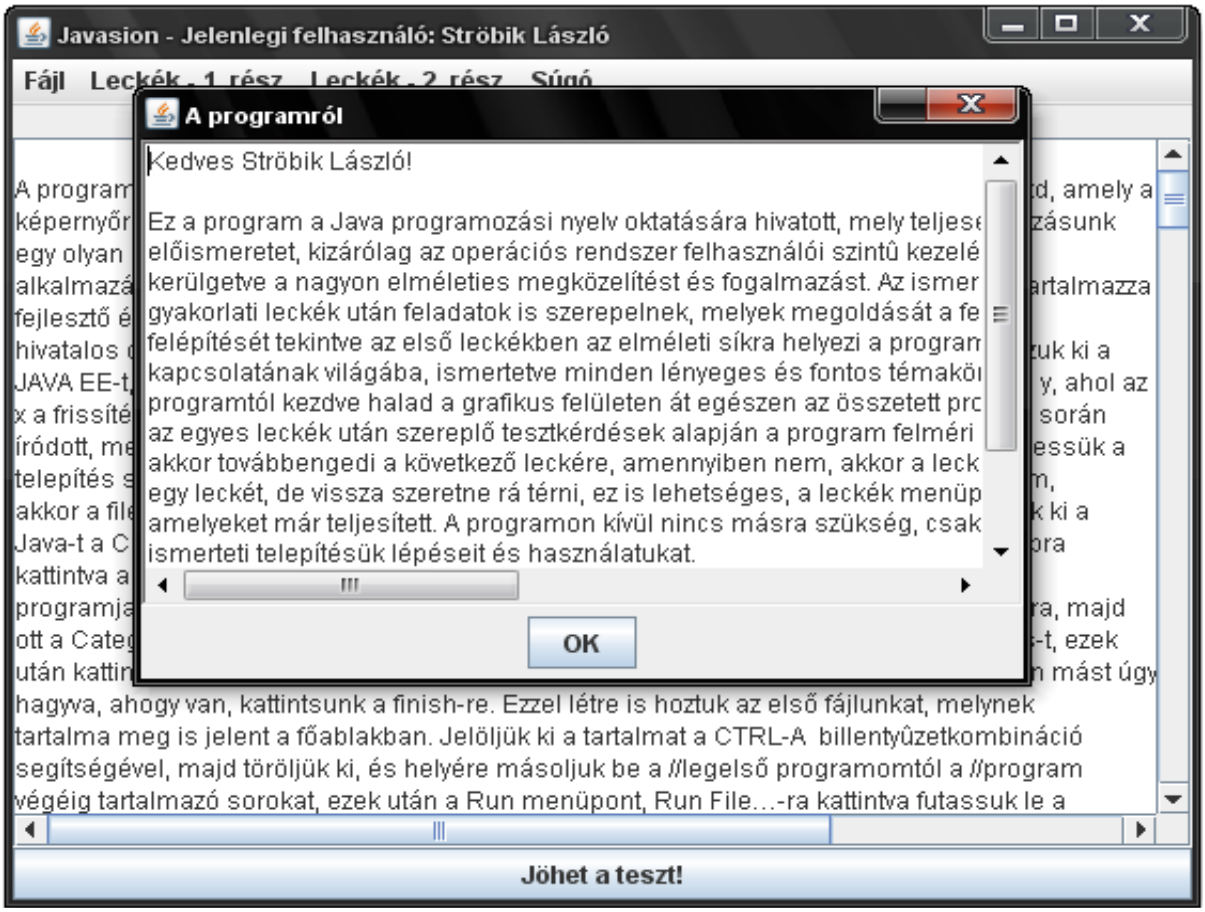


Súgó menü:

Ebben a menüben egy használatot és egy névjegyet megjelenítő menüpont van.



A használatra kattintva egy dialógusablak pattan fel, melyben ugyanazt a szöveget jelenítjük meg, ami a felhasználó első programindításánál jön be.

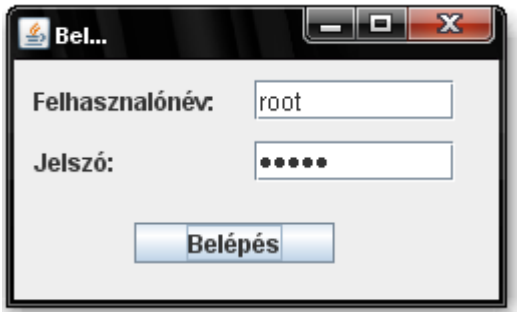


A névjegyben pedig egy olyan ablak jelenik meg, ahol egy kis animációt tartalmazó kép mellett a programról jelenít meg információkat.

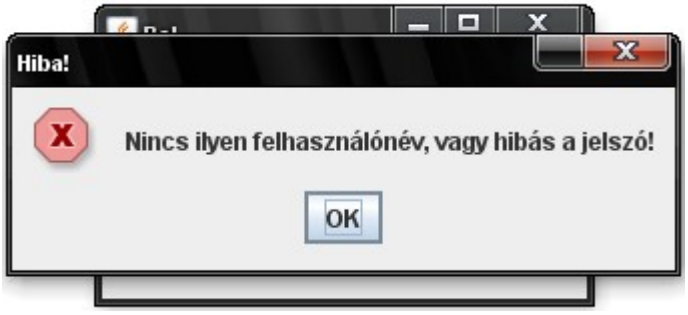
Feltesszük az i-re a pontot...

A program utolsó leckéjében egy komplett alkalmazás elkészítését mutatja be a felhasználónak, mely olyan ismereteket használ, mely az eddig leckék során ismertetve lettek. A program egy könyvtári alkalmazás, melyben dokumentumokat vihetünk be az adatbázisba, onnan lekérdezhetjük az adatait, módosíthatjuk őket, illetve ugyanezen műveleteket elvégezhetjük az olvasók tekintetében.

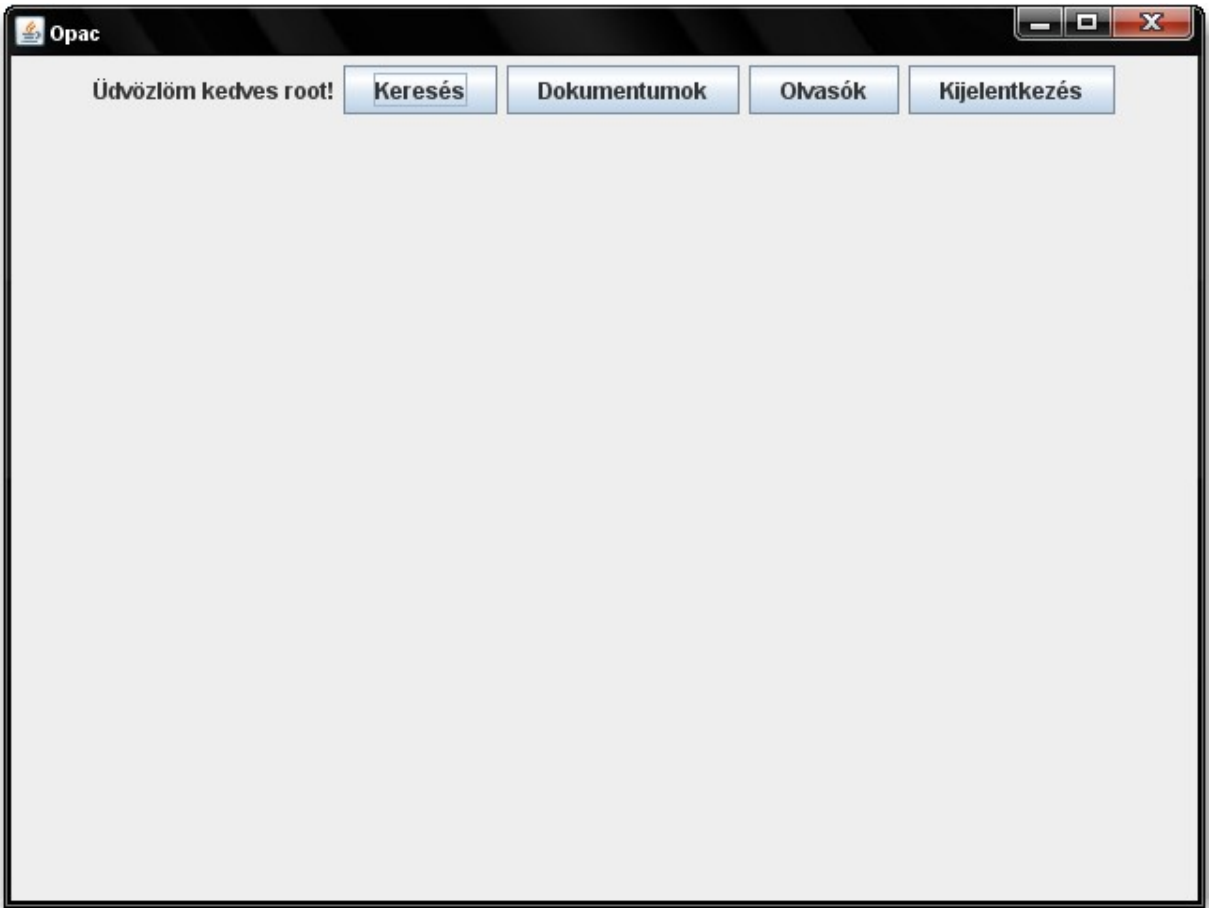
Ezen programba be tudunk jelentkezni az adatbázisba a felhasználónév és jelszó segítségével.



Amennyiben a jelszót elrontjuk úgy a program a kivételkezelést hajt végre, és egy „Nincs ilyen felhasználónév, vagy hibás a jelszó!” szöveget tartalmazó dialógusablak felugrását eredményezi.

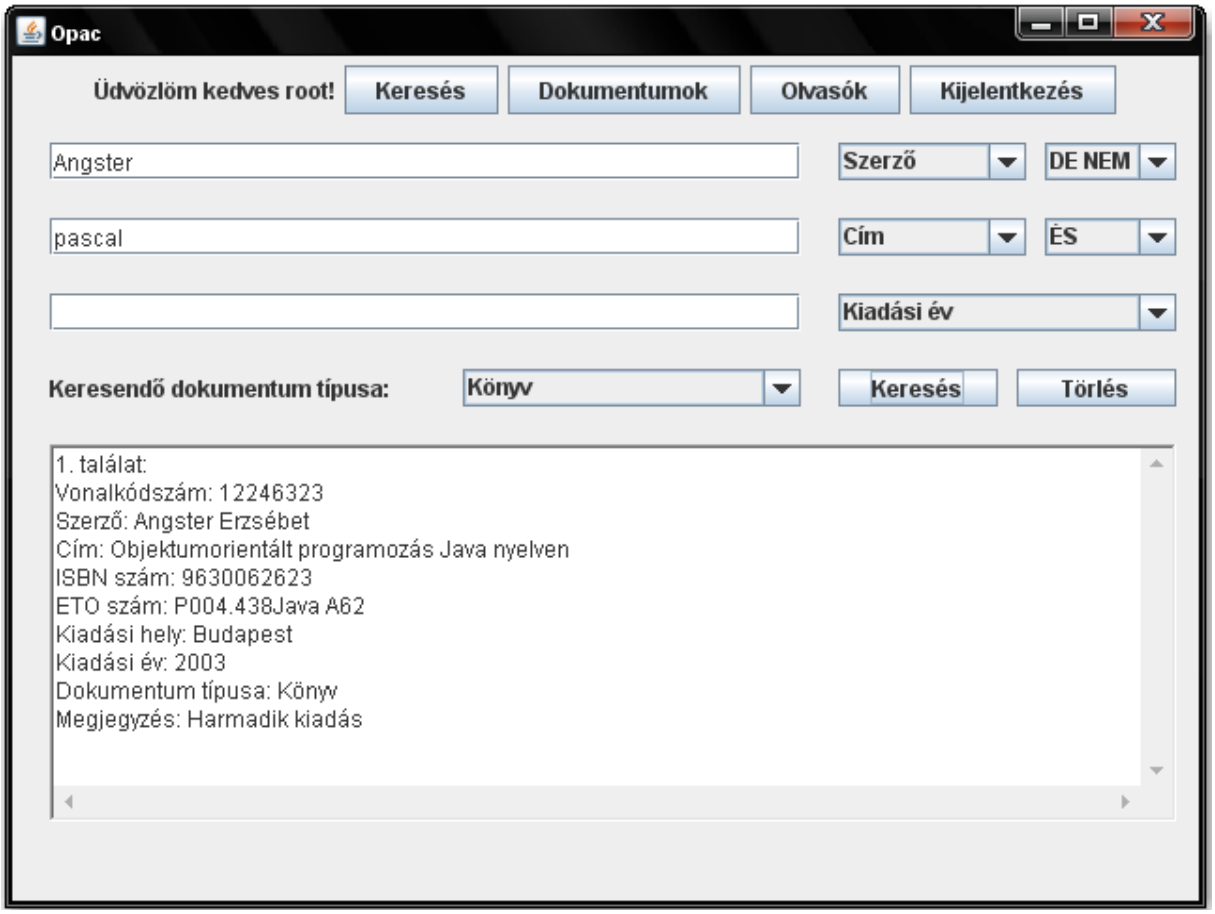


Ha sikeres a bejelentkezés, akkor ezt egy dialógusablak jelzi, és a főprogramba továbbít, ahol lehetőség van kiválasztani a megfelelő panel betöltését.

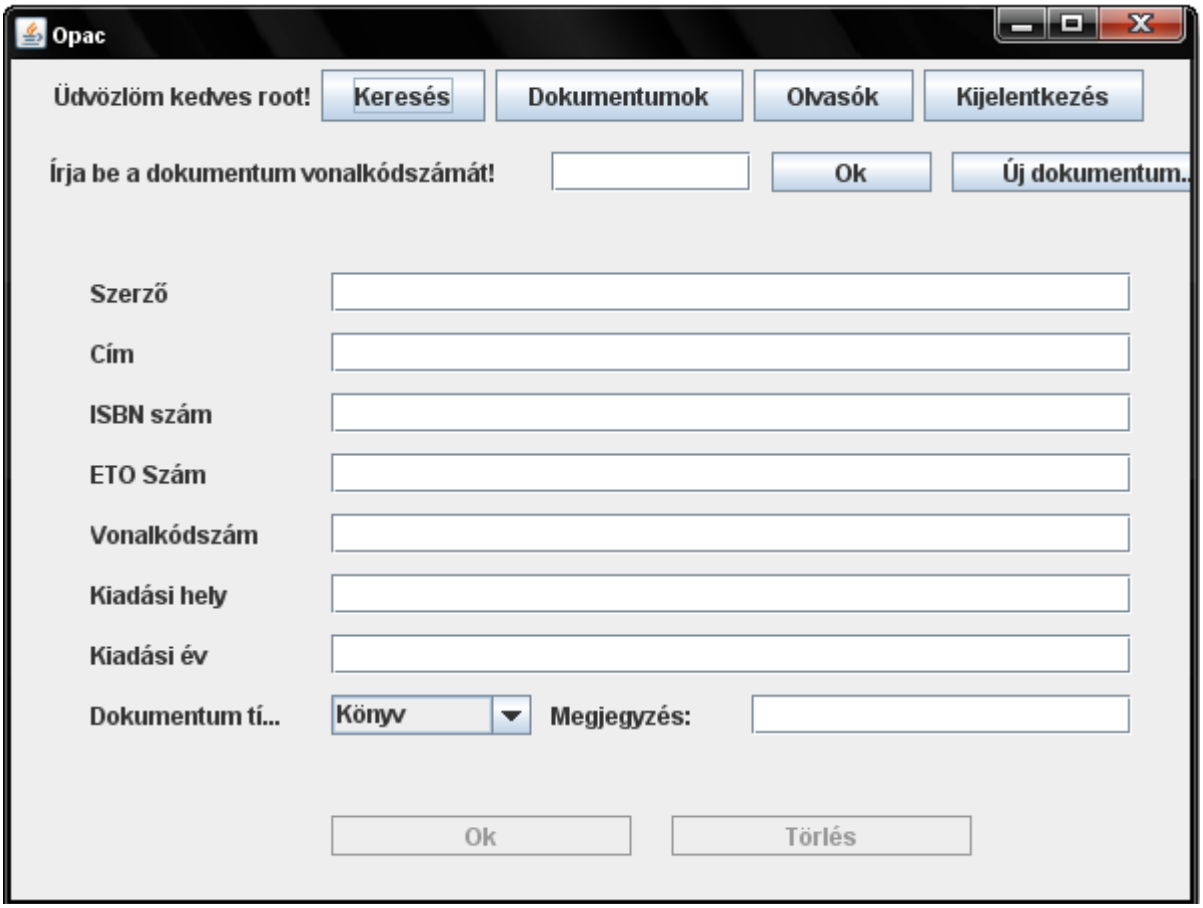


A keresés gombra rákattintva egy olyan panel jön be a keret alsó részébe – ahova majd a különböző gombok kiválasztása esetén bejönnek a tartalmak – ahol keresni tudunk az adatbázisban eltárolt dokumentumok között különböző feltételek megadásával. Ki lehet választani, hogy milyen oszlopok között keressen a program, milyen logikai operátorokat használjon a lekérdezések során, illetve a dokumentum típusát is ki lehet választani. A keresés gomb megnyomásával elindíthatjuk a lekérdezést, mely azt a felhasználónevet és jelszót fogja használni, melyet a programba való bejelentkezéskor adunk meg. Ennek megvalósítására azért volt szükség, mert ha a belépett felhasználó nem az Adminisztrátor, mint jelen esetben, hanem egy olyan, akinek nincsen jogosultsága például az adatbázisba való bevitelre, akkor nem fog tudni dokumentumot vagy olvasót bevinni. A törlés gombbal alaphelyzetbe lehet

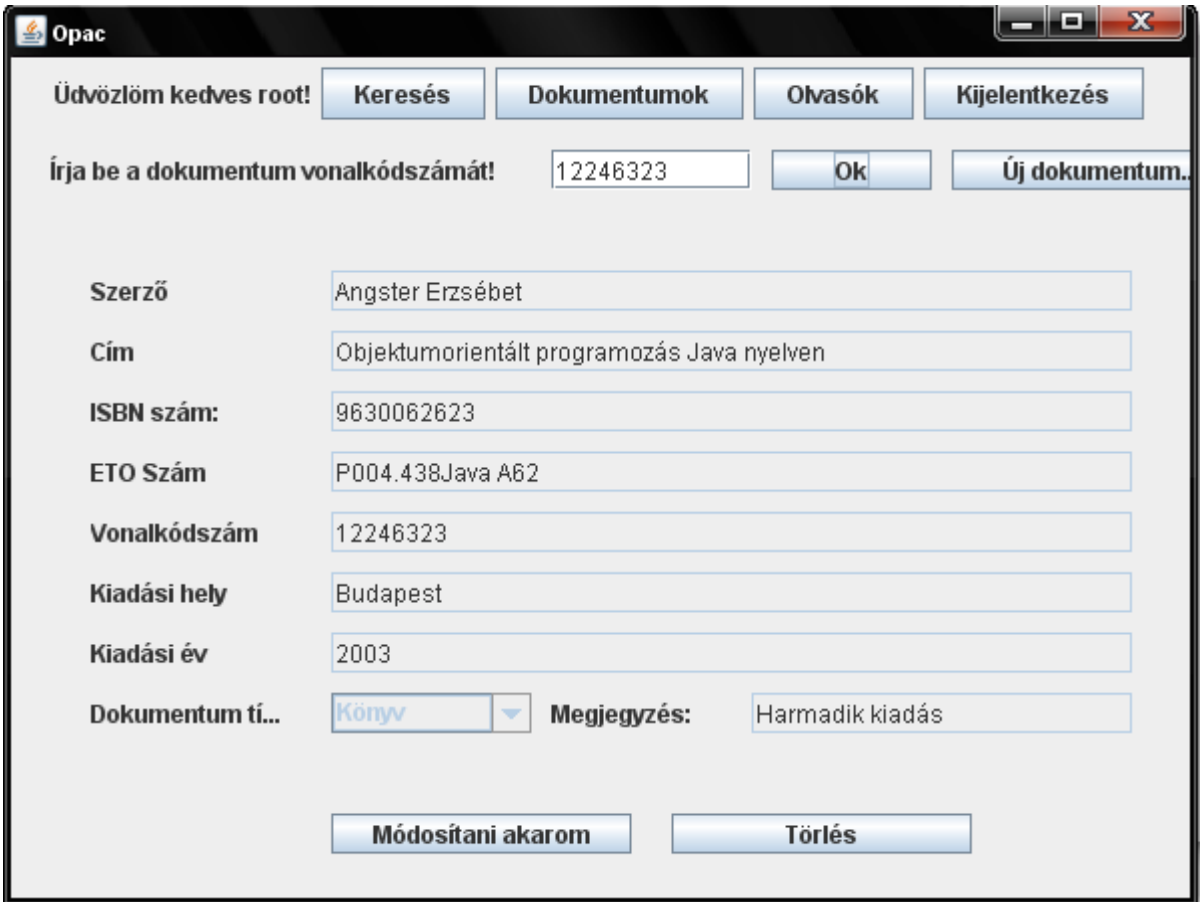
állítani a keresőkérdést kialakító komponensek tartalmát. A szövegmezőbe a keresés gomb megnyomásakor betöltődnek a keresés alapján visszakapott találatok.



A dokumentumok gombra kattintva egy másik panel jön be, ahol lehetőség van új dokumentum létrehozására, egy meglévő betöltésére, és annak adatainak módosítására, illetve törlésre.

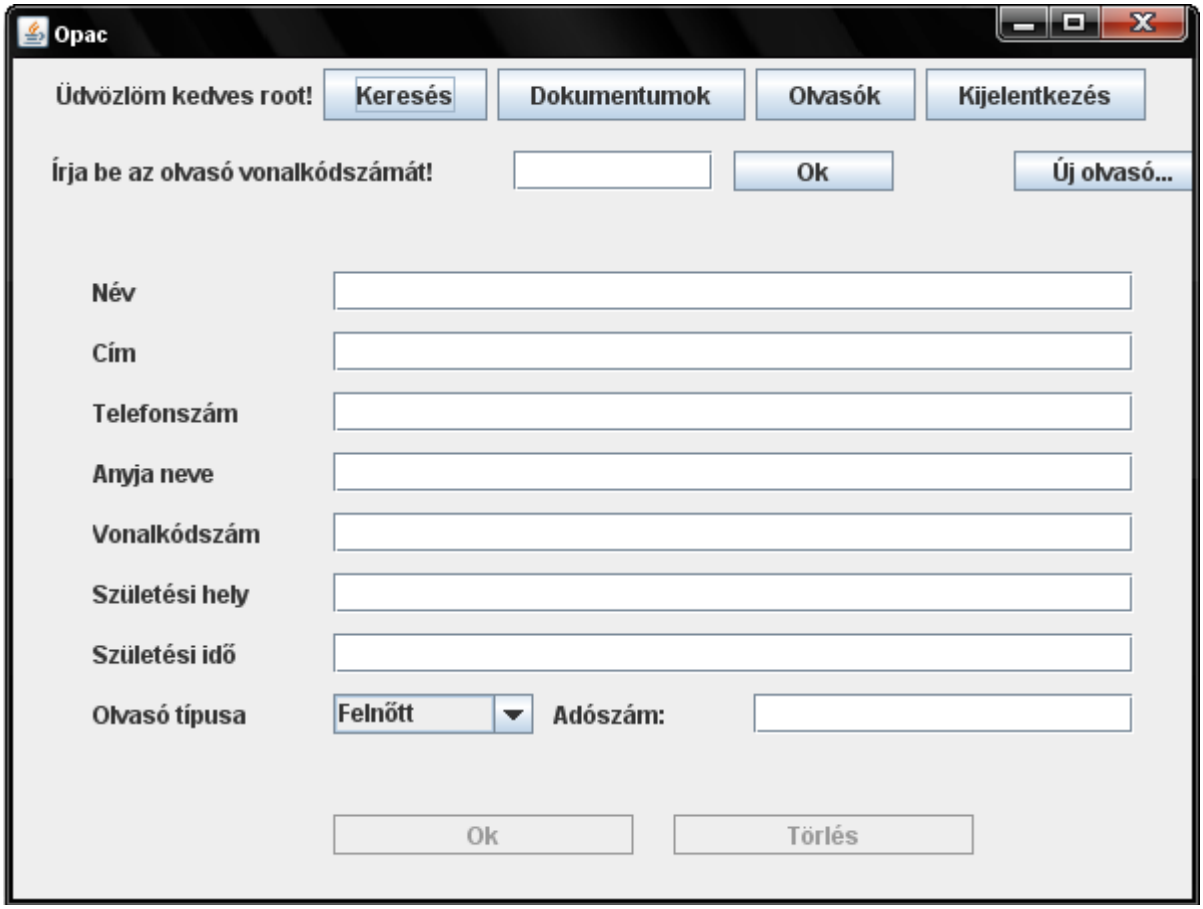


Az új dokumentumra kattintva lehetőség van új dokumentum felvitelére. Ilyenkor az Ok gomb elérhetővé válik, felirata pedig átvált Bevitelre. A különböző adatok kitöltése után a gombra kattintva az adatok betöltődnek az adatbázisba, a program részéről pedig egy nyugtázó dialógusablakot kapunk válaszul, miszerint adataink sikeresen rögzítésre kerültek. Ezek után a mezők és a választólista letiltódik, a bevitel gomb átvált módosítani akarom feliratra, a törlés gomb pedig aktívvá válik, így már tudjuk törölni a jelenlegi dokumentumot. Ugyanezen művelet fut végig akkor is, ha az Írja be a dokumentum vonalkódszámát felirat után levő szövegmezőbe beírom a dokumentum vonalkódszámát.



A törlés gombra kattintva egy megerősítést kérő dialógusablak pattan fel, melyből ha az igent választjuk, akkor a dokumentum minden a program jelenlegi állapotából, mind az adatbázisból törlődik. Amennyiben a nemre kattintunk, akkor a program visszavezet minket oda, ahol jártunk. Van még egy érdekesség, ha a dokumentum típusánál a választólistából a folyóiratot választjuk, akkor az ISBN szám címke értéke ISSN számra változik, és a szerző mező pedig letiltásra kerül.

Az olvasók gombra rákattintva ugyanazokat a műveleteket hajthatjuk végre, mint a dokumentumok esetében, újakat vihetünk fel, meglévőket törölhetünk, betöltés után az adatokat módosíthatjuk.



Itt is érdekességnek számít, hogy az Olvasó típusánál levő választólista más elemeinek kiválasztása esetén a mellette levő címke szövege más értékre vált. Felnőtt olvasó esetén az Adószám feliratot fogja tartalmazni, Nyugdíjas vagy Diák kiválasztásánál pedig az igazolvány számát kéri.

A kilépés gombra kattintva a bejelentkező képernyőre jutunk, így lehetőség nyílik egy másik használónak belépni a programba.

Összefoglalás

Szakdolgozatomnak egy Java oktatóprogram elkészítését vállaltam, melybe főként azért fogtam bele, mert a Javat nagyon megszerettem a tulajdonságai miatt. Éppen ezért, az órákon kívül is sokat foglalkoztam vele, könyveket vettem ki, hogy jobban megismerhessem ezt a nyelvet. A könyvek olvasása során sokszor az volt a tapasztalatom, hogy azok túlságosan elméletiesen és túlbonyolítva tálalják a Javat az olvasó elé. Pedig a Java tanulása lehet nagyon egyszerű is. Éppen ezért az oktatóprogram elkészítése során törekedtem arra, hogy a legegyszerűbb nyelvezetet használjam, és az elméleties tárgyalás helyett inkább a gyakorlatra helyezzem a hangsúlyt.

A program során elolvashatjuk az adott részhez tartozó ismereteket, majd tesztet tölthetünk ki, hogy teszteljük a lecke során megtanult ismereteket. Hogy még jobban ösztönözzem a felhasználót a Java gyakorlati részére, feladatokat adtam fel az egyes leckék végén, melyek igaz, nem kerülnek ellenőrzésre, hiszen egy olyan ellenőrzőprogram elkészítése, mely kijavítja, vagy valamilyen szinten pontozni tudja a felhasználó által készített programot, úgy gondolom, sokkal nagyobb feladat, mint az általam véghezvitt.

A program adatbázisszerverrel működik együtt, melynek azért láttam értelmét, mert egyrészt sokkal szebb megoldás, mint a fájlból való beolvasás. Másrészt pedig így a felhasználó nem tud „kutakodni” a teszteknél a megoldás miatt, bár igaz, hogyha ismeri az alapbeállításait az adatbázisszervernek és a kezelésében is otthon van, akkor megtudhatja azokat.

Törekedtem arra, hogy a felhasználót „fogjam”, ne engedjem addig olyan téma közelébe, amihez még nem érett meg a tudása. Ezt sikerült megvalósítani egy felhasználói rendszer megalkotásával mely, úgy gondolom jól sikerült.

A programot bárkinek szívesen ajánlom aki csak kicsit is érdeklődik a programozás iránt, hiszen a célkitűzésemnek megfelelően törekedtem arra, hogy ne legyen bonyolult a megfogalmazás.

Amit még szerettem volna, az a már említett programellenőrző, de ennek hiányának okát már kifejtettem. Ami viszont megvalósítható lett volna, az egy komplett telepítő, mely feltelepíti mindazon dolgokat, melyek szükségesek a program futtatásához. Ezt sajnos idő hiányában nem tudtam megvalósítani.

Összességében véleményem szerint sikerült a célkitűzéseket megvalósítani, megérte időt és fáradtságot nem kímélve dolgozni ezen a projekten. Bár volt, amit nem sikerült megvalósítani, vagy tudás vagy idő hiányában, de úgy gondolom, sikerült megalkotnom azt a programot, amire nagyon büszke vagyok.

Irodalomjegyzék

1. Instant Java, Java EE, SOA / Vég Csaba. - Debrecen : Logos 2000, 2007.
2. Java http://java.sun.com/
3. NetBeans http://www.netbeans.org/
4. Objektumorientált tervezés és programozás : Java / Angster Erzsébet. - 2. jav. kiad. [Budapest] : 4KÖR BT, 2003.
5. Java programozás 1.3. / Nagy Gusztáv
6. Langpop – http://langpop.com/
7. DevTopics – http://www.devtopics.com/
8. J2EE útikalauz Java programozóknak / Nyékyné Gaizler Judit. - Budapest, ELTE TTK Hallgatói Alapítvány, 2002