Debreceni Egyetem
Informatikai Kar

Tantervi háló kezelése ORACLE-ben

Témavezető:

Kósa Márk Szabolcs

Számítástechnikai munkatárs

Készítette:

Kun Anita

Programozó matematikus

Debrecen
2007

Tartalomjegyzék

1. Bevezetés

A számítástechnika fejlődésének egyik fontos jellemzője, hogy egyre több felhasználó egyre több számítógépen tárolt adatot használ fel. Az egyre növekvő információmennyiség egyre szélesebb körben válik elérhetővé. Az elkészített és alkalmazott számítógépi programrendszereknek növekvő adatmennyiséggel kell megbirkózniuk. A hétköznapjainkban is egyre gyakrabban találkozhatunk a számítógépes információs rendszerek alkalmazásával. Számítógépes információs rendszer fut az üzemekben, gyárakban a termelés irányítására, a pénzügyi, személyzeti, raktári feladatok elvégzésére. Ma már több tucat olyan információs rendszer létezik a világon, amely több terrabyte nagyságrendű adatot tartalmaz. A nagytömegű adatok feltételeknek eleget tevő kezelését adatbáziskezelő rendszerek végzik.

Tantervi háló kezelése Oracle-ban című szakdolgozatom célja az volt, hogy a már meglévő Java illetve Oracle ismereteimet felhasználva egy komplexebb feladatot oldjak meg. A téma kidolgozása során lehetőségem nyílt a Java nyelv gyakorlati alkalmazásába nagyobb betekintést nyerni, valamint egy távoli adatbázissal való kommunikáció kezelésével megismerkedni. A gyakorlatban is használható nyilvántartó alkalmazás készítése jó alkalom volt ezen ismeretek megszerzésére.

1.1. Feladat leírása

A feladat egy olyan nyilvántartó alkalmazás készítése, mely kezeli egy adott Intézmény[1] által felvett hallgatók szakjuknak/szakjaiknak megfelelő oklevelek megszerzéséhez előírt követelményeket. Az oklevélkövetelményeket a képzés meghatározott ideje alatt a Tanulmányi és vizsgaszabályzatnak megfelelően kell teljesíteni. Az egyes szakok tantervi

[1] a mi esetünkben: DE IK

hálója megmutatja, hogy melyek azok a tantárgyak, amik meghirdetésre kerülnek a hallgató tanulmányai során. A megfelelő ütemben meghirdetett tárgyakat a hallgatók a tantárgyi követelményeknek eleget téve fel tudják venni, hogy teljesítésükkel a képzési követelményeknek eleget tehessenek. Az alkalmazás lehetővé teszi, hogy egy adott hallgató teljesítéseit karbantarthassuk, vagy akár egy adott tárgyra vonatkozóan megtekinthessük azon hallgatók listáját, akik a tárgy előfeltételeit teljesítve jogosultak annak felvételére.

A biztonságos adatkezelés érdekében jelszavas beléptetés szükséges, valamint külön jogosultság, mellyel az adatok megtekinthetők ill. kezelhetők.

1.2. Programozási nyelv kiválasztása

A program kódja Java nyelven íródott[2]. A Java egy tiszta objektumorientált nyelv, nagy megbízhatóságú alkalmazások fejlesztésére hozták létre. A kitűzött célt a szigorú és erősen típusos fordítóprogrammal, valamint a futás közbeni dinamikus ellenőrzéssel érték el. Platformfüggetlen és hordozható, azaz a Java fordítóprogramja ún. bájtkódot készít, amit aztán a különböző platformokra telepített alkalmazás, az ún. Java Virtuális Gép[3] értelmezi, futtatja.

Az implementációhoz a JCreator fejlesztőeszközt használtam.

1.3. Adatbáziskezelő kiválasztása

Az Oracle adatbáziskezelő rendszert választottam. Az Oracle objektumrelációs adatbázis-kezelő rendszer. A relációs adatbázisok szokásos eszközrendszerét kibővíti egy szokásos objektumorientált eszközrendszerrel.

[2] J2SE 6.0
[3] JVM – Java Virtual Machine

2. Az alkalmazás megvalósításának lépései

Az első lépés a feladat elemzése, a megvalósítás részlépésekre bontása.

- Igényfelmérés és információgyűjtés, annak meghatározása, hogy mely felhasználók, milyen feladatokat akarnak elvégezni.
- Felhasználói felület tervezése. A feladat szempontjából célszerű, a felhasználói igényeknek megfelelő felhasználói felület készítése.
- Az adatbázis megtervezése, elkészítése.
- Megvalósítás a kiválasztott nyelven. (Java)
- Üzembe helyezés, tesztelés.

2.1. Felhasználói követelmények meghatározása

Az alkalmazás követelményeinek meghatározása természetesen a felhasználókkal történő konzultációsorozaton keresztül történik, együtt derítendők ki, elemzendők és dokumentálandók. Azt specifikálják, hogy az alkalmazásnak milyen szolgáltatásai legyenek majd, és ezek a szolgáltatások hogyan, milyen elvárásokkal működjenek. A felhasználói követelmények tehát a funkcionalitást jelentik.

2.2. Felhasználói felület megtervezése

Az utóbbi időben felerősödött az interfész-orientált tervezés. Ennek oka, hogy a tervezés nagyon korai szakaszában ha előállítunk prototípus[4] szintű felhasználói felületet, a felhasználó azonnal szembesül azzal, hogy ő milyen módon tud kapcsolatba kerülni az alkalmazással. Manapság a technológiák nagy része arról szól, hogy az interfészeket, felhasználói felületeket tervezik meg először, azonnal legyártják a prototípust, és figyelik a felhasználó reakcióit.

A felhasználó a felhasználói felületet látja, ha az alkalmazással találkozik. Szemben az összes többi tervezési kérdéssel a felhasználói felületek tervezésénél az informatikai szempontok mellett/helyett elsősorban a felhasználó fizikai és pszichikai, mentális képességeit, képtelenségeit kell figyelembe venni.

A program és felhasználó között interaktív kapcsolat, „párbeszéd” alakul ki: a felhasználó adatokat közöl a programmal, kezdeményezi ezeknek a feldolgozását, majd a program megjeleníti az eredményeket. Ez ma már többségében grafikus felhasználói felületen[5] keresztül történik. Ezek létrehozására a Java API osztályokat és interfészeket bocsát rendelkezésünkre, melyek meggyorsítják a fejlesztési munkát. Alapvetően kétfajta osztálykönyvtár közül választhatunk:

A korábban kifejlesztett osztálykönyvtár az AWT[6]. Az AWT komponenseit úgy tervezték, hogy az operációs rendszer elemeihez hasonlóak legyenek. Használatuk kissé nehézkes, mivel az AWT osztályok közvetítő feladatot látnak el a felhasználó programja és az operációs rendszer között.

Az újabb osztálykönyvtár a Swing, amely sokkal gazdagabb, mint az AWT, és a fent említett korlátozás sem létezik. Természetesen a Swing osztály vezérlői sokkal többet tudnak, felépítésük logikusabb. Mivel formatervezésük sajátos, minden platformon azonosan néznek ki a grafikus elemek. A felület elkészítéséhez így Swing-et használtam.

[4] Prototípus: az alkalmazás egy nagyon korai implementációja; implementációk verzióin keresztül jutunk el a véglegesnek mondható verzióhoz
[5] GUI – Graphical User Interface
[6] Abstract Window Toolkit

2.3. Az adatbázis megtervezése

A sémát SQLDeveloper segítségével készítettem. Az Oracle adatkezelő nyelveit, az SQL és PL/SQL nyelvet használtam. A fejlesztői környezet áttekinthető, segítségével hatékonyan és gyorsan sikerült a programban alkalmazott tárolt eljárásokat megvalósítani. A program működéséhez létre kell hozni az adatbázist, öt táblára lesz szükség. A feladat szempontjából külön kell nyilvántartani a hallgatók, szakok, tantárgyak, leckekönyv és a szakokon kötelezően teljesítendő tantárgyakra vonatkozó információkat:

A hallgatók adatai:
- -Név
- -Neptun kód
- -Születési év
- -Anyja neve

A szakok adatai:
- -Szak név
- -Szak kód

A tantárgyak adatai:
- -Tantárgy kód
- -Tantárgy név
- -Előfeltételei

A leckekönyv adatai:
- -Neptun kód
- -Szak kód
- -Beiratkozott félévek
- -Teljesített tárgyak

A kötelező tárgyak listája az adott szakon:
- -Szak kód
- -Kötelező tárgyak listája

A tantárgyak előfeltételeinek tárolásánál tárolhattuk volna az előfeltételeket egy külön táblában is, ekkor a Tantárgy és Előfeltételek táblák között M:N számosságú kapcsolat állt volna fenn. Az előfeltételek tárolásának legegyszerűbb módja azonban az, hogy kollekciót alkalmazunk. A kollekció a harmadik generációs programozási nyelvek tömbfogalmával

analóg eszköz a PL/SQL-ben. Homogén, összetett típus. Az Oracle három kollekciótípust ismer: index alapú tábla, beágyazott tábla, dinamikus tömb. Az utóbbit alkalmaztam az előfeltételek, teljesített tárgyak és a kötelező tárgyak tárolására. A létrehozásához szükség van egy felhasználói típus definíciójára.

```
CREATE  TYPE  T_Beiratkozasok IS VARRAY(20) OF VARCHAR2(30);
/
```

A `VARRAY` jelzi, hogy dinamikus tömbtípust hoztunk létre. 20 a dinamikus tömb indexeinek felső határa, a `VARCHAR2` pedig egy PL/SQL adattípus.

Az adatbázis implementációja:

```
CREATE TABLE Hallgato(
  Nev          VARCHAR2(30),
  N_kod        VARCHAR2(6),
  Szul_ido     DATE,
  Anyja_neve   VARCHAR2(30),
  CONSTRAINT Hallgato_pk PRIMARY KEY (Nev, N_kod)

 )
/

CREATE TABLE Szak(
  Szak_kod     VARCHAR2(10),
  Szak_nev     VARCHAR2(30),
  CONSTRAINT Szak_pk PRIMARY KEY (Szak_kod)
)
/

CREATE TYPE T_tantargy_lista IS VARRAY(10) OF VARRAY(10);
/


CREATE TABLE Tantargy(
  Tant_kod       VARCHAR2(10),
  T_nev          VARCHAR2(30) NOT NULL,
  Elofeltetelek T_tantargy_lista,
  Kredit         NUMBER(2),
  CONSTRAINT tantargy_pk PRIMARY KEY (Tant_kod)
)
/

CREATE TYPE T_Beiratkozasok IS VARRAY(20) OF VARCHAR2(9);
/
CREATE TYPE T_Teljesitesek IS VARRAY(150) OF VARCHAR2(10);
/
```

```
CREATE TABLE Leckekonyv(
  Hallg_nev      VARCHAR2(30),
  Szak_kod       VARCHAR2(30),
  Beiratkozások  T_Beiratkozasok,
  Teljesitesek   T_teljesitesek,
  CONSTRAINT Leckekonyv_pk PRIMARY KEY (Hallg_nev, Szak_kod),
  CONSTRAINT Leckekonyv_fk2 FOREIGN KEY (Szak_kod)
   REFERENCES Szak(Szak_kod)
)
/

CREATE TABLE KotTantargy(
   Szak_kod VARCHAR2(30),
   Targyak  T_kot_tantargy_lista
)
/
```

2.3.1. Kommunikáció az adatbázissal

A Java programok kommunikálni tudnak különféle adatbázisokkal. Az adatbázis-hozzáférést a JDBC – Java DataBase Connectivity – biztosítja. A JDBC egy programozói interfész (API) SQL – Structured Querry Language - utasítások végrehajtására.

A JDBC API szolgáltatásait három csoportba lehet sorolni:

- összekapcsolódás relációs adatbázissal
- SQL utasítások végrehajtása
- SQL lekérdezések eredményeinek feldolgozása

A JDBC használatával a Java adatbáziskezelő programjaink platform és adatbázis-kezelőktől függetlenekké válnak. JDBC API típusait a java.sql és a javax.sql csomagok tartalmazzák, melyek már a Java 2 részei, ezért a JDK 1.2-es változata már tartalmazza őket. Szükség van egy driverre[7], amely egy távoli gépen található Oracle adatbázissal kommunikál. A meghajtóprogram kiválasztása a `DriverManager` osztály segítségével történik. A

[7] meghajtóprogramra

meghajtóprogram osztályának direkt betöltése a `Class.forName` statikus metódussal történik, amely a paraméterként kapott nevű osztály dinamikus betöltését végzi el.

```
Class.forName( driver );
```

Ahol:

```
String driver = "oracle.jdbc.OracleDriver";
```

Az adatbázisspecifikus driver létrehozza a kapcsolatot a kívánt adatbázissal visszaadva egy `Connection` típusú objektumot. A program és az adatbázis közötti kapcsolatot a `Connection` objektum képviseli. A `Connection` tulajdonképpen egy interfész, amely biztosítja a kérések eljutását az adatbázishoz és a válaszok megérkezését az adatbázistól. A `DriverManager` osztály tartalmaz egy listát a regisztrált driverekkel. A `getConnection` metódus hívásakor megpróbálja megtalálni a megfelelő drivert, mely kapcsolódni tud az URL-ben megadott adatbázishoz.

```
szuloPanel.con = DriverManager.getConnection( url, m_loginName,
  m_password);
```

ahol:

```
String url = "jdbc:oracle:thin:@oracle.inf.unideb.hu:1521:oracle";
```

A `Connection` típusú objektum segítségével létrehozhatunk egy `Statement` vagy egy `PreparedStatement` objektumot, amely egy SQL kérést is tartalmaz az adatbázis felé. Ez tartalmaz egy `SQL SELECT` utasítást, amely az eredmény oszlopait állítja elő.

A kettő közti különbség a következő:

`Statement`: egyszerű SQL utasítások végrehajtására használható. A `Connection` objektum `createStatement` metódusával hozható létre egy `Statement` objektum. A `Statement` interfész egyik fontos metódusa az `executeQuery`. A paraméterben

megadott SQL utasítást végrehajtja és egy annak eredményét reprezentáló eredménytábla[8] objektumot ad vissza.

```
Statement statement = szuloPanel.con.createStatement();

ResultSet res = statement.executeQuery("SELECT Szak_kod FROM Szak" );
```

A `PreparedStatement` objektumot bemenő paraméterekkel rendelkező SQL utasítások végrehajtására használjuk. Az `SQL` parancsban kérdõjel jelzi a megadandó paraméter helyét, ezt az alábbiakban a `setString` módszerrel definiáljuk. Tehát az utasítás végrehajtása előtt minden bemenő paraméternek értéket kell adni a megfelelő `set` metódusok valamelyikével. Minden kérdőjel egy paraméter helyén áll, amely a lekérdezés végrehajtása előtt beállítható. Az adatbázis ezeket paraméterekkel helyettesíti be, mielőtt végrehajtja a lekérdezést.

```
PreparedStatement statement = szuloPanel.con.prepareStatement(
 "SELECT Teljesitesek
    FROM Leckekonyv
      WHERE Hallg_nev = ? ");

statement.setString(1,str2);

ResultSet res2 = statement2.executeQuery();
```

Az `executeQuery` metódus itt is egy `ResultSet` objektumot ad vissza, ami egy olyan Java objektum, ami egy SQL lekérdezés eredményét tartalmazza. A különböző oszlopokhoz egy soron belül `set` és `get` metódusokkal férhetünk hozzá.

[8] `ResultSet`

```
try {
  Statement statement = szuloPanel.con.createStatement();
  ResultSet res = statement.executeQuery(
    "SELECT N_kod
       FROM Hallgato
         ORDER BY N_kod" );
  String s;
  while ( res.next() ){

     s = res.getString(1);
     m_neptunBox.addItem(s);
  }
}
catch(SQLException e){
  System.err.println(e.getMessage());
}
```

A `ResultSet` objektum tartalmaz egy kurzort, ami az aktuális sorra mutat. A `ResultSet` objektum létrehozásakor az első sor elé van beállítva, és a `next` metódus első hívása beállítja az első elemre. Egy `while` ciklus segítségével így könnyen feldolgozható egy lekérdezés eredménye.

2.4. Grafikus felület készítése

2.4.1. Az ablakok általános felépítése

Az alkalmazás felhasználói felületének kialakításához a már korábban említett Swing eszközkészletet használtam. A grafikus felületek készítéséhez felhasználható építőelemek mindegyike a JDK Component osztályból származik. Az elemeket reprezentáló osztályok, metódusok sorozatát tartalmazzák, amelyekkel a grafikus elemek értéke és kinézete manipulálható.

A felületek komponenseinek megfelelő elhelyezését a tárolókban, felületmenedzser vagy elrendezéskezelők[9] segítségével oldjuk meg. A layout manager feladata, hogy optimálisan rendezze el a grafikus elemeket a rendelkezésre álló felületen. A felületmenedzser megkönnyíti a programozók feladatát, mivel nem kell különösebben törődniük azzal, hogy hogyan is fog kinézni az alkalmazás, ha a felhasználó például átméretezi az ablakot. A felület egyes részeinek külön felületmenedzserük is lehet. Ez azt jelenti, hogy az alkalmazás felületét több részre bonthatjuk (konténerek v. tárolók), s ezeken a részeken belül más és más stratégia alapján rendeztethetjük el az elemeket. Minden konténer rendelkezik egy alapértelmezett felületmenedzserrel. Néhány a programban használtak közül:

FlowLayout: Az egyik legegyszerűbb felületmenedzser. Az add(`komponens`) meghívásával a tárolóhoz adott vezérlőket sorban, egymás mellé helyezi el. Ha az ablak szélessége miatt nem fér el több egy sorban, akkor a további komponensek a következő sorba kerülnek.

BorderLayout: Ennek a segítségével a komponenseket ötféleképpen helyezhetjük el.[10] Az északi, déli, keleti, nyugati területre elhelyezett objektumokat a menedzser vízszintes irányban

[9] LayoutManager
[10] észak, dél, kelet, nyugat, közép

széthúzza, hogy azok kitöltsék a rendelkezésre álló helyet, azonban a komponensek magasságát nem változtatja meg.

BoxLayout: Függőleges vagy vízszintes irányban egymás mellett helyezkednek el a komponensek.

Mivel a komponensek tárolókban helyezkednek el, így pozíciójukat és a méreteiket annak bal felső sarkához rögzített koordináta-rendszerben értelmezzük[11]. Ha nem akarunk elrendezéskezelőt alkalmazni[12], akkor beállíthatjuk saját magunk is a pozíciót és a méretet különféle metódusok segítségével. Az alkalmazásban ilyen metódusok: a `setSize`, `setVisible`, `setEnable` stb.

2.4.2. Eseménykezelés

A programok futását a perifériákról[13], illetve a grafikus felület elemeitől érkező események vezérlik, mivel a felhasználó cselekedetei eseményeket generálnak. A program értesül ezekről az eseményekről, ami alapján megtudja, mi a felhasználó akarata és eldönti, hogy akar-e erre reagálni. Alapvetően kétféle eseményt különböztetünk meg:

- Az az esemény, amit a beviteli eszköz vált ki
- Magasabb szintű rendszeresemény (pl.: ablak bezárása, menüpont kiválasztása)

Minden esemény ősosztálya az `EventObject` és minden egyes eseményt külön osztály ír le. A komponensek a `processXXXEvent` metódusok segítségével dolgozzák fel az eseményeket. Az `XXXEvent` eseményosztályokhoz tartozik egy ún. figyelő interfész `(XXXListener)` , amit implementálnunk kell, ha figyelni szeretnénk egy bizonyos eseményt, és hozzá kell adni ahhoz a komponenshez, ahonnan az esemény származik.

[11] az x-tengely jobbra, az y-tengely pedig lefelé mutat
[12] `setLayout(null)`
[13] egér, billentyűzet

2.4.3. Felületek funkciói

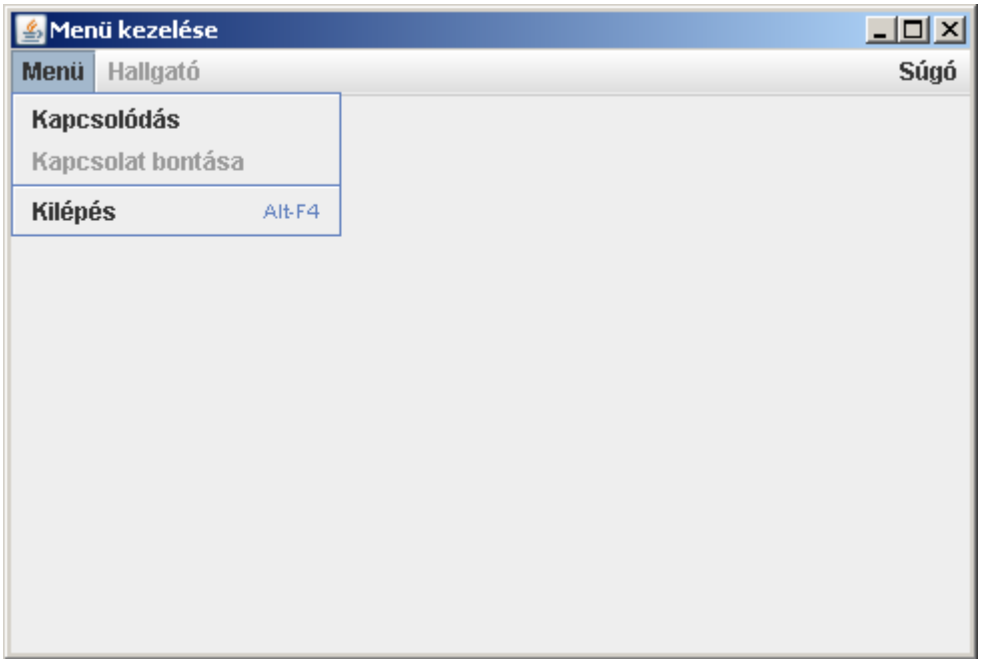
Az alábbi funkciókra osztottam a feladatot:

- A hallgatói adatbázison végrehajtható feladatok megtekintése
- Kapcsolódás a hallgatói adatbázishoz ill. kapcsolat bontása
- Új hallgató felvétele a nyilvántartásba
- Kiválasztott hallgató teljesítéseinek megtekintése, módosítása
- Adott tárgyhoz tartozó hallgatók listája

2.4.3.1. A hallgatói adatbázison végrehajtható feladatok megtekintése

`Menu1.java`

Az alkalmazás elindítása után egy ablak lesz látható:



A fenti ablak a `JFrame` egy példánya `(Menu1.java)`. Láthatóan központi helyet foglal el az alkalmazásban, a program innen indul, ez a file tartalmazza a `main` függvényt. Az egyes tevékenységek kiválasztása menüsor segítségével történik. Attól függően, hogy milyen menüelemet választunk, megjelenik a hozzátartozó panel. Az ablak menüsorát a `JMenuBar` osztály objektuma valósítja meg, melyhez lenyíló menüket csatlakoztatunk a `JMenu` osztály segítségével. Mindkét osztály listajellegű műveleteket tartalmaz, hiszen a menüsor objektum menük listája, míg a menük a menüpontok `(JMenuItem)` listája. Ezekhez gyorsbillentyűket is rendelhetünk a `MenuShortCut` segítségével.

Az konstruktorban beállítottam az ablak méretét a `setSize` metódussal, a `DefaultCloseOperation` metódussal pedig megmondtam, hogy mi történjen abban az esetben, ha kísérletet teszünk az ablak bezárására. Paraméterként a következőket adhatjuk át:

- EXIT_ON_CLOSE – az ablak bezárása után kilép a programból
- DISPOSE_ON_CLOSE – az alkalmazás tovább fut, de bezárja és eldobja az ablak objektumát
- DO_NOTHING_ON_CLOSE – nem tesz semmit, csak programból lehet bezárni az ablakot
- HIDE_ON_CLOSE – bezárja az ablakot, és tovább fut a program

Ahhoz, hogy a menüpontok kiválasztásával megjelenjenek a megfelelő ablakok, a menüpontokhoz eseményfigyelőket kell rendelni. Az osztály az ActionListener interfészt implementálja. Ennek actionPerformed metódusában a setTitle metódus segítségével kérdezhetem le az eseményt kiváltó menüpont nevét.

2.4.3.2. Kapcsolódás a hallgatói adatbázishoz

Jelszo.java

A főablak menüjében a Kapcsolódás menüpontot választva megjelenik a Bejelentkezés ablak. A JDialog osztály egy leszármazottja, örökli annak adattagjait, metódusait. Az ablak tartalmaz egy JTextField és egy JPasswordField típusú adatbeviteli mezőt: m_loginNameBox és m_passwordBox. Itt adhatjuk meg a felhasználó nevünket és jelszavunkat[14]. Az OK[15] gombra kattintva ezeket a program megvizsgálja, és ha megtalálta az adatokat az adatbázisban, akkor a felhasználó sikeresen belépett. Ezt követően egy dialógusablak jelenik meg, ami jelzi a kapcsolódás sikerességét. A Kapcsolódás menüpont nem-engedélyezetté válik. Amennyiben hibás volt a felhasználói név vagy a jelszó, szintén egy dialógusablakban kapunk erről információt, ami figyelmeztet a rossz adatbevitelre. A kapcsolódási kísérlet során SQLException váltódhat ki, amit kivételkezelőben le kell kezelni.

```
   JOptionPane.showMessageDialog(this,"Hiba!",
     "Hib\u00e1s n\u00e9v vagy jelsz\u00f3!!!",
JOptionPane.ERROR_MESSAGE);
```

Található még a felületen egy Mégsem gomb, ami az OK gombhoz hasonlóan egy JButton példány. Hatására eltűnik a Bejelentkezés ablak, köszönhetően a Menu1 osztályban definiált removePanel metódusnak, ami paraméterül megkapja ezt az ablakot. Az ablak pozícióját és méretét a setBounds metódus segítségével állítottam be. A komponensek elrendezéséhez FlowLayout elrendezéskezelőt használtam.

Az adatbázissal való kapcsolat bontására a főablak menüjében a Kapcsolat bontása menüpont kiválasztásával van lehetőségünk.

[14] A korábbiakban már volt arról szó, hogyan is lehet egy felhasználói felületen keresztül kapcsolatba lépni különféle adatbázisokkal.
[15] JButton típusú

2.4.3.3. Új hallgató felvétele az adatbázisba

Ujhallg.java

A főablak menüjében az Új hallgató hozzáadása menüpontot választva megjelenik egy ablak, ami a JDialog osztály leszármazottjaként örökli annak adattagjait, metódusait. Használatával új hallgatókat vihetünk be az adatbázisba. Az ablak JTextField típusú[16] beviteli mezőket tartalmaz, ahol megadhatjuk a szükséges információkat[17]. Tartalmaz még egy lenyíló listát[18], ahol kiválaszthatjuk, hogy milyen szakos az új hallgató. A szakkódokat a listához egy Statement objektum segítségével rendeljük hozzá, lekérdezve az adatbázisból a karon létező szakok kódját.

Az OK gomb lenyomásával létrejön egy CallableStatement típusú objektum. A CallableStatement interfész a PreparedStatement kiterjesztettje. Tárolt SQL eljárások meghívására lehet használni. Egy tárolt eljárás hívása a bemenő paramétereken kívül kimenő paramétereket is használhat. Egy paraméter egyszerre lehet kimenő és bemenő is. Létrehozása a Connection objektum prepareCall metódusával történik. Az eljárás meghívása:

```
CallableStatement cs = szuloPanel.con.prepareCall(
  "{call hallg_beszur(enev, ekod, eszulido, eanyanev)}");
```

Végrehajtás előtt be kell állítani a bemenő paraméterek aktuális értékét. Itt is, mint a PreparedStatement estében a ResultSet objektum tartalmazza a művelet eredményét, a táblázat első sorától kezdődően. A tárolt eljárás meghívásával az ablakban megadott adatokkal egy új hallgatót vehetünk fel a nyilvántartásba.
A méretezés a setBounds metódussal, a komponensek elrendezése az ablakban, FlowLayout és BorderLayout használatával történt. A Mégsem gomb működésére igazak az előző fájlban elmondottak.

[16] egysoros szövegmezőket
[17] hallgató neve, neptun kódja, születési idő, anyja neve
[18] JComboBox típusú

2.4.3.4. Kiválasztott hallgató teljesítéseinek megtekintése, módosítása

Teljesites.java

A főablak menüjében a Teljesítések karbantartása menüpontot választva megjelenik egy ablak, ami a JDialog osztály leszármazottjaként örökli annak adattagjait, metódusait. Itt neptunkód alapján megtekinthetjük vagy módosíthatjuk egy adott hallgató eddigi tanulmányai során teljesített tantárgylistáját. A neptunkódok tárolására JComboBox típusú lenyíló listát használtam. Elemeit egy Satement objektum segítségével, az adatbázisból való lekérdezéssel adtam hozzá. Kiválasztva egy neptunkódot a lenyíló listából, a képernyőn egy JTextField típusú mezőben megjelenik a neptunkód alapján beazonosított Hallgató neve. Az ablak tartalmaz két JList típusú listát. Az első tartalmazza a hallgató szakjának megfelelően teljesítendő tantárgyak listáját, a második a kiválasztott hallgató eddig teljesített tantárgyait. A listák feltöltése az adatbázisból történik. Például a második lista estében egy PreparedStatement objektum segítségével egy tömbbe lekérdezem a hallgató kollekció típusú oszlopának[19] elemeit, majd a tömb elemein végiglépkedve hozzáadom a tömbelemeket a listához.

```
try{
  PreparedStatement statement1 = szuloPanel.con.prepareStatement(
    "SELECT Teljesitesek
       FROM Leckekonyv
         WHERE Hallg_nev = ? " );
  statement1.setString(1,m_loginNameBox.getText());
  ResultSet res1 = statement1.executeQuery();
  while ( res1.next() ){
    ARRAY array = ((OracleResultSet)res1).getARRAY ("Teljesitesek");
    tr = (String[])array.getArray ();
    for(int i= 0; i<tr.length; i++)
      listModel2.addElement(tr[i]);
  }
}
catch(SQLException exx){
  JOptionPane.showMessageDialog(this,exx.getMessage(),"HIBA",
    JOptionPane.ERROR_MESSAGE);
}
```

[19] Teljesitesek

A listákat elláttam függőleges görgetősávval a `JScrollPane` osztály segítségével.

```
listagorgeto = new JScrollPane(lista1,
  ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
    ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
```

A görgetősávok működéséről a konstruktor paraméterében található konstansok gondoskodnak. Mivel a `DefaultListmodel` osztály számos metódust tartalmaz listák kezelésére, ezért a listákhoz hozzárendeltem egy-egy ilyen típusú objektumot. A listákon az elemkiválasztás is beállítható:

```
lista2.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

A felületen a két lista között elhelyeztem két gombot[20]. A `Hozzáad` gomb használatával a szak kötelező tárgyaiból egyszerre többet is hozzáadhatunk a teljesített tárgyak listájához. A `Töröl` gomb hatására a teljesített tárgyak listájából eltűnnek a kijelölt elemek. Saját eseménykezelőt hoztam létre mindkét gomb számára. A `Hozzáad` gomb eseménykezelője:

```
class HireListener implements ActionListener {
  private boolean alreadyEnabled = false;
  private JButton button;
  public HireListener(JButton button) {
    this.button = button;
  }
  public void actionPerformed(ActionEvent ea) {
    if(!torolGomb.isEnabled()){
      torolGomb.setEnabled(true);
    }
    for (int index: lista1.getSelectedIndices()){
      String s = (String)listModel1.getElementAt(index);
      if(alreadyInList(s))
        Toolkit.getDefaultToolkit().beep();
      else{
        listModel2.addElement(s);
        lista2.setSelectedIndex(index);
        lista2.ensureIndexIsVisible(index);
      }
    }
  }
}
```

[20] `Hozzáad, Töröl`

A Töröl gomb eseménykezelője:

```
class FireListener implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    int index = lista2.getSelectedIndex();
    listModel2.remove(index);
    int size = listModel2.getSize();
    if (size == 0)
      torolGomb.setEnabled(false);
    else {
      if (index == listModel2.getSize())
      index--;
    }
    lista2.setSelectedIndex(index);
    lista2.ensureIndexIsVisible(index);
  }
}
```

A felületen a OK gombra kattintva, tárolt eljárás meghívásával véglegesítődnek az adatbázisban is a hallgató teljesítésein végrehajtott módosítások. A Mégsem gombra kattintva eltűnik az ablak.

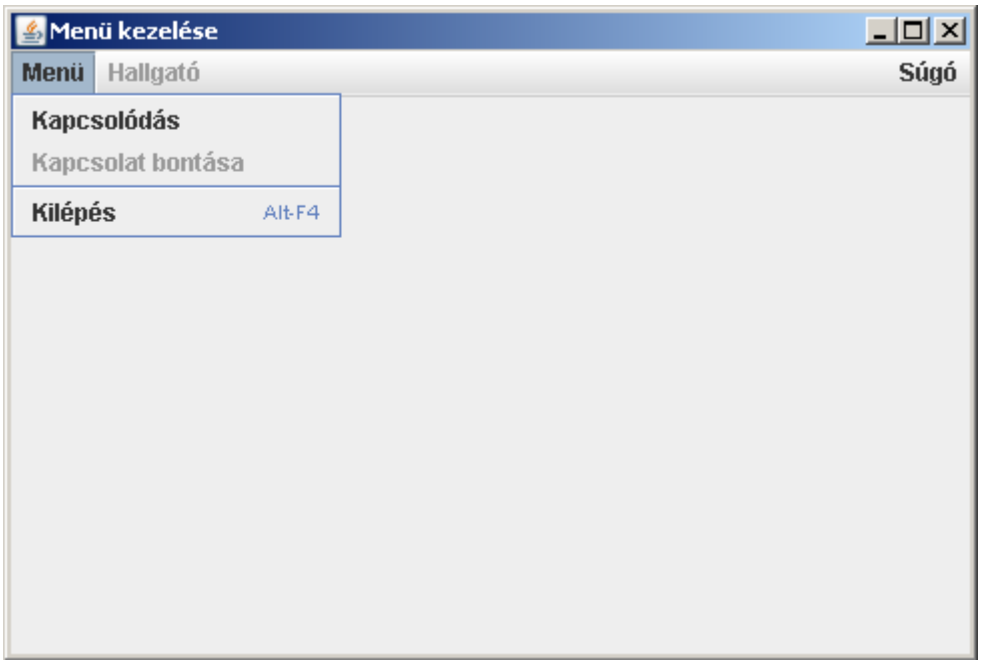
2.4.3.5 Adott tárgy felvételére jogosult hallgatók listája

Nevsor.java

A főablak menüjében a Hallgató lista menüpontot választva megjelenik a JDialog osztály leszármazottjaként egy ablak, ahol egy tantárgykód kiválasztásával lekérdezhetünk egy hallgatói listát. A lista azon hallgatókat tartalmazza, akik jogosultak a kiválasztott tárgy felvételére, azaz teljesítették annak minden előfeltételét és még nem teljesítették a tárgyat. A tantárgykódok tárolására JComboBox típusú lenyíló listát használtam. Elemeit egy Satement objektum segítségével, az adatbázisból történő lekérdezéssel adtam hozzá. Kiválasztva a tantárgykódot a lenyíló listából, a képernyőn egy JTextField típusú mezőben megjelenik a tantárgy neve és egy JList típusú listában a hallgatók neptunkódjukkal együtt, névsor szerint. A Mégsem gomb hatására eltűnik az ablak.

3. Felhasználói dokumentáció

A program futtatásához JDK szükséges, célszerű a legfrissebb verzió letöltése. (http://java.sun.com/j2se/downloads/) A program elindítása az indit.bat fájl segítségével történik. Indítás után a következő ablak lesz látható:

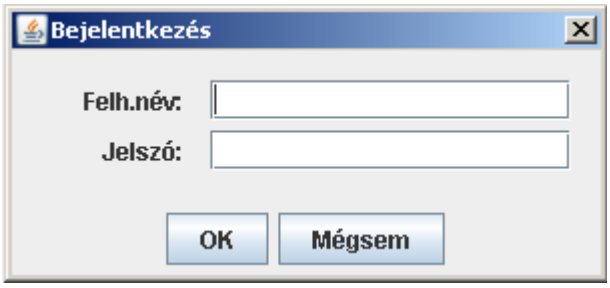


1. ábra: A Menü menü kiválasztása

A Hallgató menü akkor válik engedélyezetté, ha kapcsolódtunk az adatbázishoz.

3.1. Bejelentkezés

A felhasználónak kapcsolódnia kell az adatbázishoz, mielőtt bármit is szeretne csinálni. Ha a `Kapcsolódás` menüpontot választom a következő ablak lesz látható:



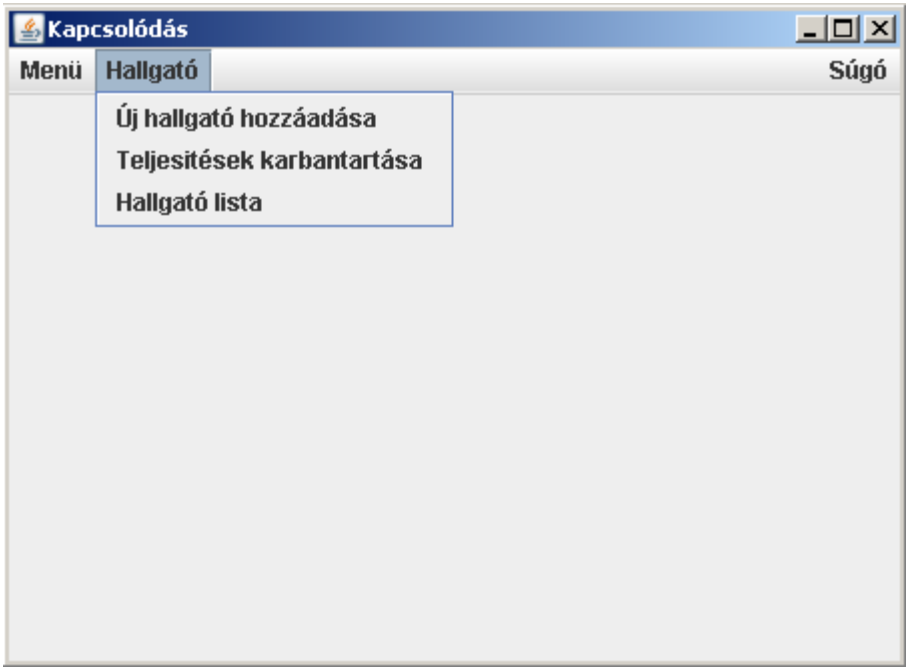
2. ábra: A Bejelentkezés panel

A szövegmezők kitöltése után megnyomva az OK gombot, csatlakozunk az adatbázishoz, eltűnik a `Bejelentkezés` ablak, és egy dialógusablak jelzi, hogy a kapcsolat létrejött:

3. ábra: Információ az adatbázishoz való sikeres kapcsolódásról

3.2. A felhasználó lehetőségei

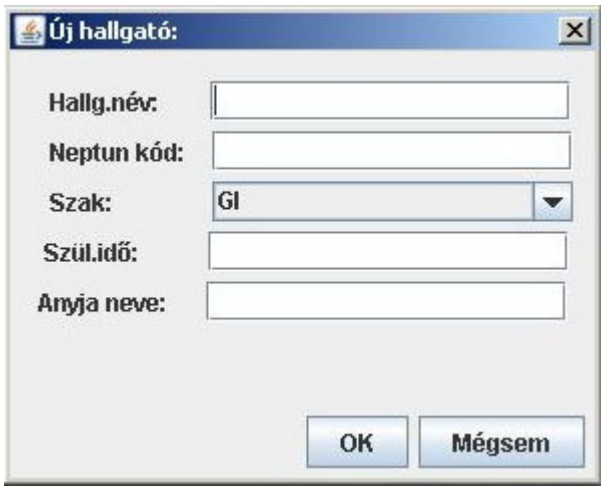
Miután sikeresen kapcsolódtunk az adatbázishoz, a `Hallgató` menü engedélyezetté válik. Ezek után a felhasználónak lehetősége nyílik új hallgatót felvenni a nyilvántartásba, egy kiválasztott hallgató teljesítéseit megtekinteni, esetleg módosítani, vagy lekérdezni azon hallgatók listáját, akik egy kiválasztott tantárgy felvételére jogosultak.



4. ábra: A felhasználó lehetőségei kapcsolódás után

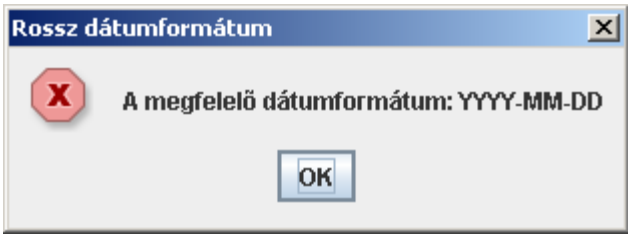
3.3.Új hallgató felvétele a nyilvántartásba

Az `Új hallgató` hozzáadása menüpontot kiválasztva megjelenik a következő ablak:



5. ábra: Új hallgató felvétele a nyilvántartásba

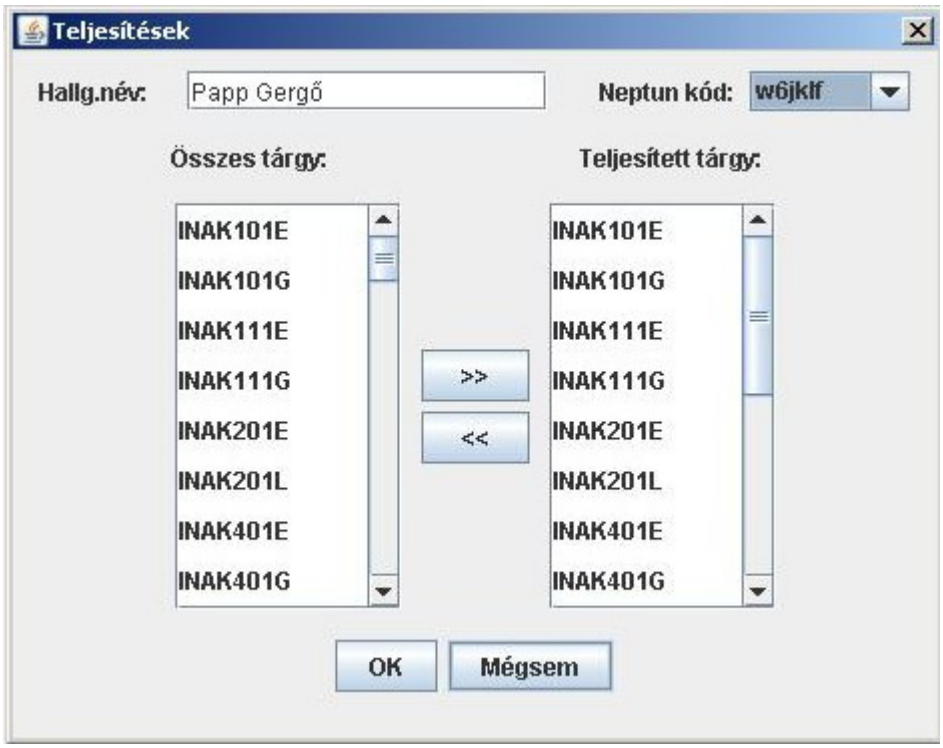
Itt lehetősége van a felhasználónak egy új hallgatót regisztrálni, megadva annak minden szükséges adatát. Ha esetleg rossz dátumformátumot adtunk meg, erről hibaüzenetet kapunk:



6. ábra: Hibaüzenet hibás dátum megadása esetén

3.4 Hallgató teljesített tárgyainak kezelése,manipulálása

A `Teljesítések karbantartása` menüpontot kiválasztva megjelenik a `Teljesítések` ablak. Itt a felhasználó megtekintheti egy hallgató teljesített tárgyainak listáját, új tárgyakat adhat ezekhez hozzá, vagy törölhet is közülük. Kiválasztani egy adott hallgatót a lenyíló listából neptunkódja alapján lehetséges. A kiválasztás hatására megjelenik a hallgató neve, a baloldali listában a hallgató szakján kötelezően teljesítendő tantárgyak kódja, a jobboldalon a hallgató eddig teljesített összes tárgya neptunkód alapján rendezetten. Ha új tárgyakat szeretnénk hozzáadni a teljesítésekhez[21], akkor a baloldalon egyszerre több kiválasztható listaelem együttes hozzáadását a >> gomb megnyomásával tehetjük meg. Ha a jobboldali listából tárgyat szeretnénk törölni, azt a << gombbal tehetjük meg. Az Ok gomb megnyomásával ezek a változások az adatbázisban is véglegesítésre kerülnek.

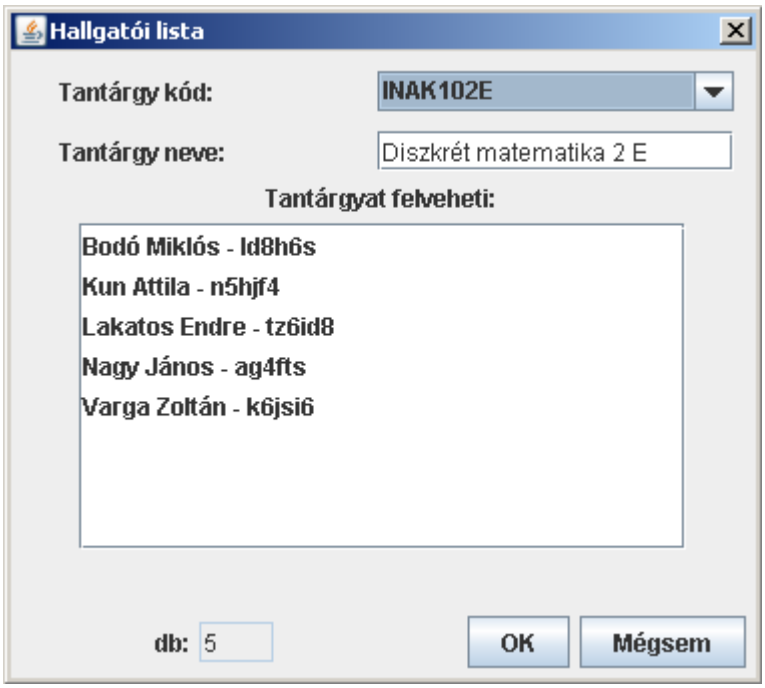


7. ábra: Hallgató teljesítéseinek kezelése

[21] jobb oldali lista

3.5. Hallgatói lista lekérdezése

A `Hallgató lista` menüpontot választva a felhasználónak lehetősége van egy adott tantárgyhoz kapcsolódóan lekérdezni egy hallgatói listát, ami azon emberek neveit tartalmazza, akik jogosultak a tárgy felvételére. A lenyíló listából kiválasztva a tantárgykódot, megjelenik a tárgy neve és a tanulók listája ábécé sorrendben neptunkódjukkal együtt. .



8. ábra: tárgyhoz tartozó hallgató lista

`A Súgó` menüt választva néhány információ jelenik meg a programról.

4. Összegzés

A szakdolgozat célja az volt, hogy egy gyakorlatban is alkalmazható információs rendszert tervezzek és valósítsak meg az Oracle adatbáziskezelő rendszer segítségével. Az elkészült alkalmazás abban könnyíti meg a felhasználó munkáját, hogy kapcsolatba kerülve egy hallgatói adatbázissal, leellenőrzi, hogy egy adott karon kik azok a hallgatók, akik jogosultak egy - az általa kiválasztott- tantárgy felvételére. A hallgatói listák lekérdezésén kívül lehetősége van még új hallgatókat felvenni a nyilvántartásba vagy a már meglévők eddig teljesített tantárgylistáján az esetleges módosításokat elvégezni.

A rendszert további funkciókkal kiegészítve, az életben is használható rendszerré fejleszthető tovább. Jövőbeni fejlesztési feladat, hogy a táblák, lekérdezések adatainak textfájlba mentését, nyomtatását meg kell oldani. Az alkalmazás külleme, a felületei tovább tökéletesíthetők a Swing osztály nyújtotta „korlátlan” eszközkészlet segítségével. A hallgatói listák ill. tantárgylisták megjelenítése megoldható táblázatok használatával is, választásom azonban mégis a lista használatára esett. Igaz, a lehetőségei kissé korlátozottak, azonban tökéletesen megfelelt az alkalmazásban használt adatok tárolására, kezelésére.
A hallgatói adatbázis is tovább bővíthető újabb táblák létrehozásával, a tárolni kívánt adatok igényeinek megfelelően. Az adatbázishoz való kapcsolódás során a jogosultságok ellenőrzésére is ki kell dolgozni egy megfelelő módszert.

5. Irodalomjegyzék

[1] Juhász István: A rendszerfejlesztés technológiája c. tantárgy kézirata

[2] Haladó Oracle I. és II. tantárgy órai kézirata Juhász István előadásai alapján

[3] Haladó Oracle I. és II. tantárgy jegyzet L. Nagy Éva gyakorlatai alapján

[4] Juhász István, Gábor András: PL/SQL programozás Alkalmazásfejlesztés ORACLE 9i-ben, Panem 2002.

[5] Kende Mária, Nagy István: ORACLE példatár, Panem 2005.

[6] Matthew Robinson, Pavel Vorobiev: Swing, Special Sales Department Manning Publications Co.

[7] http://www.sun.com

[8] http://www.oracle.com

[9] http://www.oracle.com/pls/db102/homepage

[10] http://java.sun.com/docs/books/tutorial/uiswing/components/index.html