Adatbázis rendszerek 2

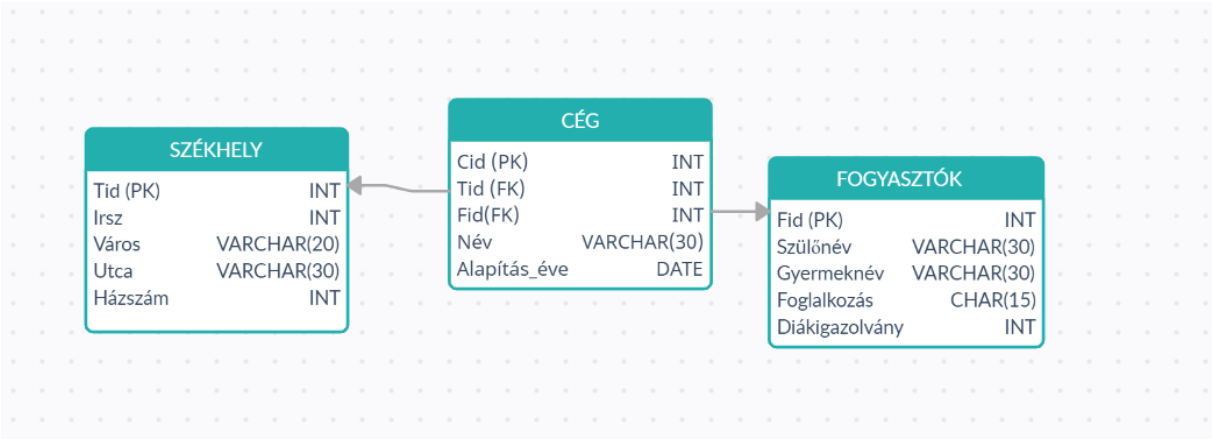
Féléves egyéni **JDBC** feladat
Egyéni oktatási cégek

Készítette:
Oravecz Áron D3U3EE

Gyakorlatvezető:
Bednarik László

Tartalom

Adatbázis modell:



Adatbázis jellemzése:

- Székhely jelen esetben 1:1 kapcsolatban áll a Céggel.
- A Fogyasztók a céggel pedig 1:N kapcsolatban állnak egymással.
- A Székhely és a Fogyasztókat kapcsolótáblával köthetjük össze a Cég tábla jelenlétében.
- **Székhely**:
  - A Cég tartózkodási helyét tartalmazza. Elsődleges kulcsként egy Tid-t alkalmazok, ami azonosító ként szolgál.
  - Irányítószám: ahol elhelyezkedik a székhely körzeti szinten, ugyanakkor egész szám típusú rekord.
  - Város és az utca szöveges típusúak.
  - A házszám numerikus számként funkcionál jelen esetben.
- **Fogyasztók**:
  - A Céget igénybe vevő csoportot foglalja magába.
  - Fid: azonosító és egyben elsődleges kulcsként működik. Ő tárolja el az összefüggő rekordok adatait.
  - Gyermeknév és a Szülőnév szöveges adattípusként tárolandó.
  - Diákigazolvány pedig numerikus adattípus.
- **Cég**:
  - A Cég azonosítóval rendelkezik.

- De ebben az esetben leginkább kapcsolótáblaként szolgál, ami összekapcsolja a Fogyasztókat a Székhelyekkel.
- A másik két táblának a kulcsait tartalmazza, így azok idegen kulcsként értelmezhetőek.
- A Cég tartalmaz egy szöveges adattípust, ami a Cég nevét tárolja.
- Tartalmaz egy Dátum típusú adatot is a tábla, ami az alapítás dátumát tartalmazza.

Elsőként a program felépítését gondoltam át. Program.java lett az a futtatható osztály, amin végbemegy minden utasítás és a Console-ban látom róla a vissza jelzéseket. (Ezeket a szöveges visszajelzéseket mostanra már kikommenteztem, mert nagyon láttam, hogy a program működik, így pedig már igazából csak zavaróak.) Illetve egy DbMethods.java osztályban az adatbázis metódusait tárolom, definiálom és ellenőrzöm nagy szét. A következő metódusokat valósítottam meg a program részére.

DbMethods.java Driver regisztráció, bejelentkezés és lekpacsolódás:

```
public static void DriverReg() { //Driver regisztráció
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("Sikeres driver regisztrálás\n");
    } catch (Exception ex) {
        System.err.println(ex.getMessage());
    }
}
```

Ezzel a metódussal ellenőrzöm a sikeres driver csatolását.

```
27
28      public Connection Connect(){ //Kapcsolódás és Bejelentkezés
29          Connection conn =null;
30          String url = "jdbc:oracle:thin:@193.6.5.58:1521:XE";
31          String user = "H22_d3u3ee";
32          String pwd = "D3U3EE";
33          try {
34              conn =  DriverManager.getConnection(url, user, pwd);
35              System.out.println("Sikeres kapcsolódás\n");
36          } catch (Exception e) {
37              System.err.println(e.getMessage());
38          }
39          return conn;
40      }
```

Majd a sikeres kapcsolódást és bejelentkezést véghez viszem és ellenőrzöm. Visszatérési értékként alkalmazom a ”conn„-t, mert szükségem lesz rá még a bekapcsolódásnál.

```
    public void DisConnect(Connection conn) { //Lekapcsolódás
        if (conn != null) {
            try {
                conn.close();
                System.out.println("Sikeres lekapcsolódás\n");
            } catch (Exception ex) {
                System.err.println(ex.getMessage());
            }
        }
    }
```

Az egyszerű parancs használatra írtam egy metódust, ami Statement-et hoz létre és ennek segít végrehajtom az SQL parancsokat.

Egyirányú utasítás kiadás

Ezt úgy lehet alkalmazni, hogy paraméterenként átadhatunk a metódusnak egy SQL típusú parancsot. Így egyfajta „egyirányú” utasítás kiadás történik.

```
53      public void ParancsExec(String command) {  //"egyirányú" parancsok kiadása Statment egyszerűsítése
54          Connection conn = Connect();
55          String sqlp = command;
56          System.out.println("Parancs: "+sqlp);
57          try {
58              Statement s = conn.createStatement();
59              s.execute(sqlp);
60              System.out.println(sqlp+"\n");
61              System.out.println("Sikeres Parancs");
62          } catch(SQLException e) {
63              System.out.println("ParancsExec: "+e.getMessage());
64          }
65          DisConnect(conn);
66      }
67  }
```

Program.java

Táblák létrehozása:

Székhely:

```
Sikeres driver regisztrálás

Sikeres kapcsolódás

Parancs: CREATE TABLE Szekhely (Tid INT PRIMARY KEY, Irsz INT,Varos CHAR(30), Utca CHAR(30), Hazszam NUMBER(10))
CREATE TABLE Szekhely (Tid INT PRIMARY KEY, Irsz INT,Varos CHAR(30), Utca CHAR(30), Hazszam NUMBER(10))
Sikeres lekapcsolódás

Sikeres kapcsolódás

Sikeres lekapcsolódás
```

Fogyasztók:

```
Parancs: CREATE TABLE Fogyasztok (Fid INT PRIMARY KEY, Szulonev CHAR(30),Gyermeknev CHAR(30), Foglalkozas CHAR(30),Diakigazolvany NUMBER(11))
CREATE TABLE Fogyasztok (Fid INT PRIMARY KEY, Szulonev CHAR(30),Gyermeknev CHAR(30), Foglalkozas CHAR(30),Diakigazolvany NUMBER(11))

Sikeres Parancs
```

Cég:

```
Parancs: CREATE TABLE Ceg (Cid INT PRIMARY KEY, Tid INT,Fid INT, Nev CHAR(30), Alapitas_eve DATE, FOREIGN KEY(Tid) REFERENCES Szekhely(Tid), FOREIGN KEY(Tid) REFERENCES Fogyasztok
CREATE TABLE Ceg (Cid INT PRIMARY KEY, Tid INT,Fid INT, Nev CHAR(30), Alapitas_eve DATE, FOREIGN KEY(Tid) REFERENCES Szekhely(Tid), FOREIGN KEY(Tid) REFERENCES Fogyasztok(Fid) )

Sikeres Parancs
```

Táblák feltöltése

Székhely:

```
String sqlp="INSERT INTO Szekhely VALUES(1, 1027, 'Budapest', 'Pákász utca','7')";
dbm.ParancsExec(sqlp);
```

```
Parancs: INSERT INTO Szekhely VALUES(1, 1027, 'Budapest', 'Pákász utca','7')
INSERT INTO Szekhely VALUES(1, 1027, 'Budapest', 'Pákász utca','7')

Sikeres Parancs
```

Fogyasztók:

```
String sqlp ="INSERT INTO Fogyasztok VALUES (1, 'Braskó Péter', 'Braskó Áron', 'Programozás', '7417175879')";
dbm.ParancsExec(sqlp);
```

```
Parancs: INSERT INTO Fogyasztok VALUES (1, 'Braskó Péter', 'Braskó Áron', 'Programozás', '7417175879')
INSERT INTO Fogyasztok VALUES (1, 'Braskó Péter', 'Braskó Áron', 'Programozás', '7417175879')

Sikeres Parancs
```

```
String sqlp ="INSERT INTO Fogyasztok VALUES (2, 'Szász Réjna', 'Farkas Ákos', 'Villamosmérnök', '7417175321')";
dbm.ParancsExec(sqlp);
```

```
Parancs: INSERT INTO Fogyasztok VALUES (2, 'Szász Réjna', 'Farkas Ákos', 'Villamosmérnök', '7417175321')
INSERT INTO Fogyasztok VALUES (2, 'Szász Réjna', 'Farkas Ákos', 'Villamosmérnök', '7417175321')

Sikeres Parancs
```

```
String sqlp ="INSERT INTO Fogyasztok VALUES (3, 'Kovács Anett', 'Tóth József', 'Rendszermérnök', '7417175765')";
dbm.ParancsExec(sqlp);
```

```
Parancs: INSERT INTO Fogyasztok VALUES (3, 'Kovács Anett', 'Tóth József', 'Rendszermérnök', '7417175765')
INSERT INTO Fogyasztok VALUES (3, 'Kovács Anett', 'Tóth József', 'Rendszermérnök', '7417175765')

Sikeres Parancs
```

```
 String sqlp ="INSERT INTO Fogyasztok VALUES (4, 'Puszta Tamás', 'Puszta Ildikó', 'Jogász', '7417175435')";
 dbm.ParancsExec(sqlp);
```

```
Parancs: INSERT INTO Fogyasztok VALUES (4, 'Puszta Tamás', 'Puszta Ildikó', 'Jogász', '7417175435')
INSERT INTO Fogyasztok VALUES (4, 'Puszta Tamás', 'Puszta Ildikó', 'Jogász', '7417175435')

Sikeres Parancs
```

```
String sqlp ="INSERT INTO Fogyasztok VALUES (5, 'Réti Norbert', 'Réti Mihály', 'Orvos', '7417175545')";
dbm.ParancsExec(sqlp);
```

```
Parancs: INSERT INTO Fogyasztok VALUES (5, 'Réti Norbert', 'Réti Mihály', 'Orvos', '7417175545')
INSERT INTO Fogyasztok VALUES (5, 'Réti Norbert', 'Réti Mihály', 'Orvos', '7417175545')

Sikeres Parancs
```

Cég:

```
 String sqlp ="INSERT INTO Ceg VALUES (1, 1, 1,'Logiscool', TO_DATE('20080521','YYYYMMDD'))";
 dbm.ParancsExec(sqlp);
```

```
Parancs: INSERT INTO Ceg VALUES (1, 1, 1,'Logiscool', TO_DATE('20080521','YYYYMMDD'))
INSERT INTO Ceg VALUES (1, 1, 1,'Logiscool', TO_DATE('20080521','YYYYMMDD'))

Sikeres Parancs
Sikeres lekapcsolódás
```

Adatok Kiírása:

Program.java:

```
dbm.ReadDataFogyaszto();
dbm.ReadDataCeg();
dbm.ReadDataSzekhely();
```

meghívom a függvényeket.

```
3       Kovács Anett                Tóth József                 Rendszermérnök              7417175765
4       Puszta Tamás                Puszta Ildikó               Jogász                      7417175435
5       Réti Norbert                Réti Mihály                 Orvos                       7417175545
2       Szász Réjna                 Farkas Ákos                 Villamosmérnök              7417175321
6       Mező Pista                  Mező Géza                   Programozás                 7417175546
1       Braskó Péter                Braskó Áron                 Programozás                 7417175879
Sikeres lekapcsolódás

Sikeres kapcsolódás

4       2       4       MavirDK                     2001-01-20
1       1       1       Logiscool                   2008-05-21
2       4       2       JogiSchool                  2010-12-25
3       5       3       MentőTisztiDE               2010-12-25
5       3       5       RendszerÜzemSchool          2020-03-09
Sikeres lekapcsolódás

Sikeres kapcsolódás

4       1043    Tárnok                      Bessenyei út                1
2       4080    Hajdunánás                  Hősök út                    18
3       3530    Miskolc                     József Attila út            40
1       1027    Budapest                    Pákász utca                 7
5       2076    Sopron                      Petőfi utca                 88
Sikeres lekapcsolódás

Sikeres kapcsolódás
```

DbMethods.java beolvasás

```
    public String ReadData(String s) {
        Scanner scanInput = new Scanner(System.in);
        String data = "";
        System.out.println(s);
        data = scanInput.nextLine();
        return data;
    }
```

beolvasási metódus. Ahhoz, hogy ne legyenek fölösleges vagy inkább bent maradt enterek, ahhoz a scanner változót lokálisan kellett létrehozni.

```
    public void ReadDataCeg() {
        String Nev="", x="\t";
        int Cid=0,Fid=0, Tid=0;
        Date Alapitas;
        String sqlp="SELECT Cid, Fid, Tid, Nev, Alapitas_eve FROM Ceg";
        Connection conn = Connect();
        try {
            Statement s = conn.createStatement();
            ResultSet rs = s.executeQuery(sqlp);
            while(rs.next()) {
                Cid= rs.getInt("Cid");
                Fid= rs.getInt("Fid");
                Tid= rs.getInt("Tid");
                Nev = rs.getString("Nev");  //Adatok beolvasása rekordonként ciklusba
                Alapitas = rs.getDate("Alapitas_eve");

                System.out.println(Cid+x+Fid+x+Tid+x+Nev+x+Alapitas); //kiírás
            }
            rs.close();
        }catch(SQLException e) {
            System.out.println("ReadDataCeg: "+e.getMessage());
        }
        DisConnect(conn);
    }
```

A Cég,Székhely és Fogyasztók beolvasásának metódusa

- Definiálunk kell a változók típusait és az sqlp String változóban definiálnunk kell a lekérdezést, ami jelen esetben a Cég összes adatát jelenti. Jelentősen figyelnünk

kell az adatok helyes sorrendjére és a megfelelő elnevezésű rekordokat hívjuk meg mert máskülönben a program híváskor hibát fog dobni számunkra. A ’try ’és ’ctach’ segítségével ellenőrizzük a megfelelő adatok beolvasását. Majd kiírtatjuk azokat. Ugyanezt a másik táblán is elvégezzük.

```
public void ReadDataFogyaszto() {
    String Sznev="", Gynev="", Foglalkozas="",x="\t";
    int Fid=0;
    long Digazolvany=0;

    String sqlp="SELECT Fid, Szulonev, Gyermeknev, Foglalkozas, Diakigazolvany FROM Fogyasztok";
    //CommandExec(sqlp);
    Connection conn = Connect();
    try {
        Statement s = conn.createStatement();
        ResultSet rs = s.executeQuery(sqlp);
        while(rs.next()) {
            Fid= rs.getInt("Fid");
            Sznev = rs.getString("Szulonev");  //Adatok beolvasása rekordonként ciklusba
            Gynev = rs.getString("Gyermeknev");
            Foglalkozas = rs.getString("Foglalkozas");
            Digazolvany= rs.getLong("Diakigazolvany"); //ITT A PROBLEM
            System.out.println(Fid+x+Sznev+x+Gynev+x+Foglalkozas+x+Digazolvany); //kiírás
        }
        rs.close();
    }catch(SQLException e) {
        System.out.println("ReadDataFogyaszto: "+e.getMessage());
    }
    DisConnect(conn);
}
```

```
public void ReadDataSzekhely() {
    String Varos="", Utca="", x="\t";
    int Tid=0,Irsz=0, Hazszam=0;
    String sqlp="SELECT Tid, Irsz, Varos, Utca, Hazszam FROM Szekhely";
    Connection conn = Connect();
    try {
        Statement s = conn.createStatement();
        ResultSet rs = s.executeQuery(sqlp);
        while(rs.next()) {
            Tid= rs.getInt("Tid");
            Irsz= rs.getInt("Irsz");
            Hazszam= rs.getInt("Hazszam");
            Varos = rs.getString("Varos");  //Adatok beolvasása rekordonként ciklusba
            Utca = rs.getString("Utca");

            System.out.println(Tid+x+Irsz+x+Varos+x+Utca+x+Hazszam); //kiírás
        }
        rs.close();
    }catch(SQLException e) {
        System.out.println("ReadDataSzekhely: "+e.getMessage());
    }
    DisConnect(conn);
}
```

Táblából való törlés:

```
public void DeleteDataFromFogyasztok() {
    Connection conn = Connect();
    System.out.println("Rekord törlése fogyasztó ID alapján: ");
    String fid = ReadData("Kérem a fogyasztó ID-ját: ");
    String sqlp = "DELETE From Fogyasztok WHERE Fid = '"+ fid +"'";
    try {
        Statement s = conn.createStatement();
        int db = s.executeUpdate(sqlp);
        if (db == 0) System.out.println("A megadott ID-val rendelkező fogyasztó nem létezik, ezért nem törölhető!");
        else System.out.println("Törlődött a(z) " + fid + " ID-val rendelkező fogyasztó!");
    } catch (SQLException e) {
        System.out.println("JDBC DeleteDataFromFogyasztok: "+ e.getMessage());
    } DisConnect(conn);
}
```

```
public void DeleteDataFromCeg() {
    Connection conn = Connect();
    System.out.println("Rekord törlése cég ID alapján: ");
    String cid = ReadData("Kérem a cég ID-ját: ");
    String sqlp = "DELETE From Ceg WHERE cid = '"+ cid +"'";
    try {
        Statement s = conn.createStatement();
        int db = s.executeUpdate(sqlp);
        if (db == 0) System.out.println("A megadott ID-val rendelkező cég nem létezik, ezért nem törölhető!");
        else System.out.println("Törlődött a(z) " + cid + " ID-val rendelkező cég!");
    } catch (SQLException e) {
        System.out.println("JDBC DeleteDataFromCeg: "+ e.getMessage());
    } DisConnect(conn);
}

public void DeleteDataFromSzekhely() {
    Connection conn = Connect();
    System.out.println("Rekord törlése székhely ID alapján: ");
    String tid = ReadData("Kérem a székhely ID-ját: ");
    String sqlp = "DELETE From Szekhely WHERE tid = '"+ tid +"'";
    try {
        Statement s = conn.createStatement();
        int db = s.executeUpdate(sqlp);
        if (db == 0) System.out.println("A megadott ID-val rendelkező székhely nem létezik, ezért nem törölhető!");
        else System.out.println("Törlődött a(z) " + tid + " ID-val rendelkező székhely!");
    } catch (SQLException e) {
        System.out.println("JDBC DeleteDataFromSzekhely: "+ e.getMessage());
    } DisConnect(conn);
}
```

Hasonlóképpen történik az adatok törlésének a metódusának a definiálása, mint a kilistázása. SQL parancsunk változik, illetve egyéb elágazások segítségével töröljük az adott rekordot. Ha a megadott rekord nem létezik hibával tér vissza.

Tábla módosítás

```java
public void UpdateDataCeg()
    {
        Connection conn = Connect();
        System.out.println("Rekord módosítása cég ID alapján: ");
        String id = ReadData("Kérem a cég ID-ját: ");
        String mezo = ReadData("Kérem a módosítani kívánt mező nevét: ");
        String ujertek = ReadData("Kérem az értéket, amelyre módosítani szeretné a régit: ");
        String sqlp = "UPDATE Ceg SET " + mezo + " = " + ujertek + " WHERE cid = '" + id + "'";
        try {
            Statement s = conn.createStatement();
            int db = s.executeUpdate(sqlp);
            if (db == 0) System.out.println("A módosítást nem lehet végrehajtani!");
            else System.out.println("A módosítás megtörtént, az adott ID-val rendelkező cégnél a(z) " + mezo + " új értéke " + ujertek);
        }catch (SQLException e) {
            System.out.println("JDBC UpdateDataCeg: " + e.getMessage());
        } DisConnect(conn);
    }


    public void UpdateDataSzekhely()
    {
        Connection conn = Connect();
        System.out.println("Rekord módosítása székhely ID alapján: ");
        String id = ReadData("Kérem a székhely ID-ját: ");
        String mezo = ReadData("Kérem a módosítani kívánt mező nevét: ");
        String ujertek = ReadData("Kérem az értéket, amelyre módosítani szeretné a régit: ");
        String sqlp = "UPDATE Szekhely SET " + mezo + " = " + ujertek + " WHERE tid = '" + id + "'";
        try {
            Statement s = conn.createStatement();
            int db = s.executeUpdate(sqlp);
            if (db == 0) System.out.println("A módosítást nem lehet végrehajtani!");
            else System.out.println("A módosítás megtörtént, az adott ID-val rendelkező székhelynél a(z) " + mezo + " új értéke " + ujertek);
        }catch (SQLException e) {
            System.out.println("JDBC UpdateDataSzekhely: " + e.getMessage());
        } DisConnect(conn);
    }

    public void UpdateDataFogyasztok()
    {
        Connection conn = Connect();
        System.out.println("Rekord módosítása fogyasztó ID alapján: ");
        String id = ReadData("Kérem a fogyasztó ID-ját: ");
        String mezo = ReadData("Kérem a módosítani kívánt mező nevét: ");
        String ujertek = ReadData("Kérem az értéket, amelyre módosítani szeretné a régit: ");
        String sqlp = "UPDATE Fogyasztok SET " + mezo + " = " + ujertek + " WHERE fid = '" + id + "'";
        try {
            Statement s = conn.createStatement();
            int db = s.executeUpdate(sqlp);
            if (db == 0) System.out.println("A módosítást nem lehet végrehajtani!");
            else System.out.println("A módosítás megtörtént, az adott ID-val rendelkező fogyasztónál a(z) " + mezo + " új értéke " + ujertek);
        }catch (SQLException e) {
            System.out.println("JDBC UpdateDataFogyasztok: " + e.getMessage());
        } DisConnect(conn);
    }
```

A rekordok módosítása a Fid alapján történik, ennek az érték ellenőrzésével és típus ismertetése alapján tudjuk módosítani.

A program bekéri a módosítandó rekord ID-ját, majd a módosítandó mező nevét és végül a mező új értékét.

Az executeUpdate megnézi hány rekordot érintett az adott SQL parancs.

Rekord beszúrása

```
public void InsertIntoSzekhely() {

    int check = -1;
    String tid = null, irsz = null, varos = null, utca = null, hazszam = null;
    Connection conn = Connect();

    System.out.println("Rekord adatainak beolvasása, rekord beszúrása a Székhely táblába:");


        try {
            Statement s = conn.createStatement();

            do{
                tid = ReadData("Kérem a székhely ID-ját: ");
                check = s.executeUpdate("SELECT tid FROM Szekhely WHERE tid = "+ tid);
                if(check == 1) System.out.println("Már létezik ilyen ID-val rendelkező székhely!");
            }while(check != 0);


            do {
                irsz = ReadData("Kérem a székhely irányítószámát: ");
                varos = ReadData("Kérem a székhely városát: ");
                utca = ReadData("Kérem a székhely utcáját: ");
                hazszam = ReadData("Kérem a székhely házszámát: ");

                check = s.executeUpdate("SELECT irsz, varos, utca, hazszam FROM Szekhely WHERE irsz = "+ irsz + ", varos = '"
                + varos + "', utca = '" + utca + "', hazszam = " + hazszam );
                if (check == 1) System.out.println("Ez a székhely már fel van véve az adatbázisba egy másik ID-val!");
            }while(check != 0);

                String sqlp = "INSERT INTO Szekhely Values ("+tid+", "+ irsz +", '"+ varos + "', '" + utca + "', " + hazszam + ")";
                s.execute(sqlp);
                System.out.println("A rekord beszúrása megtörtént!");
            } catch(SQLException e) {
                System.out.println("JDBC InsertIntoSzekhely: " + e.getMessage());
            } DisConnect(conn);
    }
```

A székhely tábla esetén először is definiáljuk a változókat. Majd a try catch segítségével kezeljük a kivételeket. Majd ciklusok segítségével és néhány elágazással ellenőrizzük, hogy a beszúrással nem e okozunk redundanciát vagyis ismétlődő elemeket nem e hozunk létre.

```
public void InsertIntoFogyasztok() {

    int check = -1;
    int checkMore = -1;
    String szulonev = null, gyermeknev = null, foglalkozas = null, fid = null, diakigazolvany = null;
    Connection conn = Connect();

    System.out.println("Rekord adatainak beolvasása, rekord beszúrása a fogyasztók táblába:");


        try {
            Statement s = conn.createStatement();

            do{
                fid = ReadData("Kérem a fogyasztó ID-ját: ");
                check = s.executeUpdate("SELECT fid FROM Fogyasztok WHERE fid = '"+ fid +"'");
                if(check == 1) System.out.println("Már létezik ilyen ID-val rendelkező fogyasztó!");
            }while(check != 0);

                szulonev = ReadData("Kérem a fogyasztó szülőjének a nevét: ");
                gyermeknev = ReadData("Kérem a fogyasztó nevét: ");
                foglalkozas = ReadData("Kérem a fogyasztó foglalkozását: ");


            do{
                diakigazolvany = ReadData("Kérem a fogyasztó diákigazolványát: ");
                check = s.executeUpdate("SELECT fid FROM Fogyasztok WHERE fid = '"+ fid +"'");
                if(check == 1) System.out.println("Már létezik ilyen ID-val rendelkező fogyasztó!");
                checkMore = diakigazolvany.length();
                //System.out.println(checkMore);
                if(checkMore != 11) System.out.println("A diákigazolvány 11 számjegyből áll!");
            }while(check != 0 || checkMore != 11);

                String sqlp = "INSERT INTO Fogyasztok Values ("+fid+", '"+ szulonev +"', '"+ gyermeknev + "', '" + foglalkozas + "', " + diakigazolvany + ")";
                s.execute(sqlp);

                System.out.println("A rekord beszúrása megtörtént!");
            } catch(SQLException e) {

                System.out.println("JDBC InsertIntoFogyasztok: " + e.getMessage());
            } DisConnect(conn);
    }
```

A fogyasztók táblába való rekord beszúrás esetén is először ellenőrizzük, hogy a bekért fogyasztói ID már létezik e. Ezt követően, ha nem létezik ilyen akkor elkezdjük bekérni a felhasználótól az adatokat. Majd a végén a diákigazolványt kérjük be és ellenőriztetjük, ha ugyan azt adja a redundanciát megelőzve visszadobjuk, ha túl hosszú vagy nagyon rövid akkor visszadobjuk és újat kéretünk be, amíg nem lesz megfelelő a méret és típusa. A végén jelezzük, hogy a rekord sikeresen be lett szúrva táblába.

```
public void InsertIntoCeg() {

    int check = -1;
    String cid = null, tid = null, fid = null, nev = null, alapitas = null;
    Connection conn = Connect();

    System.out.println("Rekord adatainak beolvasása, rekord beszúrása a cég táblába:");


        try {
            Statement s = conn.createStatement();
            int checkMore;

            do{
                cid = ReadData("Kérem a cég ID-ját: ");
                check = s.executeUpdate("SELECT cid FROM Ceg WHERE cid = "+ cid);
                if(check == 1) System.out.println("Már létezik ilyen ID-val rendelkező cég!");
            }while(check != 0);

            do {
                checkMore = 0;

                do{
                    tid = ReadData("Kérem a székhely ID-ját: ");
                    check = s.executeUpdate("SELECT tid FROM Szekhely WHERE tid = " + tid);
                    if(check == 0) System.out.println("Nem létezik ilyen ID-val rendelkező székhely!");
                }while(check != 1);

                do {
                    fid = ReadData("Kérem a fogyasztó ID-ját: ");
                    check = s.executeUpdate("SELECT fid FROM Fogyasztok WHERE fid = " + fid);
                    if(check==0) System.out.println("Nem létezik ilyen ID-val rendelkező fogyasztó!");
                }while(check!=1);

                checkMore = s.executeUpdate("SELECT tid, fid FROM Ceg WHERE tid = " + tid + " AND fid = " + fid);
                if(checkMore==1) System.out.println("E között a székhely és fogyasztó között már létezik kapcsolat!");

            }while(checkMore!=0);

         do {

             nev = ReadData("Kérem a cég nevét: ");

             check = s.executeUpdate("SELECT nev FROM Ceg WHERE nev = '"+ nev + "'");

             if (check == 1) System.out.println("Ilyen névvel már van felvéve cég az adatbázisba!");

         }while(check != 0);

         alapitas = ReadData("Kérem az alapítás dátumát: (a dátumot kérlek ilyen formában add meg: 2015-05-10)");

             String sqlp = "INSERT INTO Ceg Values ("+ cid +", "+ tid +", "+ fid + ", '" + nev + "', TO_DATE('" + alapitas + "', 'YYYY-MM-DD'))";
             s.execute(sqlp);
             System.out.println("A rekord beszúrása megtörtént!");
         } catch(SQLException e) {
             System.out.println("JDBC InsertIntoCeg: " + e.getMessage());
         } DisConnect(conn);
}
```

Számomra a legnehezebb és legbonyolultabb metódus a Cég táblánál a rekord beszúrás volt. Mivel itt már a dátum formátumának az ellenőrzésével és bevitelével kellett foglalkozni, de mindezek előtt bekértem a felhasználótól az ID-t

ellenőriztem létezik e már, ha igen visszadobtam, majd az idegen kulcsokkal folytattam, ami a Tid és a Fid volt. Itt is vizsgáltam a redundancia lehetőségét vagy éppen, ha nem létezett olyan ID akkor azt dobtam vissza hiba üzenetként. Ha egy Fogyasztó és egy Székhely között már van kapcsolat, akkor a program nem hagyja, hogy létrehozzuk ugyanezt a kapcsolatot egy különböző ID-val. Ezek után a cég nevét ellenőrizzük, amit megad a felhasználó. Az utolsó ellenőrzés pedig a dátumra hivatkozik, ami egy megadott formátum alapján érvényes csak, de ezt feltüntetjük a felhasználó számára, majd a to_date segítségével átalakítjuk, hogy a jdbc is rendesen tudja kezelni a dátumot, mivel egy string-ként adjuk meg és date típussá kell alakítani.

Menü kezelés:

```
public void menu() {
    System.out.println("\n");
    System.out.println("MENÜ:");
    System.out.println("\n");


    int m, n;
    int ok;

    do{
        ok = 0;
        System.out.println("Melyik táblával szeretnél foglalkozni?");
        System.out.println("\n");
        System.out.println("0 Kilépés");
        System.out.println("1 Fogyasztók");
        System.out.println("2 Székhely");
        System.out.println("3 Cég");

        m = ReadInt("Add meg a választott tábla számát: ");

        if(m>-1 && m<4) ok = 1;
        else System.out.println("Kérlek a felsorolt számok közül válassz!");
    }while(ok != 1);

    if (m == 0)
    {
        System.out.println("A program leállt!");
        System.exit(0);
    }
```

```
            switch(m)
            {
                case 11: ReadDataFogyaszto(); break;
                case 12: InsertIntoFogyasztok(); break;
                case 13: DeleteDataFromFogyasztok(); break;
                case 14: UpdateDataFogyasztok(); break;
                case 21: ReadDataSzekhely(); break;
                case 22: InsertIntoSzekhely(); break;
                case 23: DeleteDataFromSzekhely(); break;
                case 24: UpdateDataSzekhely(); break;
                case 31: ReadDataCeg(); break;
                case 32: InsertIntoCeg(); break;
                case 33: DeleteDataFromCeg(); break;
                case 34: UpdateDataCeg(); break;

            }

        }

    }
```

Létrehoztunk egy végtelen ciklust a Program osztályban.

```
while (1 != 0) {
    dbm.menu();
}
```

A DbMethods-ban létrehoztam a menüt és abban egy kiskaput, ami jelen esetben a ’0’, ami megállítja a programot. Első esetben megkérdezi a felhasználót, melyik táblával szeretne foglalkozni, majd ezt ellenőriztük, hogy a megfelelő intervallumban kaptuk e a számot. Ha nem, akkor újra megkérdezi a felhasználót. Utána megkérdezi a felhasználót, hogy mit szeretne csinálni a kiválasztott táblával.

```
do {
    ok = 0;
    System.out.println("Milyen műveletet szeretnél végezni a táblán?");
    System.out.println("\n");
    System.out.println("0 Kilépés");
    System.out.println("1 Listázás");
    System.out.println("2 Beszúrás");
    System.out.println("3 Törlés");
    System.out.println("4 Módosítás");

    n = ReadInt("Add meg a választott művelet számát?");

    if(n>=-1 && n<5) ok=1;
    else System.out.println("Kérlek a felsorolt számok közül válassz!");
}while (ok != 1);

if(n == 0)
{
    System.out.println("A program leállt!");
    System.exit(0);
}
else
{
    m = m*10 + n;
```

Ezek után switch-case

szerkezet segítségével a program kiválasztja a megfelelő metódust, elvégzi, majd újraindul a menü.

Bejelentkezés

```
public int LogIn() {
    int in = 0, check = 0;
    String user = null, passwd = null;
    Connection conn = Connect();


    user = ReadData("Van már fiókod? (igen/nem)");
    if(user.equals("igen"))
    {try {
        Statement s = conn.createStatement();
        do{
            user = ReadData("Felhasználónév: ");
            check = s.executeUpdate("SELECT felhasznalonev FROM Bejelentkezes WHERE felhasznalonev = '" + user + "'");
            if(check == 0) System.out.println("Ilyen felhasználónévvel nem létezik felhasználó!");
        }while(check != 1);
        do {
            passwd = ReadData("Jelszó: ");
            check = s.executeUpdate("SELECT felhasznalonev, jelszo FROM Bejelentkezes WHERE felhasznalonev = '" + user + "' AND jelszo = '" + passwd + "'");
            if(check == 0) System.out.println("Hibás jelszó!");
        }while(check != 1);
        in = 1;
        return in;
    }catch(SQLException e) {
        System.out.println("JDBC Bejelentkezes: " + e.getMessage());
    }}
    else if(user.equals("nem"))
    {
        String reg = ReadData("Szeretnél regisztrálni? (igen/nem)");
        if(reg.equals("nem"))
        {
            System.out.println("Rendben, viszlát!");
            System.exit(0);
        }

         .
         else if(reg.equals("igen")) {
             int jo = 0;
         try {
             Statement s = conn.createStatement();
             do{
                 user = ReadData("Adj meg egy felhasználónevet: ");
                 jo = s.executeUpdate("SELECT felhasznalonev FROM Bejelentkezes WHERE felhasznalonev = '" + user + "'");
                 if(jo != 0) System.out.println("Ez a felhasználónév már foglalt!");
             }while(jo!=0);

             do{
                 passwd = ReadData("Adj meg egy jelszót: ");
                 String passwd2 = ReadData("Add meg újra a jelszavadat: ");
                 if (passwd.equals(passwd2))
                 {
                     ParancsExec("INSERT INTO Bejelentkezes VALUES ('" + user + "', '" + passwd + "')");
                     jo = 1;
                     System.out.println("Sikeres regisztráció!");
                 }
                 else System.out.println("Nem egyezik a két jelszó!");
             }while(jo != 1);
             return in;

         }catch(SQLException e) {
             System.out.println("Regisztracio: " + e.getMessage());
         }
         }
     }   return in;
}
```

A bejelentkezés úgy történik, hogy a program megkérdezi, hogy van-e már fiókod, ekkor lehetőséged van választani igen és nem között. Ha már van, akkor bekéri a felhasználónevet és jelszót, amikkel, ha talál egyezést a Bejelentkezés táblában, akkor 1-et fog visszaadni, ez majd a Program osztályban hozzáírtak miatt lesz fontos.

Ha azt válaszolod, hogy nem, akkor megkérdezi, hogy szeretnél-e regisztrálni. Itt is választhatsz igen és nem közül. Ha azt válaszolod, hogy nem, akkor a program leáll és

elköszön, ha viszont azt válaszolod, hogy igen, akkor kér egy felhasználónevet, ami nem egyezhet a többivel a táblában, illetve kétszer kéri a jelszót, amelyek, ha nem egyeznek, akkor újra kéri a jelszót. Ha a regisztráció megtörtént, akkor kiírja, hogy „Sikeres regisztráció” és visszadob a „Van már fiókod?” kérdésre.

A bejelentkezés és menü megvalósulása a Program.java-ban

```
int k = dbm.LogIn();
do{
    if (k == 1)
    {
        System.out.println("Sikeres bejelentkezés!");
    }
    else {
        k = dbm.LogIn();
        if(k == 1) System.out.println("\nSikeres bejelentkezés!");
    }
}while(k != 1);
while (1 != 0) {
    dbm.menu();
}
```

A k változó tárolja a bejelentkezés metódusa által visszaadott értéket. Ezután egy ciklus ellenőrzi, hogy milyen értékkel tér vissza. Ha 1-gyel, akkor kiírja, hogy „Sikeres bejelentkezés” és tovább dob a menüre, ha azonban nem 1 a visszatérési értéke, akkor újra a LogIn (a bejelentkezés) metódussal találkozunk, mindaddig, amíg nem sikerül bejelentkeznünk.