



Java Programozás

4. óra



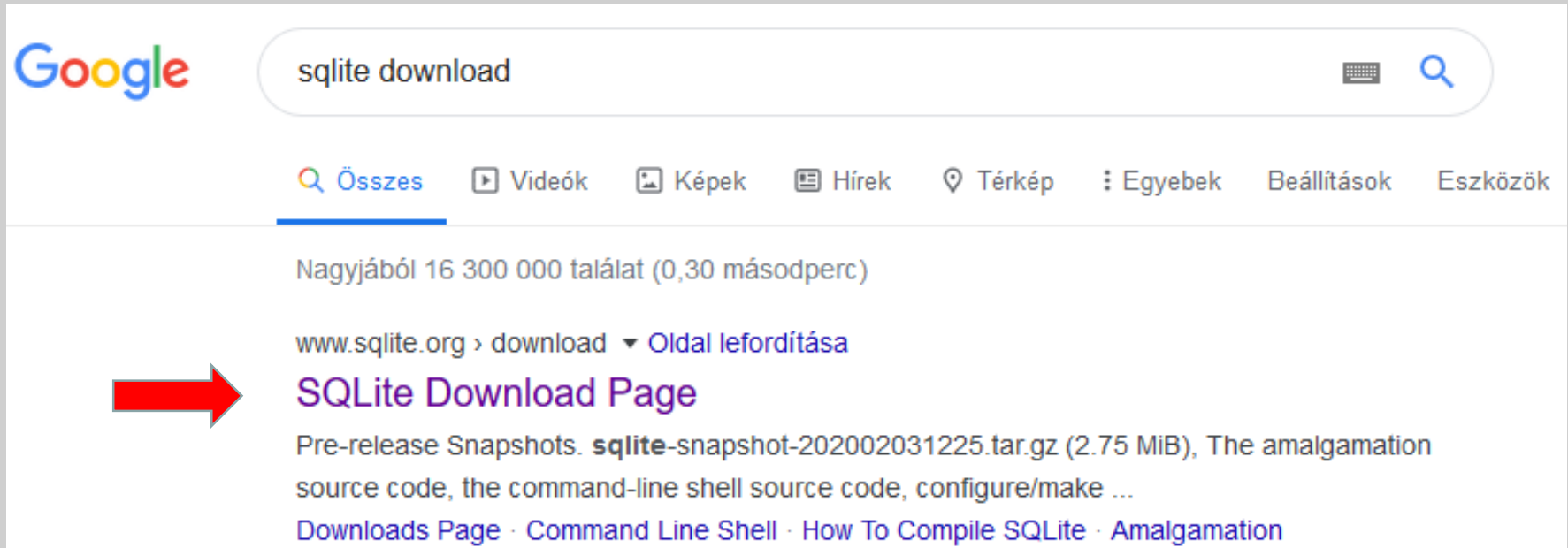
BITMAN

v: 2020.04.04

JDBC

- Alakítsuk át az adatkezelő projektünket úgy, hogy **adatbázist kezelve működjön**.
- Használjuk a leglazább adatbázis motort, az SQLite-ot, mely:
 - egyetlen fájlból áll,
 - aprócska, helyfoglalása néhány megabájt,
 - rendkívül egyszerű telepíteni,
 - távolról is elérhető, létezik hozzá JDBC driver

■ Letöltése



Google

sqlite download

Összes Videók Képek Hírek Térkép Egyebek Beállítások Eszközök

Nagyjából 16 300 000 találat (0,30 másodperc)

www.sqlite.org > download ▾ [Oldal lefordítása](#)

SQLite Download Page

Pre-release Snapshots. **sqlite-snapshot-202002031225.tar.gz** (2.75 MiB), The amalgamation source code, the command-line shell source code, configure/make ...

[Downloads Page](#) · [Command Line Shell](#) · [How To Compile SQLite](#) · [Amalgamation](#)

■ Letöltése:



The screenshot shows a web browser window with the title "SQLite Download Page". The address bar contains the URL "http://www.sqlite.org/download.html". The page header features the SQLite logo on the left and the tagline "Small. Fast. Reliable. Choose any three." on the right. A dark blue navigation bar contains links for Home, About, Documentation, Download, License, Support, and Purchase, along with a search bar. The main content area is titled "SQLite Download Page" and has a subheading "Precompiled Binaries for Windows". Below this, there are three download options, each with a red arrow pointing to it. The first two options are for DLLs (32-bit and 64-bit), and the third is for a bundle of command-line tools. Each option includes the filename, a description, a SHA1 hash, and the file size.

SQLite Download Page

<http://www.sqlite.org/download.html>

Small. Fast. Reliable.
Choose any three.

Home About Documentation Download License Support Purchase Search

SQLite Download Page

Precompiled Binaries for Windows

sqlite-dll-win32-x86-3220000.zip (438.67 KiB)	32-bit DLL (x86) for SQLite version 3.22.0. (sha1: e316282a617c5f0c40c488de79c73cf13c8baaf2)
sqlite-dll-win64-x64-3220000.zip (729.28 KiB)	64-bit DLL (x64) for SQLite version 3.22.0. (sha1: 94402e914b0caaacc7b5f9d8f41c6f6adb0fc0d7)
sqlite-tools-win32-x86-3220000.zip (1.62 MiB)	A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff.exe program, and the sqlite3_analyzer.exe program. (sha1: 9b0e0a6dc63601f2ddb2028f44547d65b2da7d27)

SQLite



■ JDBC driver letöltése:

The screenshot shows a Google search interface. The search bar contains the text "sqlite jdbc driver download". Below the search bar, there are tabs for "Összes", "Képek", "Videók", "Hírek", "Vásárlás", "Egyebek", "Beállítások", and "Eszközök". The "Összes" tab is selected. Below the tabs, it says "Nagyjából 294 000 találat (0,32 másodperc)". A red arrow points to the first search result, which is a link to "bitbucket.org > xerial > sqlite-jdbc > downloads". The link is followed by "Oldal lefordítása" and "xerial / sqlite-jdbc / Downloads — Bitbucket". Below this, there is a table with columns "Name", "Size", "Uploaded by", "Date", and "Download repository". The table contains one row with the following data: "sqlite-jdbc-3.27.2.1.jar", "5.7 MB", "Taro L. Saito", "59991", "2019-03-20".

Google

sqlite jdbc driver download

Összes Képek Videók Hírek Vásárlás Egyebek Beállítások Eszközök

Nagyjából 294 000 találat (0,32 másodperc)

bitbucket.org > xerial > sqlite-jdbc > downloads ▼ Oldal lefordítása

xerial / sqlite-jdbc / Downloads — Bitbucket

Name, Size, Uploaded by, **Downloads**, Date, **Download** repository, 107.8 MB ... 2019-12-23.

sqlite-jdbc-3.27.2.1.jar, 5.7 MB, Taro L. Saito, 59991. 2019-03-20.

SQLite



■ JDBC driver letöltése:

xerial / sqlite-jdbc / Downloads X +

<https://bitbucket.org/xerial/sqlite-jdbc/downloads/>

java

sqlite-jdbc

Overview

Source

Commits

Branches

Pull requests

Pipelines

Issues

Wiki

Downloads

Taro L. Saito / sqlite-jdbc







Downloads

Downloads Tags Branches

Name	Size	Uploaded by	Downloads	Date
Download repository	107.8 MB			
sqlite-jdbc-3.21.0.jar	6.4 MB	xerial	20563	2017-11-14
sqlite-jdbc-3.20.1.jar	6.3 MB	xerial	998	2017-11-14
sqlite-jdbc-3.20.0.jar	6.3 MB	xerial	23819	2017-08-05
sqlite-jdbc-3.19.3.jar	5.9 MB	xerial	11740	2017-06-23
sqlite-jdbc-3.18.0.jar	5.9 MB	xerial	10584	2017-05-18
sqlite-jdbc-3.16.1.jar	5.9 MB	xerial	33214	2017-01-11

■ Telepítés:

- Hozzunk létre egy könyvtárat az adatbázis kezelő számára, és csomagoljuk ki oda a **dll-t** és a **tools-t**. Kész!
- Ugyanide csomagoljuk ki oda a **jdbc** drivert is. Kész!
- A mappa tartalma:

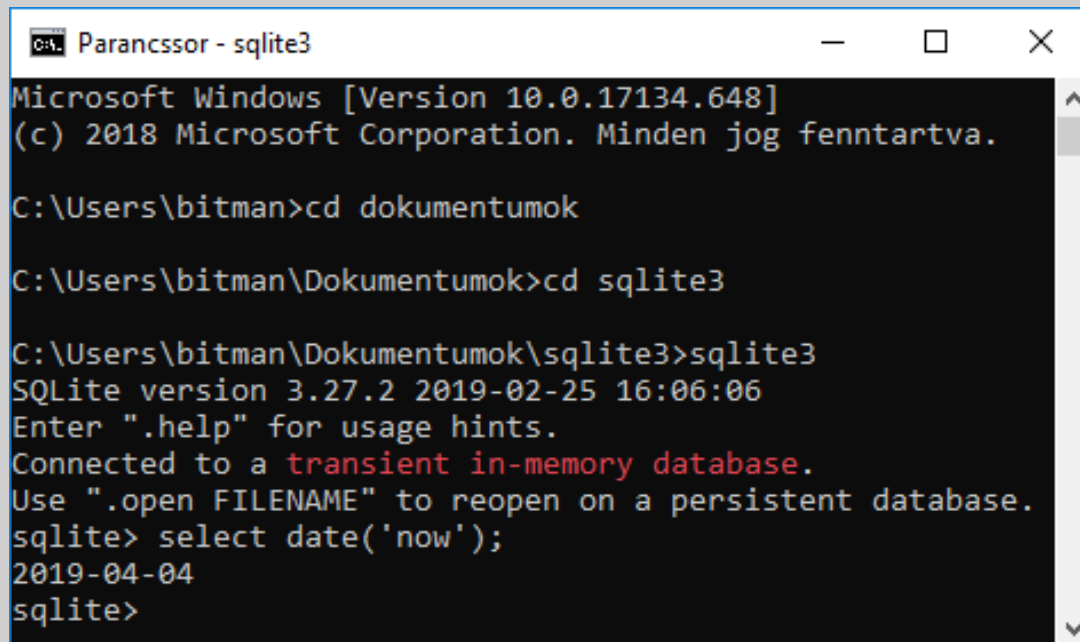
 sqldiff.exe	2019. 02. 25. 17:33	Alkalmazás	481 KB
 sqlite3.def	2019. 02. 26. 1:42	DEF fájl	6 KB
 sqlite3.dll	2019. 02. 26. 1:43	Alkalmazáskiterjes...	1 838 KB
 sqlite3.exe	2019. 02. 25. 17:34	Alkalmazás	898 KB
 sqlite3_analyzer.exe	2019. 02. 25. 17:33	Alkalmazás	1 959 KB
 sqlite-jdbc-3.27.2.1.jar	2019. 04. 04. 16:15	Executable Jar File	5 872 KB

■ Kipróbálás:

- Indítsuk el a Parancssort
- Lépünk át az adatbázis könyvtárba.
- Indítsuk el az adatbázist: `sqlite3`
- Írassuk ki a rendszer dátumot:

```
select date('now');
```

- Működik!
- Help: `.help`
- Kilépés: `.quit`

A screenshot of a Windows Command Prompt window titled "Parancssor - sqlite3". The window shows the following text:

```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. Minden jog fenntartva.

C:\Users\bitman>cd dokumentumok

C:\Users\bitman\Dokumentumok>cd sqlite3

C:\Users\bitman\Dokumentumok\sqlite3>sqlite3
SQLite version 3.27.2 2019-02-25 16:06:06
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> select date('now');
2019-04-04
sqlite>
```


■ Adatbázis létrehozása

- Ha az `sqlite3` paranccsal elindítjuk a szerveret, akkor az csak a memóriában kezeli az adatokat.
- Ahhoz, hogy az adatok megmaradjanak, létre kell hozni egy adatbázist.
- A létrehozott adatbázisfájl a munkakönyvtárba kerül, neve az lesz, amit megadtunk (lehet tetszőleges a kiterjesztése!)
- Adjuk ki a `.open empdb` parancsot, így létrejön az **empdb** adatbázis.
- Ha nem kapunk hibaüzenetet, a parancs rendben lefutott!

```
sqlite> .open empdb
sqlite>
```

■ Tábla létrehozása

- Megnyitni egy adatbázis a `.open név` paranccsal lehet
- A létrehozott / megnyitott adatbázisban gépeljük be a következő parancsot:

```
create table emp(  
kod integer primary key,  
nev text,  
szulido date,  
lakohely text,  
fizetes integer);
```

```
sqlite> create table emp(  
...> kod integer primary key,  
...> nev text,  
...> szulido date,  
...> lakohely text,  
...> iq integer);  
sqlite>
```

- Több sorba beírt parancsnál a `...>` jelzi, hogy a parancs még nem fejeződött be.
- A parancs végét a `;` jelzi az értelmezőnek
- A beírt parancs végén **ENTER**
- Ha nem kapunk hibaüzenetet, a parancs rendben lefutott!

■ Adatbevitel

– Írjuk be a következő parancsokat:

```
insert into emp values(31,"Jég Elek","1985.05.06","Miskolc",225000);  
insert into emp values(32,"Rossz Géza","1981.11.09","Miskolc",180000);  
insert into emp values(33,"Keksz Zoé","1993.02.13","Eger",411000);
```

■ Írassuk ki az adatainkat:

– select * from emp;

```
sqlite> insert into emp values(31,"Jég Elek","1985.05.06","Miskolc",112);  
sqlite> insert into emp values(32,"Rossz Géza","1981.11.09","Miskolc",105);  
sqlite> insert into emp values(33,"Keksz Zoé","1993.02.13","Eger",125);  
sqlite> select * from emp;  
31|Jég Elek|1985.05.06|Miskolc|112  
32|Rossz Géza|1981.11.09|Miskolc|105  
33|Keksz Zoé|1993.02.13|Eger|125  
sqlite>
```

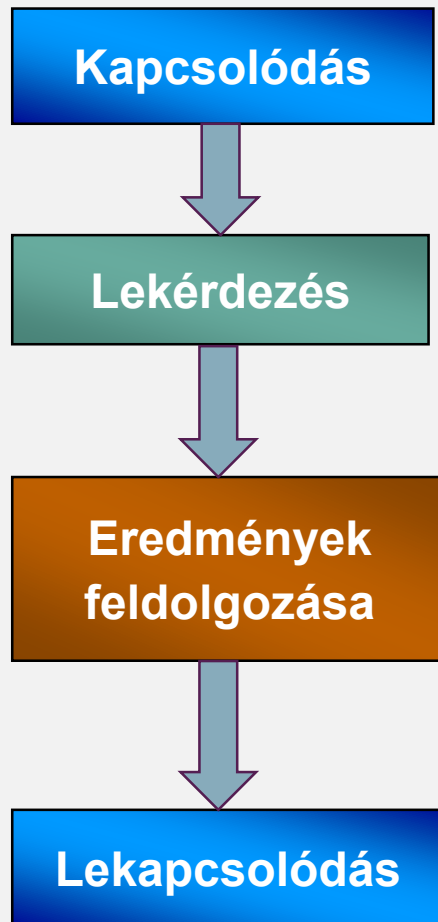
■ Kész! Jöhet a távoli adatelérés JDBC-vel!

■ Lépünk ki (**.quit**) és zárjuk be a Parancssort!



Java
JDBC

Az adatbázis-programozás lépései



Driver regisztrálása

Kapcsolódás a DBMS-hez

SQL kérés (**STATEMENT**)
összeállítása

SQL kérés elküldése

Az eredményhalmaz (**CURSOR**)
rekordonkénti bejárása

Az értékek átadása
programváltozóknak

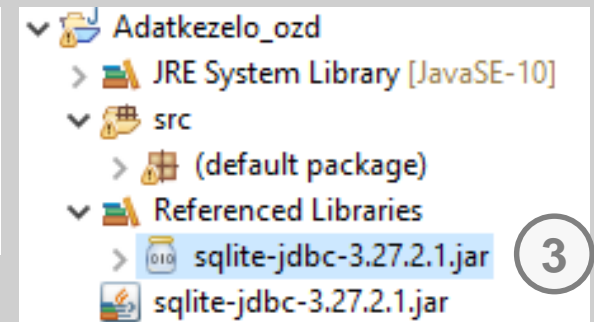
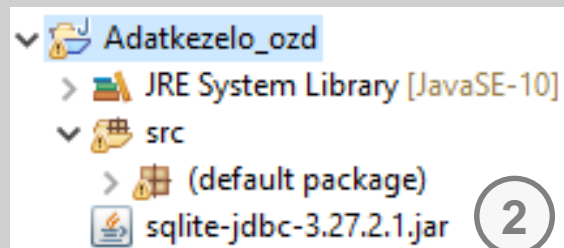
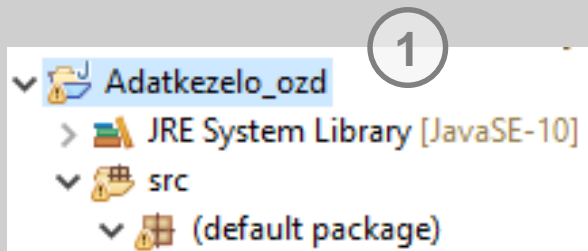
Eredményhalmaz lezárása

SQL kérés lezárása

Kapcsolat lezárása

A driver betöltése a projektbe

1. Az Intézőben nyissuk meg az SQLite program mappáját.
2. Kattintsunk az [sqlite-jdbc.x.x.x.jar](#) fájl nevére
3. Másoljuk a vágólapra a kijelölt drivert: **CTRL** + **C**
- 1 4. Az Eclipse-ben kattintsunk a projektünk nevére (Adatkezelő)
- 2 5. Másoljuk ide a drivert: **CTRL** + **V**
6. Adjuk hozzá a drivert a projekt elérési útvonalához:
 1. Jobb klikk a bemásolt driver nevére
 2. Build Path
 3. Add to Build Path



Adatbázis-kezelés Eclipsben

- Az adatbázis-kezelő metódusok számára hozzunk létre egy új osztályt:
 - File\New\Class – Neve: DbMethods
- Az ide kerülő metódusok:
 - Driver regisztráció
 - Kapcsolódás
 - Lekapcsolódás
 - Lekérdezés
 - Beszúrás
 - Módosítás
 - Törlés

Írjuk meg az adatbázis kezelés alapmetódusait

- Sokat fogunk kiírogatni, ezért másoljuk ide a Checker osztályból (vagy írjuk meg) az **SM** metódust!

```
public class DbMethods {  
    private Statement s = null;  
    private Connection conn=null;  
    private ResultSet rs = null;
```

```
    public void Reg(){  
        try {  
            Class.forName("org.sqlite.JDBC");  
            SM("Sikeres driver regisztráció!", 1);  
        } catch (ClassNotFoundException e) {  
            SM("Hibás driver regisztráció!" + e.getMessage(), 0);  
        }  
    }
```


```
    public void SM(String msg, int tipus) {  
        JOptionPane.showMessageDialog(null, msg, "Program üzenet", tipus);  
    }
```

A mindig szükséges változókat célszerű globálisan deklarálni.

Az elérési útvonalon megkeresi, és a memóriába tölti az sqlite jdbc drivert.

Írjuk meg az adatbázis kezelés alapmetódusait

Az útvonalat mindenki aktualizálja!



```
public void Connect() {  
    try {  
        String url = "jdbc:sqlite:C:/Users/bitman/Documents/sqlite3/empdb";  
        conn = DriverManager.getConnection(url);  
        SM("Connection OK!", 1);  
    } catch (SQLException e) {  
        SM("JDBC Connect: "+e.getMessage(), 0);  
    }  
}
```

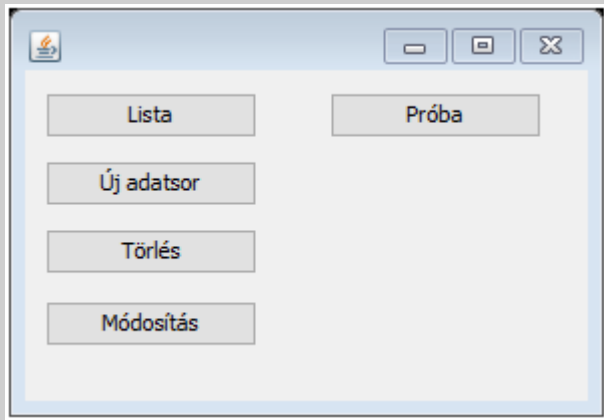
Kapcsolódik, és visszaadja a kapcsolat változót (conn).

```
public void Disconnect(){  
    try {  
        conn.close();  
        SM("DisConnection OK!", 1);  
    } catch (SQLException e) {SM(e.getMessage(), 0);}  
}
```

Lezárja a kapcsolat változót

Program

- Rakjunk ideiglenesen egy nyomógombot a program ablakába (**Próba**), és rakjuk alá a kapcsolódás, lekapcsolódás kódját.
- Mivel a drivert regisztrálni csak egyszer kell, ezért annak kódja a panelt létrehozó konstruktor elejére kerül, így minden indításkor le fog futni.



Program

- Adjunk eseménykezelőt a gombhoz (duplakatt), és írjuk meg a szükséges kódokat:

```
public class Program extends JFrame {  
  
    private JPanel contentPane;  
    private EmpTM etm;  
    private DbMethods dbm = new DbMethods();
```

```
public Program() {  
    dbm.Reg();  
  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    setBounds(100, 100, 297, 204);
```

```
JButton btnPrba = new JButton("Pr\u00F3ba");  
btnPrba.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent arg0) {  
        dbm.Connect();  
        dbm.DisConnect();  
    }  
});
```

Próbáljuk ki a programunkat!

Ha elakadtál küldj
ímélt a help@help.com
címmre!

A man with a bald head and a light blue shirt is sitting at a desk, looking extremely stressed. He has his hand on his forehead and a wide-eyed, frantic expression. In front of him is a silver laptop. The laptop screen is bright blue and displays the text 'HIBA :-)' in white, bold, sans-serif capital letters. The background is a plain, light-colored wall. There are some papers and a pen on the desk behind the laptop.

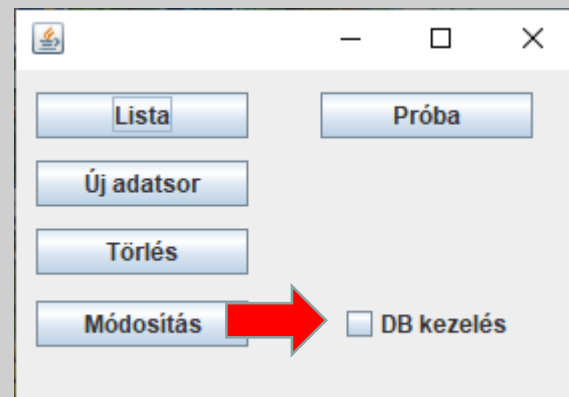
HIBA :-)

Vagy nézz így!

Program

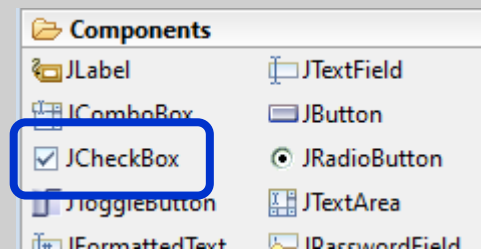
■ Rakjunk egy jelölőnégyzetet a panelra. Jelentése:

- Ha be van kapcsolva adatbázist kezelünk
- Ha ki van kapcsolva, akkor fájl



■ Design nézetben tegyünk a panelra egy JCheckBox-ot:

- Szövege: DB kezelés
- Adjunk hozzá akciókezelőt:
 1. Jobb klikk a jejlőnégyzetre
 2. Add event handler
 3. action
 4. action Performed



Program

- Írjuk meg a jelölőnégyzet kezelését:
 - Az állapotának figyelésére vezessünk be egy változót:

```
public class Program extends JFrame {  
  
    private JPanel contentPane;  
    private EmpTM etm;  
    private DbMethods dbm = new DbMethods();  
    private int dbkez = 0;
```



- Írjuk meg a jelölőnégyzet akció-kezelőjét:

```
JCheckBox chckbxDbKezels = new JCheckBox("DB kezel\u00E9s");  
chckbxDbKezels.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        if (chckbxDbKezels.isSelected()) dbkez = 1;  
        else dbkez = 0;  
    }  
});
```

A beolvasás kódja a DbMethods osztályban


```
public EmpTM ReadAllData() {
    Object emptmn[] = {"Jel", "Kód", "Név", "Szülidő", "Lakóhely", "Fizetés"};
    EmpTM etm = new EmpTM(emptmn, 0);
    String nev="", szid="", lak="";
    int kod=0, fiz=0;
    String sqlp= "select kod,nev,szulido,lakohely,fizetes from emp";
    try {
        s = conn.createStatement();
        rs = s.executeQuery(sqlp);
        while(rs.next()) {
            kod = rs.getInt("kod");
            nev = rs.getString("nev");
            szid = rs.getString("szulido");
            lak = rs.getString("lakohely");
            fiz = rs.getInt("fizetes");
            etm.addRow(new Object[]{false, kod, nev, szid, lak, fiz});
        }
        rs.close();
    } catch (SQLException e) {SM(e.getMessage(), 0);}
    return etm;
}
```

Program

- Írjuk meg a listázás kódját:

```
JButton btnLista = new JButton("Lista");
btnLista.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        if (dbkez == 0) etm = FileManager.CsvReader();
        else {
            dbm.Connect();
            etm = dbm.ReadAllData();
            dbm.DisConnect();
        }
        EmpList el = new EmpList(Program.this, etm);
        el.setVisible(true);
    }
});
```

- Próbáljuk ki a programot:



Jel	Kód	Név	Születő	Lakóhely	Fizetés
<input type="checkbox"/>	31	Jég Elek	1985.05.06	Miskolc	112
<input type="checkbox"/>	32	Rossz Géza	1981.11.09	Miskolc	105
<input type="checkbox"/>	33	Keksz Zoé	1993.02.13	Eger	125

Bezár

Új rekord beszúrása

- Először írjuk meg a **DbMethods** osztályba az **Insert** metódust:

```
public void Insert(String kod, String nev, String szid, String lak, String fiz) {  
    String sqlp = "insert into emp values("+kod+", '"+nev+"', '"+szid+"', '"+lak+"', '"+fiz+"");  
    try {  
        s = conn.createStatement();  
        s.execute(sqlp);  
        SM("insert OK!", 1);  
    } catch (SQLException e) {  
        SM("JDBC insert: "+e.getMessage(), 0);  
    }  
}
```

Új rekord beszúrása

- A **NewEmp** panelnek tudnia kell, hogy fájlba, vagy adatbázisba kell beszúrnia, ezért mikor elindítjuk, át kell neki adni a **dbkez** változó értékét.
 - Paramétert a konstruktorban lehet átadni, ezért módosítsuk a **NewEmp** osztály konstruktorát:



```
public NewEmp(int dbkez) {  
    getContentPane().setBackground(new Color(72, 209, 204));  
    setBounds(100, 100, 241, 240);  
    getContentPane().setLayout(null);  
}
```

- Az adatbázis eléréséhez szükség lesz egy **DbMethods** példányra:


```
public class NewEmp extends JDialog {  
    private JTextField kod;  
    private Checker c = new Checker();  
    private DbMethods dbm = new DbMethods();  
}
```



Új rekord beszúrása

- Módosítsuk a **NewEmp** panel **Beszúrás** gombjának kódját:


```
JButton btnBeszr = new JButton("Besz\u00FAr");
btnBeszr.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (c.goodInt(kod, "Kód"))
            if (c.filled(nev, "Név"))
                if (c.goodDate(szid, "Születési idő"))
                    if (c.filled(lak, "Lakcím"))
                        if (c.goodInt(fiz, "Fizetés"))
                            if (dbkez == 0) {
                                FileManager.Insert(RTF(kod), RTF(nev),
                                    RTF(szid), RTF(lak), RTF(fiz));
                            } else {
                                dbm.Connect();
                                dbm.Insert(RTF(kod), RTF(nev),
                                    RTF(szid), RTF(lak), RTF(fiz));
                                dbm.DisConnect();
                            }
                        }
                    }
                }
            }
        }
    });
```



Program

- A Programban át kell adnunk a **dbkez** változó értékét a **NewEmp** panelnek. Módosítsuk az **Új adatsor** gomb kódját:

```
 JButton btnUjAdat = new JButton("\u00DAj adatsor");  
 btnUjAdat.addActionListener(new ActionListener() {  
     public void actionPerformed(ActionEvent e) {  
         NewEmp ne = new NewEmp(dbkez);  
         ne.setVisible(true);  
     }  
 });
```



- Kész! Próbáljuk beszúrni egy rekordot az adatbázisba!

Rekord törlése


- Először írjuk meg a `DbMethods` osztályba a `DeleteData` metódust:

```
public void DeleteData(String kod) {  
    String sqlp = "delete from emp where kod="+kod;  
    try {  
        s = conn.createStatement();  
        s.execute(sqlp);  
        SM("Delete OK!", 1);  
    } catch (SQLException e) {  
        SM("JDBC Delete: "+e.getMessage(), 0);  
    }  
}
```

Rekord törlése


- Módosítsuk a **EmpDel** osztály konstruktorát:

```
public EmpDel(JFrame f, EmpTM betm, int dbkez) {  
    super(f, "Dolgozók törlése", true);  
    etm = betm;  
}
```



- Az adatbázis eléréséhez szükség lesz egy **DbMethods** példányra:

```
private final JPanel contentPanel = new JPanel();  
private JTable table;  
private EmpTM etm;  
private Checker c = new Checker();  
private DbMethods dbm = new DbMethods();
```



Rekord törlése

- Módosítsuk a **EmpDel** osztályban a **Törlés** gomb kódját:

```

JButton btnAdatsorTrlse = new JButton("Adatsor t\u00F6rl\u00E9se");
btnAdatsorTrlse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int db=0, jel=0, x=0;
        for(x = 0; x < etm.getRowCount(); x++)
            if ((Boolean)etm.getValueAt(x,0)) {db++; jel=x;}
        if (db==0) c.SM("Nincs kijelölve a törlendő rekord!",0);

        if (db>1) c.SM("Több rekord van kijelölve!\nEgyszerre csak egy"
                        + " rekord törölhető!",0);



        if (db==1) {
            String kod = etm.getValueAt(jel, 1).toString();
            etm.removeRow(jel);
            if (dbkez==0) FileManager.Insert(etm);
            else {
                dbm.Connect();
                dbm.DeleteData(kod);
                dbm.DisConnect();
            }
            dispose();
            c.SM("A rekord törölve!",1);
        }
    }
});
```

Program

- A Programban át kell adnunk a **dbkez** változó értékét az **EmpDel** panelnek. A panel elindítása előtt az adatokat is be kell tölteni az adatbázisból:

```

JButton btnTrls = new JButton("T\u00F6rl\u00E9s");
btnTrls.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (dbkez == 0) etm = FileManager.CsvReader();
        else {
            dbm.Connect();
            etm = dbm.ReadAllData();
            dbm.DisConnect();
        }
        EmpDel ed = new EmpDel(Program.this, etm, dbkez);
        ed.setVisible(true);
    }
});
```



- Kész! Próbáljuk törölni egy rekordot az adatbázisból!

Adatmódosítás

- Olyan metódust kell írunk, amely képes bármelyik mezőt módosítani.
 - Kihasználjuk, hogy az adatbázisokban mindenféle adattípus módosítható szöveges adatként.
- Egy olyan metódust írunk, melynek bemenő paraméterei:
 - A rekord kódja
 - A módosítandó mező neve
 - A módosítandó mező értéke
- A módosításokat a rekordok kódja alapján fogjuk elvégezni, ezért ha több adatot módosítanak, és köztük van a kód is, akkor a kódot kell utoljára módosítani!

Adatmódosítás


- Írjuk meg a `DbMethods` osztályba az `Update` metódust:

```
public void Update(String kod, String mnev, String madat) {  
    String sqlp = "update emp set "+mnev+ "='"+madat+"' where kod="+kod;  
    try {  
        s = conn.createStatement();  
        s.execute(sqlp);  
        SM("Update OK!", 1);  
    } catch (SQLException e) {  
        SM("JDBC Update: "+e.getMessage(), 0);  
    }  
}
```

Adatmódosítás


- Módosítsuk az **EmpMod** osztály konstruktorát:

```
public EmpMod(JFrame f, EmpTM betm, int dbkez) {  
    super(f, "Dolgozók módosítása", true);  
    etm = betm;  
}
```



- Az adatbázis eléréséhez szükség lesz egy **DbMethods** példányra:

```
public class EmpMod extends JDialog {  
  
    private final JPanel contentPanel = new JPanel();  
    private JTable table;  
    private EmpTM etm;  
    private Checker c = new Checker();  
    private DbMethods dbm = new DbMethods();  
}
```



Adatmódosítás

- A fájlban tárolt adatok módosítása úgy működik, hogy először módosítjuk a táblázatban az adatokat, utána kiírjuk az összes adatot fájlba.
- Ha ezt a logikát követnénk adatbázis esetén is, és módosítanánk egy rekord kódját, aztán próbálnánk meg az adatbázisban módosítani a rekordot, akkor elveszítenénk a kód mező eredeti értékét.
- Ezért adatbázis esetén előbb módosítjuk az adatokat az adatbázisban, és utána a látható táblázatban.

Adatmódosítás

- Az EmpMod osztály Módosítás gombjának kódja: (1. rész)

```

JButton btnModosit = new JButton("M\u00F3dos\u00EDt\u00E1s");
btnModosit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int db=0, jel=0, x=0;
        for(x = 0; x < etm.getRowCount(); x++)
            if ((Boolean)etm.getValueAt(x,0)) {db++; jel=x;}
        ① if (db==0) c.SM("Nincs kijelölve a módosítandó rekord!",0);

        if (db>1) c.SM("Több rekord van kijelölve!\nEgyszerre csak egy"
            + " rekord módosítható!",0);

        if (db==1) {
            if (modDataPc() > 0) {
                boolean ok = true;
                if (c.filled(kod)) ok = c.goodInt(kod, "Kód");
                ② if (ok && c.filled(fiz)) ok = c.goodInt(fiz, "Fizetés");
                if (ok) {
                    if (dbkez == 1) {
                        String mkod = etm.getValueAt(jel, 1).toString();
                        dbm.Connect();
                        if (c.filled(nev)) dbm.Update(mkod, "nev", c.RTF(nev));
                        if (c.filled(szip)) dbm.Update(mkod, "szulido", c.RTF(szip));
                        if (c.filled(lak)) dbm.Update(mkod, "lakohely", c.RTF(lak));
                        if (c.filled(fiz)) dbm.Update(mkod, "fizetes", c.RTF(fiz));
                        if (c.filled(kod)) dbm.Update(mkod, "kod", c.RTF(kod));
                        dbm.DisConnect();
                    }
                }
            }
        }
    }
});
        ③ →
```

Adatmódosítás

- Az **EmpMod** osztály **Módosítás** gombjának kódja: (2. rész)

```

    ④ if (c.filled(kod)) etm.setValueAt(c.stringToInt(c.RTF(kod)), jel, 1);
      if (c.filled(nev)) etm.setValueAt(c.RTF(nev), jel, 2);
      if (c.filled(szip)) etm.setValueAt(c.RTF(szip), jel, 3);
      if (c.filled(lak)) etm.setValueAt(c.RTF(lak), jel, 4);
      if (c.filled(fiz)) etm.setValueAt(c.stringToInt(c.RTF(fiz)), jel, 5);

    ⑤ if (dbkez == 0) FileManager.Insert(etm);

      c.SM("A rekord módosítva!",1);
      reset(jel);
    }
  } else {
    c.SM("Nincs kitöltve egyetlen módosító adatmező sem!",1);
  }
}
});
```

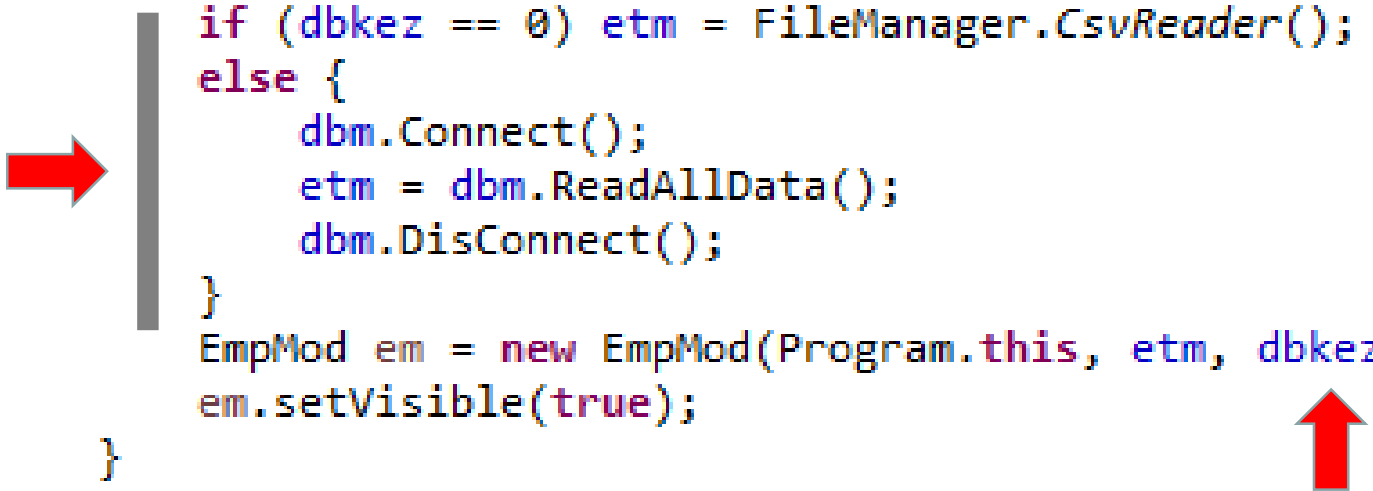
1. Táblázatban kijelölt sorok vizsgálata
2. Módosító mezők kitöltöttségének, tartalmának ellenőrzése
3. Adatbázis módosítása (csak ha van módosító adat a kijelölt rekordban)
4. Táblázat adatainak módosítása
5. Fájl tartalmának módosítása (ha szükséges)

Program

- A Programban át kell adnunk a **dbkez** változó értékét az **EmpMod** panelnek. A panel elindítása előtt az adatokat is be kell tölteni az adatbázisból:

```

JButton btnMdosts = new JButton("M\u00F3dos\u00EDt\u00E1s");
btnMdosts.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (dbkez == 0) etm = FileManager.CsvReader();
        else {
            dbm.Connect();
            etm = dbm.ReadAllData();
            dbm.DisConnect();
        }
        EmpMod em = new EmpMod(Program.this, etm, dbkez);
        em.setVisible(true);
    }
});
```



The diagram illustrates the data flow in the code. A red arrow points from the `etm = FileManager.CsvReader();` line to a vertical grey bar. From the bottom of this bar, another red arrow points down to the `dbkez` parameter in the `EmpMod em = new EmpMod(Program.this, etm, dbkez);` line. A third red arrow points up to the `dbkez` parameter in the same line from below.

- Kész! Próbáljuk adatokat módosítani az adatbázisban!

Ha elakadtál küldj
ímélt a help@help.com
címmre!

Hol a hiba?

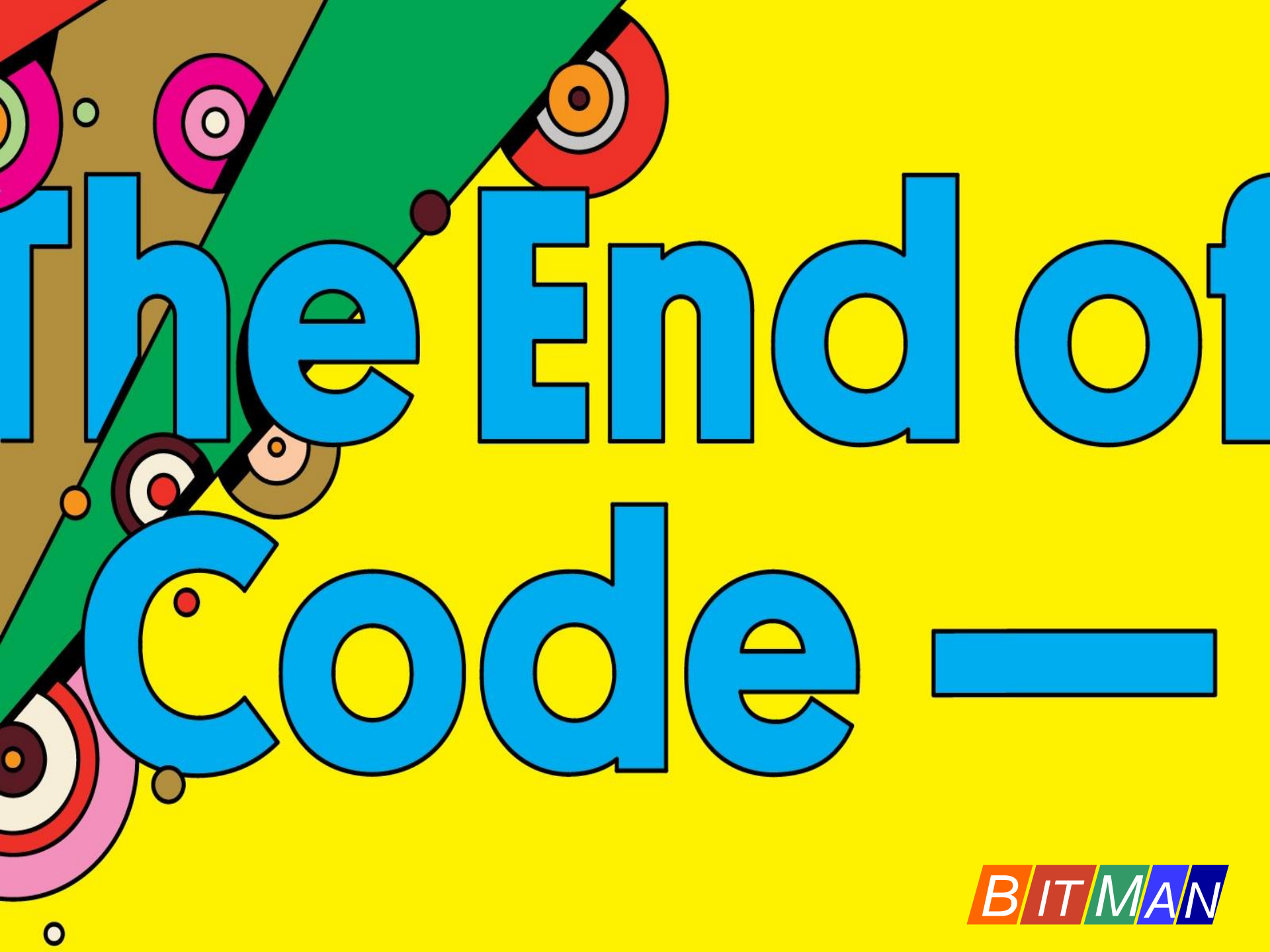


Vagy keress
segítséget a
neten!





Question
van?



The End of Code —

VÉGE

