2) Készítsen asztali alkalmazást, amely a könyvtát működését segíti.

Elérhető pontszám: 10 pont

Elért pontszám: pont

1) A program jelenítsen meg, ízlésen, két ListBox-ot és két gombot (Visszahozva és Bezár).

2) A Bezár gomb megnyomására az alkalmazás lépjen ki.

3) A Visszahozva gomb indításkor legyen inaktív.

4) A felhasználó választhat a felsorolt nevek közül. (Minta1)

5) Ha van választott név, akkor töltődjen be az ügyfél által kölcsönzött könyvek listája.

6) Ha van kijelölt könyv akkor váljon aktívvá a Visszahozva feliratú gomb.

7) A Visszahozva gomb megnyomása esetén törölje a könyvet (az adatbázisban) a

kikölcsönözzek közül.

**Megoldás:**

Csináljunk Vscode-val egy java projektet az első feladatban leírt módon. /library\_helper\_gui/

A *lib* mappába tegyük bele a mariadb drivert, úgy, mint az első feladatnál.

Hozzunk létre egy új java fájlt az *src* mappán belül és nevezzük el MainFrame-nek.

Örököltessük a JFrame-től:

public class MainFrame extends JFrame {

}

Hozzuk létre a JList-eket és a gombokat:

public class MainFrame extends JFrame {

    JList<Kolcsonzok> kolcsonzokList = new JList<>();

  JList<Kolcsonzesek> kolcsonzesekList = new JList<>();

    Button visszahozva = new Button("Visszahozva");

    Button bezar = new Button("Bezár");

}

Készítsünk egy metódust, ami létrehozza az ablakot.

private void createWindowAndElements() {

        this.kolcsonzokList.setBounds(20, 20, 200, 500);

        this.kolcsonzesekList.setBounds(240, 20, 500, 170);

        this.setLayout(null);

        this.kolcsonzokList.setBorder(BorderFactory.createLineBorder(Color.GRAY, 1));

        this.kolcsonzesekList.setBorder(BorderFactory.createLineBorder(Color.GRAY, 1));

        this.add(kolcsonzokList);

        this.add(kolcsonzesekList);

        this.visszahozva.setBounds(240, 220, 80, 20);

        this.bezar.setBounds(680, 500, 60, 20);

        this.add(visszahozva);

        this.add(bezar);

        this.visszahozva.setEnabled(false);

        bezar.addActionListener((ActionEvent e) -> {

            dispose();

        });

        this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        this.setSize(800, 600);

        this.getRootPane().setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

        this.setTitle("Könytár\_GUI");

        this.setResizable(false);

        this.setVisible(true);

    }

A konstruktorból meghívjuk:

public MainFrame() {

        createWindowAndElements();

    }

Az ablak kész.

Hozzuk létre a Kolcsonzok és a Kolcsonzesek osztályokat a konstruktorral együtt. /*src mappában*/

Generáljuk le a getter és setter metódusokat: jobb klikk->Source action->Generate getters and setters

public class Kolcsonzok {

    int id;

    String nev;

    String szulIdo;

    public Kolcsonzok(int id, String nev, String szulIdo) {

        this.id = id;

        this.nev = nev;

        this.szulIdo = szulIdo;

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public String getNev() {

        return nev;

    }

  @Override

    public String toString() {

        return nev;

    }

    public void setNev(String nev) {

        this.nev = nev;

    }

    public String getSzulIdo() {

        return szulIdo;

    }

    public void setSzulIdo(String szulIdo) {

        this.szulIdo = szulIdo;

    }

}

public class Kolcsonzesek {

    int id;

    int kolcsonzoId;

    String iro;

    String mufaj;

    String cim;

    public Kolcsonzesek(int id, int kolcsonzoId, String iro, String mufaj, String cim) {

        this.id = id;

        this.kolcsonzoId = kolcsonzoId;

        this.iro = iro;

        this.mufaj = mufaj;

        this.cim = cim;

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public int getKolcsonzoId() {

        return kolcsonzoId;

    }

    public void setKolcsonzoId(int kolcsonzoId) {

        this.kolcsonzoId = kolcsonzoId;

    }

    public String getIro() {

        return iro;

    }

@Override

    public String toString() {

        return cim + " - " + iro + " ( " + mufaj + " ) ";

    }

    public void setIro(String iro) {

        this.iro = iro;

    }

    public String getMufaj() {

        return mufaj;

    }

    public void setMufaj(String mufaj) {

        this.mufaj = mufaj;

    }

    public String getCim() {

        return cim;

    }

    public void setCim(String cim) {

        this.cim = cim;

    }

}

Hozzunk létre egy Service.java fájlt az *src* mappában.

Készítsünk egy metódust, ami felépíti az adatbázis kapcsolatot. Ua. mint az előző feladatban.:

private Connection getConnection() throws SQLException {

        String user = "root";

        String password = "";

        String url = "jdbc:mariadb://localhost:3306/konyvtar";

        return DriverManager.getConnection(url, user, password);

    }

Azért rakjuk külön függvénybe mert három helyről is meg kell majd hívni.

Indítsuk el a XAMPP-ot azon belül az *Apache*-tés a *MySQL*-t.

A *Service.java* fájlba hozzunk létre két *ArrayList*-et az adatok tárolására.

ArrayList<Kolcsonzok> kolcsonzokLista = new ArrayList<>();

ArrayList<Kolcsonzesek> kolcsonzesekLista = new ArrayList<>();

int kolcsonzoId;

Ezek fogják tárolni az adatbázisból lekért adatokat.

Írjuk meg a metódust, ami lekéri a MySQL-től a kölcsönzők adatait.

public ArrayList<Kolcsonzok> getKolcsonzok() throws SQLException {

        Connection conn = getConnection();

        String sql = "select \* from kolcsonzok";

        PreparedStatement stmt = conn.prepareStatement(sql);

        ResultSet rs = stmt.executeQuery();

        while (rs.next()) {

            Kolcsonzok kolcsonzok = new Kolcsonzok(

                    rs.getInt("id"),

                    rs.getString("nev"),

                    rs.getString("szulIdo"));

            kolcsonzokLista.add(kolcsonzok);

        }

        return kolcsonzokLista;

    }

Ugyanúgy létrehozzuk a kölcsönzések lekérését is, annyi a különbség hogy csak azokat kérjük le amelyeknek a kolcsonzoId-je megegyezik annak a kölcsönzőnek az id-jével amire kattintottunk.

public ArrayList<Kolcsonzesek> getKolcsonzesek(int id) throws SQLException {

        Connection conn = getConnection();

        String sql = "select \* from kolcsonzesek where kolcsonzoId = " + id;

        PreparedStatement stmt = conn.prepareStatement(sql);

        ResultSet rs = stmt.executeQuery();

        while (rs.next()) {

            Kolcsonzesek kolcsonzesek = new Kolcsonzesek(

                    rs.getInt("id"),

                    rs.getInt("kolcsonzoId"),

                    rs.getString("cim"),

                    rs.getString("iro"),

                    rs.getString("mufaj"));

            kolcsonzesekLista.add(kolcsonzesek);

        }

        return kolcsonzesekLista;

    }

Végül létrehozzuk a törlő metódust is.

public void torol(int id) throws SQLException {

        Connection conn = getConnection();

        String sql = "delete from kolcsonzesek where id = " + id;

        PreparedStatement stmt = conn.prepareStatement(sql);

        stmt.executeUpdate();

    }

Vissza a *MainFrame.java*-ba.

Írjuk meg a kölcsönzők lekérését.

private void getKolcsonzok() throws SQLException {

        DefaultListModel<Kolcsonzok> kolcsonzokListModel = new DefaultListModel<>();

        ArrayList<Kolcsonzok> kolcsonzok = new Service().getKolcsonzok();

        for (Kolcsonzok kolcsonzokObj : kolcsonzok) {

            kolcsonzokListModel.addElement(kolcsonzokObj);

        }

        kolcsonzokList.setModel(kolcsonzokListModel);

        kolcsonzokList.addListSelectionListener((ListSelectionEvent e) -> {

            if (!e.getValueIsAdjusting()) {

                Kolcsonzok selectedKolcsonzok = kolcsonzokList.getSelectedValue();

                if (selectedKolcsonzok != null) {

                    try {

                        getKolcsonzesek(selectedKolcsonzok.getId());

                    } catch (SQLException ex) {

                        ex.printStackTrace();

                    }

                }

            }

        });

    }

Majd a kölcsönzések-ét is.

private void getKolcsonzesek(int id) throws SQLException {// ez kéri le a db-ből a kölcsönzött könyveket

        visszahozva.setEnabled(false);// a visszahozva gombot inaktívvá tesszük hogy amíg a lekérdezés folyik ne

                                      // nyomogassák

        DefaultListModel<Kolcsonzesek> kolcsonzesekListModel = new DefaultListModel<>();

        ArrayList<Kolcsonzesek> kolcsonzesek = new Service().getKolcsonzesek(id);

        for (Kolcsonzesek kolcsonzesekObj : kolcsonzesek) {

            kolcsonzesekListModel.addElement(kolcsonzesekObj);

        }

        kolcsonzesekList.setModel(kolcsonzesekListModel);

    }

Írjuk meg a függvényt, ami figyeli, hogy ki van-e választva könyv, amit vissza kell adni.

private void initEvents() {

        kolcsonzesekList.addListSelectionListener((ListSelectionEvent f) -> {

            if (!f.getValueIsAdjusting()) {

                Kolcsonzesek selectedKolcsonzesek = kolcsonzesekList.getSelectedValue();

                if (selectedKolcsonzesek != null) {

                    System.out.println(selectedKolcsonzesek.getId());

                    visszahozva.setEnabled(true);

                }

            }

        });

        visszahozva.addActionListener((ActionEvent e) -> {

            int id = kolcsonzesekList.getSelectedValue().getId();

            int kolcsonzoId = kolcsonzesekList.getSelectedValue().getKolcsonzoId();

            System.out.println("töröl " + id);

            try {

                new Service().torol(id);

                getKolcsonzesek(kolcsonzoId);

            } catch (SQLException e1) {

                e1.printStackTrace();

            }

        });

    }

Hát „röviden” ennyi. Ebből, ami mindenképp kell:

* adatbázis kapcsolat létrehozása
* sql lekérések futtatása
* eseményfigyelők beállítása

Nem hiszem, hogy a vizsgán JList-es feladat lesz. Ezt a típusú feladatot a végére kell hagyni mert talán ez a legbonyolultabb. Ha a többi kész akkor érdemes utána ezzel szenvedni.