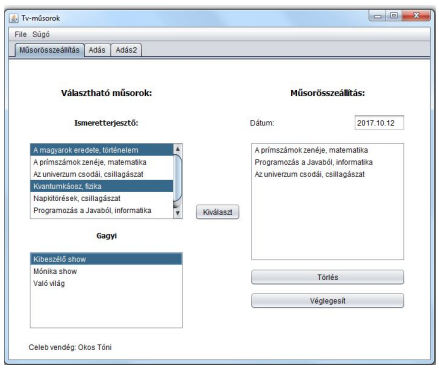
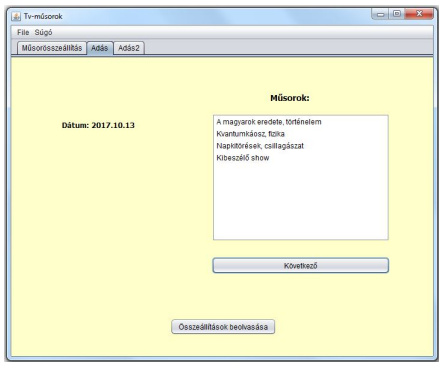
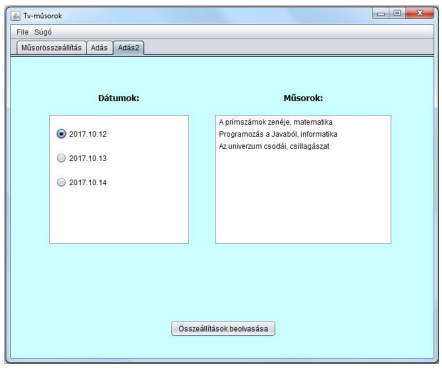
Megoldásvázlat a 6. gyakorlat feladatához

Feladat:

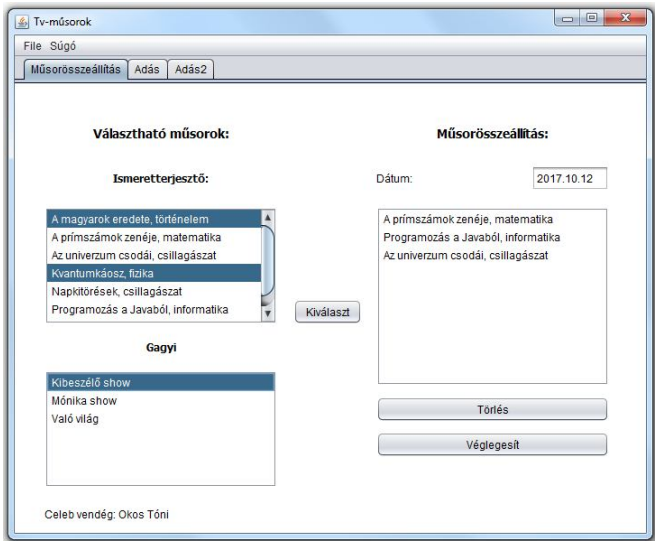
Segítsük a tévéműsorok összeállítását egy Java programmal!







A 700x500-as belső méretű panelek egyikén lehet összeállítani az adásba kerülő műsorokat, valamelyik másikon pedig „megnézni” azokat.



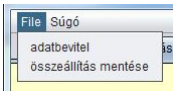
Kétféle műsort gyártanak, mindkettőt a címe és a műsoridő hossza (elég egy int, mondjuk ennyi perc) jellemzi. Utólag már nem vágnak a műsorból, de a műsor minden adata lekérdezhető. Minden műsort *néznek()*. Ekkor a nézők száma eggyel növekszik. Minden műsornak van valamilyen kiszámítható, egész számmal mért *hatása()*. (Mondjuk, ennyivel növekszik vagy csökken a társadalom össz okossága. ☺) Az alaphatás minden műsor esetén azonos a műsoridő hosszának és a néző-számnak a szorzatával.

Az ismeretterjesztő műsoroknál azt is meg kell adni, hogy az illető műsor milyen tudományághoz tartozik, a műsor hatása pedig az alaphatás és egy, az összes ismeretterjesztő műsorra egyformán érvényes szorzónak a szorzata.

A népbutító gagyi műsor mindig egy celeb körül forog. Ez azt jelenti, hogy a műsorhoz elengedhetetlen egy celeb, ugyanakkor szükség esetén ki lehet cserélni egy másikra, illetve az is előfordulhat, hogy csak a műsor létrejötte után találják ki, hogy ki legyen ez a celeb. Nyilván csak akkor hat, ha van benne celeb, ekkor a hatása egy negatív érték: az alaphatás -1-szerese és egy, a gagyi műsorokra jellemző egységes szorzó, osztva a celeb IQ-jával.

A celeb nevével és IQ-jának értékével definiálható. Az IQ nem változik az élet során.

Tesztelje az alaposztályokat!



A megfelelő menüpont hatására adatbázisból olvassuk be a szükséges adatokat. (Az adatbázis két táblája: MUSOR, illetve CELEB.)

Beolvasás után minden gagyi műsorhoz rendeljünk egy véletlen celebet, (egy celeb akár több műsorban is szerepelhet), majd értelemszerűen névsorba rendezetten jelenítsük meg a lehetséges műsorokat a listafelületen.

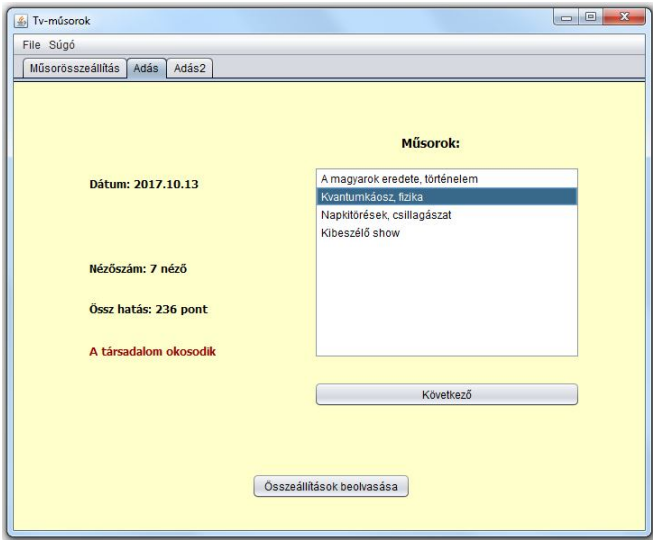
A majdan adásba kerülő műsort úgy lehet összeállítani, hogy a kijelölt műsorok a Kiválaszt gomb hatására bekerülnek az összeállított műsort tartalmazó listába. (Ha tudja, megoldhatja egyetlen gombbal is.) Természetesen figyeljen arra, hogy minden műsor csak egyszer szerepelhet az aktuális kínálatban.
Legalább annyival küzdjünk a butaság ellen, hogy az ismeretterjesztő műsorok közül egyszerre többet is ki lehet választani, a népbutítók közül csak egyet. Az utóbbi listafelület alatt megjelenik a kijelölt műsor celebjének neve is.

A törlés gomb hatására még meg lehet gondolni, hogy valóban bekerüljön-e a műsor a műsor-összeállításba. Ha késznek gondoljuk az összeállítást, akkor a véglegesít gomb megnyomásával véglegesíthetjük döntésünket, feltéve, hogy megadtuk a műsorba kerülés helyes dátumát (formailag is helyes, de arra is figyeljen, hogy a mai dátumnál későbbi dátumot adjunk).

A véglegesítés a következőt jelenti: Hatására a listafelületen szereplő műsorokból létrejön egy adás. Az adást a dátum és benne lévő műsorlista jellemzi, melyet az adasbaVesz() és a torol() metódussal lehet módosítani, mindkét metódusban a hozzáadandó, illetve törlendő műsor szerepel paraméterként.

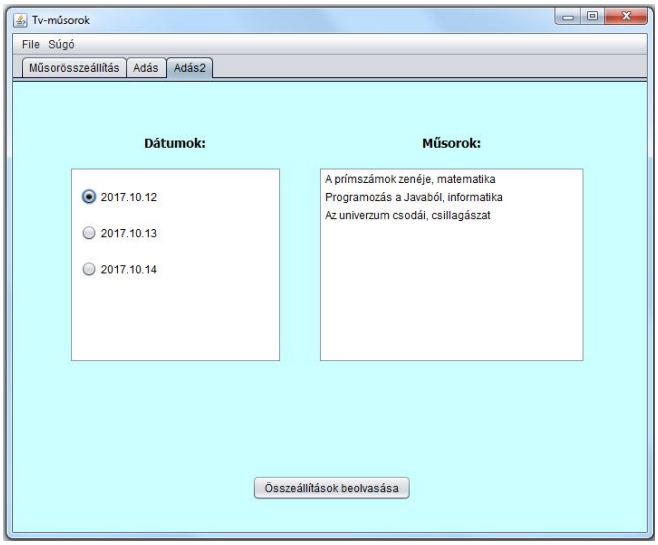
A létrejött adás bekerül az adások listájába, a listafelület és a dátummező pedig törlődik, és jöhet a következő válogatás. Ha már nem akarunk több adást létrehozni, akkor a menüpontban elmenthetjük a teljes összeállítást (a felhasználó által választott fájlba). Objektumként mentsen!

Két alternatív adásfelület van:



Az Összeállítás beolvasása feliratú gomb hatására beolvassuk a korábban elmentett műsorfájlt, és már meg is jelenik a legelső adás műsorlistája, illetve az adás dátuma. A következő gomb hatására megnézhetjük a következő adás műsorait. Esetleg úgy is megoldhatja, hogy az utolsó után ismét az elsőt lehessen látni.

A műsorfelületre kattintva nézik az egyes műsorokat, vagyis ekkor növekszik a kiválasztott műsor nézettsége. Az is jelenjen meg, hogy az eddigi nézettségek alapján a teljes adásnapnak milyen hatása van a társadalomra, mekkora az összhatás, és ez alapján aznap okosodott vagy butult a társadalom.



A másik felületen megjelenik az összes adásnap, és megnyomva a megfelelő rádiógombot, megjelenik az adott dátumhoz tartozó műsorlista.

Természetesen ezt is kiegészítheti a nézettségre vonatkozó eseményekkel, információkkal.

Megoldásvázlat

Mivel ebben a feladatban sok ismétlő jellegű részlet van, ezért sok részletet különösebb magyarázat nélkül, vagy esetleg sehogy sem illesztek ide.

Az alaposztályokban nem sok különlegesség van, azonban van néhány, amelyről szót kell ejteni. Ezek a későbbi feladatok miatt kerültek bele:

1. Névsor szerint rendezve akarjuk megjelentetni a műsorcímeket, ezért meg kell oldanunk a rendezést. Ezt vagy egy külső rendező osztály segítségével tehetjük meg, vagy úgy, hogy a rendezendő objektumok osztályában adjuk meg a hasonlítási szempontot. Jelenleg ez utóbbi megoldást választottuk, de lényegében mindegy, hogy melyik módon oldjuk meg. Ahhoz, hogy a példányok összehasonlíthatóak legyenek, a `Musor` osztályt `Comparable` típusúra kell deklarálnunk (vagyis implementálnia kell a `Comparable` interfészt), és meg kell valósítanunk benne az interfész által előírt `compareto()` metódust.

2. Bár a konkrét feladat csak meglévő műsor példányokkal manipulál, de elvileg úgy is bővíthetnénk a programot, hogy kívülről (a műsor-összeállításért felelős panelen) újabb cím-hossz párok megadásával újabb műsorpéldányokat hozzunk létre. Ha ilyen esetben szeretnénk kivédeni azt, hogy ugyanaz (illetve ugyanolyan című és hosszúságú) műsor ne kerülhessen be a műsorlistába (azaz „működjön” a lista `contains()` metódusa, az kell, hogy az osztályban generáljuk az `equals()` és `hashcode()` metódusokat. (Ezeket nem másolom ide, generálhatóak.)

3. Bár nem igazán logikus az, hogy adatbázisból olvasunk és fájlba írunk, de jó lenne, ha szó esne az objektumként való fájlba mentésre, ill. onnan való olvasásra. Bár kicsit kapkodva, de erre órán is sort kerítettünk. Maga a fájlba írás, és az onnan való olvasás nagyon egyszerű, hiszen egyetlen utasítással egy egész objektum-listát (és még bonyolultabb szerkezeteket is) ki lehet írni, és be lehet olvasni, de van egy feltétele, mégpedig az, hogy a mentésben szereplő összes objektum osztálya szerializálható legyen. Az `ArrayList` és a többi használt típus (pl. `String`) ilyen, de az általunk írt osztályok nem. Ezért ahhoz, hogy objektumként tudjuk menteni, a `Musor` osztályt is szerializálhatóvá kell tenni, azaz implementálnia kell a `Serializable` interfészt is. (Az utódok ezt is öröklik.)

Ezek után a beolvasáshoz, műsor-összeállításhoz szükséges alaposztályok (a generált metódusok, azaz setter/getter, `equals()`, `hashcode()` nélkül):

```
public class Musor implements Comparable<Musor>, Serializable{
    private String cim;
    private int hossz;
    private int nezoSzam;

    public Musor(String cim, int hossz) {
        this.cim = cim;
        this.hossz = hossz;
    }

    public void nezik(){
        nezoSzam++;
    }

    public int hatas(){
        return nezoSzam*hossz;
    }

    @Override
    public String toString() {
        return cim;
    }

    @Override
    public int compareTo(Musor o) {
        return this.cim.compareTo(o.cim);
    }
```

```
public class IsmeretTerjesztoMusor extends Musor{
    private String tudomanyAg;
    private static int szorzo;

    public IsmeretTerjesztoMusor(String cim, int hossz, String tudomanyAg) {
        super(cim, hossz);
        this.tudomanyAg = tudomanyAg;
    }

    @Override
    public int hatas() {
        return super.hatas()*szorzo;
    }

    @Override
    public String toString() {
        return super.toString()+ ", " + tudomanyAg;
    }
```

```java
public class GagyiMusor extends Musor{

    private Celeb celeb;
    private static int szorzo;

    public GagyiMusor(String cim, int hossz, Celeb celeb) {
        super(cim, hossz);
        this.celeb = celeb;
    }

    public GagyiMusor(String cim, int hossz) {
        super(cim, hossz);
    }

    @Override
    public int hatas() {
        return (celeb != null) ? -super.hatas()*szorzo/celeb.getIq() : 0;
    }
```

```java
public class Celeb implements Serializable{

    private String nev;
    private int iq;

    public Celeb(String nev, int iq) {
        this.nev = nev;
        this.iq = iq;
    }

    @Override
    public String toString() {
        return nev;
    }
```

Az adatbevitelhez hasonlót is vettünk korábban. Ha csak a konkrét feladatot nézzük, akkor a beolvasást megírhatnánk a panelen is, de ha számítunk egy esetleges későbbi módosítási lehetőségre, akkor célszerű egy interfészben megfogalmazni az elvárásunkat, és ezt implementálni (esetleg több különböző) beolvasási osztállyal :

```java
public interface AdatInput {
    public List<Musor> musorLista() throws Exception;
    public List<Celeb> celebLista() throws Exception;
}
```

```java
public class AdatBazisInput implements AdatInput{

    private Connection kapcsolat;

    public AdatBazisInput(Connection kapcsolat) {
        this.kapcsolat = kapcsolat;
    }


    @Override
    public List<Musor> musorLista() throws Exception {
        List<Musor> musorok = new ArrayList<>();
        String sqlUtsitas = "SELECT * from MUSOR";
        final String GAGYI = "gagyi";
        String cim, jellemzo;
        int hossz;
        Musor musor;

        try (Connection kapcsolat = this.kapcsolodas();
                Statement utasitasObjektum = kapcsolat.createStatement();
                ResultSet eredmenyHalmaz =
                                    utasitasObjektum.executeQuery(sqlUtsitas)) {
            while (eredmenyHalmaz.next()) {
                cim = eredmenyHalmaz.getString("cim");
                hossz = eredmenyHalmaz.getInt("perchossz");
                jellemzo = eredmenyHalmaz.getString("jellemzo");
                if(jellemzo.equals(GAGYI)){
                    musor = new GagyiMusor(cim, hossz);
                }else{
                    musor = new IsmeretTerjesztoMusor(cim, hossz, jellemzo);
                }
                musorok.add(musor);
            }
        }
        return musorok;
    }

        @Override
        public List<Celeb> celebLista() throws Exception {
            List<Celeb> celebek = new ArrayList<>();

            String sqlUtsitas = "SELECT * from CELEB";
            final String GAGYI = "gagyi";
            String nev;
            int iq;

            try (Connection kapcsolat = this.kapcsolodas();
                    Statement utasitasObjektum = kapcsolat.createStatement();
                    ResultSet eredmenyHalmaz =
                                        utasitasObjektum.executeQuery(sqlUtsitas)) {
                while (eredmenyHalmaz.next()) {
                    nev = eredmenyHalmaz.getString("nev");
                    iq = eredmenyHalmaz.getInt("iq");
                    celebek.add(new Celeb(nev, iq));
                }
            }
            return celebek;
        }
    }
```

A frame-n csak a menüpontok működése szerepel – és persze, a `main()` metódusban az egész program indítása. A menüpontok hatására a panel két metódusát hívjuk meg:

```
private void adatBeviteljMenuItemActionPerformed(java.awt.event.ActionEvent
    musorPanel1.adatBevitel();
}

private void mentesMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    musorPanel1.mentes();
}
```

A panel metódusai (a mentésről majd később ejtek néhány szót):

Elöljáróban még annyit, hogy miért is olvastunk listába? Abban a kérdésben, hogy a beolvasott adatokat azonnal modellbe rakjuk vagy listába, az lehet támpont, hogy mi történik a beolvasáskor létrehozott objektumokkal: azonnal megjelennek-e egy listafelületen, vagy sem. Esetünkben a celebek egyáltalán nem jelennek meg listafelületen, a beolvasáskor létrehozott műsorok is csak rendezés és szelektálás után, vagyis a celebeket nem szabad modellbe raknunk, a műsorokat pedig nem érdemes, mert legföljebb szelektálva két modellbe kellene tennünk őket a beolvasás után, ez azonban elbonyolítaná a megoldást.

A beolvasás, celeb-hozzárendelés, szelektálás után még a „baloldali” eseményeket is iderakom (vagyis a műsor összerakásához szükséges eseményeket):

```
public class MusorPanel extends javax.swing.JPanel {

    private DefaultListModel<Musor> ismeretterjesztoModell =
                                                new DefaultListModel<>();
    private DefaultListModel<Musor> gagyiModell = new DefaultListModel<>();
    private DefaultListModel<Musor> musorModell = new DefaultListModel<>();
    private List<Adas> adasLista = new ArrayList<>();

    public MusorPanel() {
        initComponents();
        lstGagyi.setModel(gagyiModell);
        lstIsmeretterjeszto.setModel(ismeretterjesztoModell);
        lstMusorok.setModel(musorModell);
    }

    void adatBevitel() {
        try {
            AdatInput adatInput = AdatBazisInput.getPeldany();
            List<Musor> musorok = adatInput.musorLista();
            List<Celeb> celebek = adatInput.celebLista();
            // A gagyi műsorokhoz véletlenszerűen hozzárendelünk
            // egy-egy celebet.
            celebValasztas(musorok, celebek);
            // A műsorokat szortírozzuk fajtájuk szerint.
            musorokListaba(musorok);
        } catch (Exception ex) {
            Logger.getLogger(MusorPanel.class.getName()).log(Level.SEVERE, null, 
        }
    }
```

```
private void celebValasztas(List<Musor> musorok, List<Celeb> celebek) {
    int celebIndex;
    for (Musor musor : musorok) {
        if(musor instanceof GagyiMusor){
            celebIndex = (int) (Math.random()*celebek.size());
            ((GagyiMusor) musor).setCeleb(celebek.get(celebIndex));
        }
    }
}

private void musorokListaba(List<Musor> musorok) {
    Collections.sort(musorok);
    for (Musor musor : musorok) {
        if(musor instanceof IsmeretTerjesztoMusor){
            ismeretterjesztoModell.addElement(musor);
        }else {
            gagyiModell.addElement(musor);
        }
    }
}
```

```
private void btnKivalasztActionPerformed(java.awt.event.ActionEvent evt) {
    List<Musor> musorok = lstIsmeretterjeszto.getSelectedValuesList();
    for (Musor musor : musorok) {
        if(!musorModell.contains(musor)){
            musorModell.addElement(musor);
        }
    }
    lstIsmeretterjeszto.clearSelection();

    Musor musor = (Musor) lstGagyi.getSelectedValue();
     if(musor != null && !musorModell.contains(musor)){
         musorModell.addElement(musor);
     }
     lstGagyi.clearSelection();
}
```

```
private void lstGagyiValueChanged(javax.swing.event.ListSelectionEvent evt) {
    Musor musor = (Musor) lstGagyi.getSelectedValue();
    if(musor != null)lblCeleb.setText("Celeb vendég: " +
                                      ((GagyiMusor)musor).getCeleb());
}

private void btnTorlesActionPerformed(java.awt.event.ActionEvent evt) {
    List<Musor> musorok = lstMusorok.getSelectedValuesList();
    for (Musor musor : musorok) {
        musorModell.removeElement(musor);
    }
}
```

A véglegesítéshez szükség lesz egy újabb osztályra. Ez az adás aktuális dátumát és az aznapra kiválasztott műsorokat tartalmazza. Az osztályban lehetőséget adunk rá, hogy az adáshoz új műsort lehessen rendelni, illetve arra is, hogy egy meglévőt törölni lehessen:

Az osztály getterek nélkül:

```
public class Adas implements Serializable{
    private Date datum;
    private List<Musor> musorLista = new ArrayList<>();

    public Adas(Date datum) {
        this.datum = datum;
    }

    public void adasbaVesz(Musor musor){
        if(!musorLista.contains(musor)){
            musorLista.add(musor);
        }
    }

    public void torol(Musor musor){
        musorLista.remove(musor);
    }

    // Megadja az aktuális dátum string alakját
    public String stringAlakuDatum(){
        DateFormat df = new SimpleDateFormat("yyyy.MM.dd");
        return df.format(datum) ;
    }
```

Véglegesítéskor `Adas` típusú példányt kell létrehoznunk, feltölteni a kiválasztott műsorokkal, és a létrejött adást hozzáadni az adások listájához.

```
private void btnVeglegesitoActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String temp = txtDatum.getText();
        Date datum = new SimpleDateFormat("yyyy.MM.dd").parse(temp);

        // Ha a megadott dátum korábbi a mai dátumnál, akkor hibaüzenet.
        Date ma = new Date();
        if(datum.before(ma)) throw new Exception();

        Adas adas = new Adas(datum);
        for (int i = 0; i < musorModell.size(); i++) {
                adas.adasbaVesz(musorModell.get(i));
        }
        adasLista.add(adas);
        musorModell.clear();
        txtDatum.setText("");
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(btnTorles, "Hibás dátum");
    }
}
```

Megjegyzés: az is lehet, hogy a műsorlistát az `Adas` osztály konstrukorában adjuk meg, akkor még egyszerűbben jön létre a példány.

Mentéskor objektumként akarjuk elmenteni az adáslistát – emiatt szerializáltuk korábban a `Musor` osztályt, és ugyanezért kellett szerializálni az `Adas` osztályt is.

```
void mentes() {
    JFileChooser fajlValaszto = new JFileChooser(new File("."));
    if (fajlValaszto.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {
        try {
            File fajl = fajlValaszto.getSelectedFile();
            if (fajl.exists()) {
                if (JOptionPane.showConfirmDialog(this,
                        " a fajl már létezik, felülírjam?", "Hibaüzenet",
                        JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION) {
                    binfajlbaIr(fajl);
                }
            } else {
                binfajlbaIr(fajl);
            }
        } catch (IOException ex) {
            Logger.getLogger(MusorPanel.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

private void binfajlbaIr(File fajl) throws FileNotFoundException, IOException {
    FileOutputStream fout = new FileOutputStream(fajl);
    ObjectOutputStream oout =
                    new ObjectOutputStream(new BufferedOutputStream(fout));
    oout.writeObject(adasLista);
    oout.flush();
}
```

Röviden megmutatom a másik panelt is. Ahhoz, hogy ezt is láthassuk, a mostani panelt le kell venni a frame-ről, rátenni a frame-re egy `JTabbedPane` példányt, arra visszarakni a műsorpanelt is és az adáspanelt is.

A panel kódját magyarázatok nélkül közlöm.

```
public class AdasPanel extends javax.swing.JPanel {

    private List<Adas> adasLista = new ArrayList<>();
    private DefaultListModel<Musor> musorModell = new DefaultListModel<>();
    private int aktualis = 0;

    public AdasPanel() {
        initComponents();
        lstMusorok.setModel(musorModell);
    }
```

```
private void btnBeolvasasActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fajlValaszto = new JFileChooser(new File("."));
    if (fajlValaszto.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
        try {
            File fajl = fajlValaszto.getSelectedFile();
            binfajlbol(fajl);
            kiir(adasLista.get(aktualis));
        } catch (IOException | ClassNotFoundException ex) {
            Logger.getLogger(AdasPanel.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

private void binfajlbol(File fajl) throws FileNotFoundException, IOException,
                                          ClassNotFoundException {
    FileInputStream fajlIn = new FileInputStream(fajl);
    ObjectInputStream oin = new ObjectInputStream(new BufferedInputStream(fajlIn));
    adasLista = (List<Adas>) oin.readObject();
}

private void kiir(Adas adas) {
    musorModell.clear();
    for (Musor musor : adas.getMusorLista()) {
        musorModell.addElement(musor);
    }
    lblDatum.setText("Dátum: " + adas.stringAlakuDatum());
}

private void btnKovetkezoActionPerformed(java.awt.event.ActionEvent evt) {
    aktualis++;
    if(aktualis >= adasLista.size()){
       aktualis = 0;
    }
    kiir(adasLista.get(aktualis));
}
```

A másik adáspanelről azt mutatom meg, hogy hogyan lehet a beolvasott fájl alapján felrakni és működőképessé tenni a rádiógombokat:

```
    private void binfajlbol(File fajl) throws FileNotFoundException, IOException,
                                              ClassNotFoundException {
        FileInputStream fajlIn = new FileInputStream(fajl);
        ObjectInputStream oin
                = new ObjectInputStream(new BufferedInputStream(fajlIn));
        adasLista = (List<Adas>) oin.readObject();
        kirak(adasLista);
    }
```

```
private void kirak(List<Adas> adasLista) {
    JRadioButton rdButton;
    int x = 10, y = 20;
    int tav = 40;
    int gombX = 300, gombY = 20;

    for (Integer i = 0; i < adasLista.size(); i++) {
        rdButton = new JRadioButton(adasLista.get(i).stringAlakuDatum());
        rdButton.setLocation(x, y + i * tav);
        rdButton.setSize(gombX, gombY);
        rdButton.setName(i.toString());
        rdButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ae) {
                int index
                        = Integer.parseInt(((JRadioButton)ae.getSource()).
                                                        getName());
                kiir(adasLista.get(index));
            }
        });
        this.hatterPanel.add(rdButton);
        buttonGroup1.add(rdButton);
    }
}

private void kiir(Adas adas) {
    musorModell.clear();
    for (Musor musor : adas.getMusorLista()) {
        musorModell.addElement(musor);
    }
}
```

Végül egy rövid teszt:

```java
public class Teszt {

    public Teszt() {
    }

    private static final int SZORZO1 = 2;
    private static final int SZORZO2 = 3;

    private static final String CIM1 = "cim1";
    private static final String CIM2 = "cim2";
    private static final String TUDOMANYAG = "tudomany";
    private static final String CELEB_NEV = "Celebnév";

    private static final int HOSSZ = 10;
    private static final int CELEB_IQ = 100;
    // mai dátum
    private static final Date DATUM = new Date();

    private IsmeretTerjesztoMusor musor1;
    private GagyiMusor musor2;

    @Before
    public void setUp() {
        Celeb celeb = new Celeb(CELEB_NEV, CELEB_IQ);
        musor1 = new IsmeretTerjesztoMusor(CIM1, HOSSZ, TUDOMANYAG);
        musor2 = new GagyiMusor(CIM2, HOSSZ, celeb);
        IsmeretTerjesztoMusor.setSzorzo(SZORZO1);
        GagyiMusor.setSzorzo(SZORZO2);
    }

    @Test
    public void alapTeszt() {
        assertTrue(musor1.getNezoSzam() == 0);
        assertTrue(musor2.getNezoSzam() == 0);

        musor1.nezik();
        assertTrue(musor1.getNezoSzam() == 1);

        // A köv. 3 feltétel ugyanazt mondja, mindegyik elfogadható,
        // talán az utolsó a legkevésbé
        assertTrue(musor1.hatas()
                == musor1.getHossz() * musor1.getNezoSzam() *
                                        IsmeretTerjesztoMusor.getSzorzo());
        assertTrue(musor1.hatas() == HOSSZ * 1 * SZORZO1);
        assertTrue(musor1.hatas() == 20);

        musor1.nezik();
        assertTrue(musor1.getNezoSzam() == 2);
        assertTrue(musor1.hatas() == HOSSZ * 2 * SZORZO1);

        musor2.nezik();
        assertTrue(musor2.getNezoSzam() == 1);
        assertTrue(musor2.hatas() == HOSSZ * 1 *
                                        SZORZO2 / musor2.getCeleb().getIq());
```

```
        musor2 = new GagyiMusor(CIM2, HOSSZ);
        musor2.nezik();
        assertTrue(musor2.hatas() == 0);

        Adas adas = new Adas(DATUM);
        assertTrue(adas.getMusorLista().isEmpty());
        adas.adasbaVesz(musor1);
        assertTrue(adas.getMusorLista().size() == 1);
        adas.torol(musor1);
        assertTrue(adas.getMusorLista().isEmpty());
    }
}
```