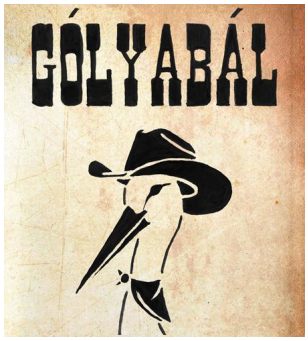
Gólyabál



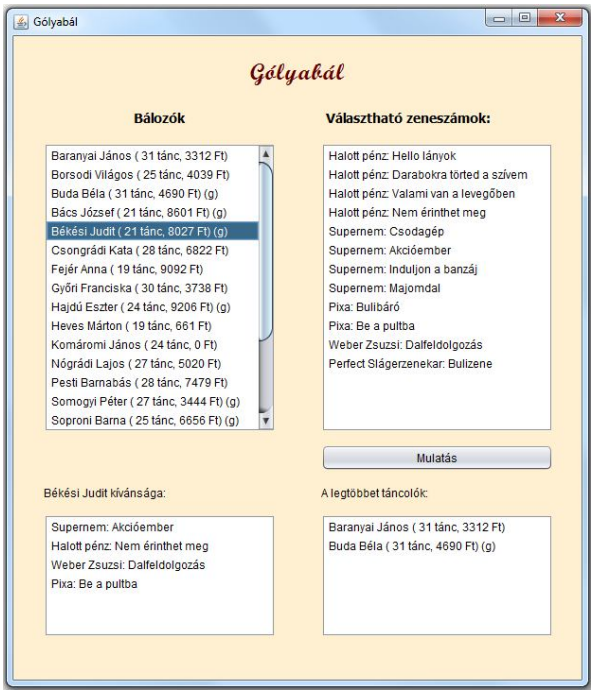
Szimuláljon egy gólyabált.

Minden **bálozó**nak *egyedi sorszám*a van, és természetesen *nev*e. Mindegyikük *fogyaszt()*, ekkor *költségei* a metódus paraméterében lévő értékkel növekszenek, persze, csak addig, ameddig a *zsebpénz*éből futja. Mindegyikük *táncol()* is időnként, ekkor *táncainak száma* eggyel növekszik.

A **gólyá**k kedvezményt kapnak. Bármit is fogyasztanak, kedvezményes áron kapják, mégpedig az összes gólyára egyaránt érvényes *kedvezményszázalék*kal olcsóbban. Még egy privilégiumuk van: választhatnak zeneszámot – természetesen ugyanazt többször is. Amikor egy gólya *kiválaszt()* egy zeneszámot, akkor az bekerül a *kívánt zeneszámai listájá*ba.

Egy **zeneszám** az *előadó*jával és *cím*ével adható meg.

Szimulálja a bált egy 600*650 pixel belső méretű felületen:



A program indulásakor azonnal megjelenik a bálozók névsora (lehetőleg ábécé sorrendben), illetve a választható zeneszámok listája. A bálozók adatainak beolvasásakor mindenki kapjon 0 és egy egységes határ közötti véletlen zsebpénz-mennyiséget.

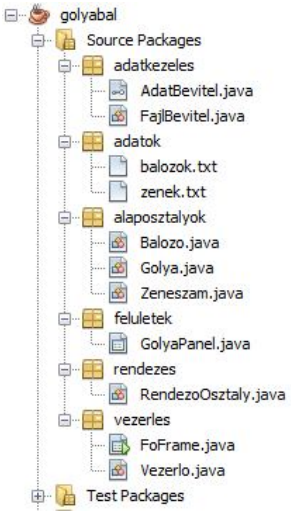
A bálozók listájára kattintva, ha gólya a kiválasztott illető, akkor „válasszon” egy véletlen zenét az adott kínálatból (nem a másik listára való kattintással, hanem véletlen generálással, bár szorgalmiként megoldhatja, hogy kattintással választhasson, de csak a gólya). Az alsó lista-felületen az általa választott zeneszámok listája legyen látható (az eddigi összes választott), a lista fölött pedig egy felirat jelezze, hogy kinek a választását látjuk. Ha nem gólya az illető, akkor az alsó listafelület maradjon üresen, a kiírt szöveg pedig tartalmazza azt, hogy a választott ember nem gólya.

A „Mulatás” feliratú gombot megnyomva fusson le egy ciklus, amelyben egy véletlenül választott ember táncol (nem muszáj párban), egy másik véletlen ember pedig véletlen mennyiségű pénzt költ. A jobboldali alsó listában pedig – minden gombnyomáskor frissülve – jelenjen meg a legtöbbet táncolók listája (első alkalommal a lista feletti felirat is). Az induló felületen csak a nevek legyenek olvashatóak a bálozók listájában (gólyák esetén mellette a „g” betű), a mulatás után viszont lássuk a táncok számát és az elköltött Ft mennyiséget is.

Néhány megoldásrészlet:

A felületkialakítás részleteivel és a fejlesztőkörnyezet által generált részletekkel nem foglalkozunk, csak a többiről ejtünk néhány szót.

A projekt szerkezete:



Az alaposztályok kódját most is getterek, setterek nélkül közöljük.

```
public class Balozo  {

    private String nev;
    private int azonosito;
    private int tancokSzama;
    private int koltsegek;
    private int zsebpenz;

    private static int utolsoIndex;
    private static boolean voltMulatas;

    public Balozo(String nev) {
        this.nev = nev;
        this.azonosito = ++utolsoIndex;
    }

    /** Fogyaszt, ha van elég pénze, ekkor költségei növekszenek ...4 lines */
    public void fogyaszt(int fogyasztás) {
        if (this.zsebpenz > this.koltsegek + fogyasztás) {
            this.koltsegek += fogyasztás;
        }
    }

    /** Táncoláskor a táncok száma növekszik ...3 lines */
    public void tancol() {
        this.tancokSzama++;
    }

    /** Feltételtől függően más-más a kiíratási kép ...4 lines */
    @Override
    public String toString() {
        if(!voltMulatas) return nev;
        return String.format("%s (%d tánc, %d Ft)", this.getNev(),
                    this.getTancokSzama(),this.getKoltsegek());
    }
```

```java
public class Golya extends Balozo {

    private static double kedvezmeny;
    private List<Zeneszam> kivantZeneLista = new ArrayList<>();

    public Golya(String nev) {
        super(nev);
    }

    /** Kedvezményesen fogyaszt ...5 lines */
    @Override
    public void fogyaszt(int fogyasztás) {
        int akcio = fogyasztás - (int) (fogyasztás * Golya.kedvezmeny);
        super.fogyaszt(akcio);
    }

    public void kivalaszt(Zeneszam zeneszam) {
        kivantZeneLista.add(zeneszam);
    }

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

```java
public class Zeneszam {

    private String eloado;
    private String cim;

    public Zeneszam(String eloado, String cim) {
        this.eloado = eloado;
        this.cim = cim;
    }

    @Override
    public String toString() {
        return this.eloado + " - " + this.cim;
    }
```

Az adatbevitel:

```java
public interface AdatBevitel {

    public List<Balozo> balozoBevitelLista() throws Exception;
    public List<Zeneszam> zeneszamBevitelLista() throws Exception;
}
```

```java
public class FajlBevitel implements AdatBevitel {

    private String balozoUtvonal;
    private String zenekUtvonal;
    private final String KODOLAS = "UTF-8";

    public FajlBevitel(String balozoUtvonal, String zenekUtvonal) {
        this.balozoUtvonal = balozoUtvonal;
        this.zenekUtvonal = zenekUtvonal;
    }

    @Override
    public List<Balozo> balozoBevitelLista() throws Exception {
        List<Balozo> balozok = new ArrayList<>();
        try (InputStream ins = this.getClass().getResourceAsStream(balozoUtvonal);
                Scanner sc = new Scanner(ins, KODOLAS)) {
            String sor;
            String adatok[];
            while (sc.hasNextLine()) {
                sor = sc.nextLine();
                if (!sor.isEmpty()) {
                    adatok = sor.split(";");
                    if ("1".equals(adatok[1])) {
                        balozok.add(new Golya(adatok[0]));
                    } else {
                        balozok.add(new Balozo(adatok[0]));
                    }
                }
            }
        }
        return balozok;
    }

    @Override
    public List<Zeneszam> zeneszamBevitelLista() throws Exception {
        List<Zeneszam> zeneszamok = new ArrayList<>();
        try (InputStream ins = this.getClass().getResourceAsStream(zenekUtvonal);
                Scanner sc = new Scanner(ins, KODOLAS)) {
            String sor;
            String adatok[];
            while (sc.hasNextLine()) {
                sor = sc.nextLine();
                adatok = sor.split(";");
                zeneszamok.add(new Zeneszam(adatok[0], adatok[1]));
            }
        }
        return zeneszamok;
    }
}
```

Mivel most több szempont szerint is kell majd rendezni, érdemes egy rendező osztályt írni. Az osztály getterek nélküli része:

```
public class RendezoOsztaly implements Comparator<Balozo> {

    public enum Szempont{ABC,KOLTSEG}

    public static final boolean NOVEKVOEN=true;
    public static final boolean CSOKKENOEN=false;

    private static Szempont valasztottSzempont;
    private static boolean miModon;

    @Override
    public int compare(Balozo o1, Balozo o2) {
        switch(valasztottSzempont){
            case ABC:
                return miModon? o1.getNev().compareTo(o2.getNev()) :
                        o2.getNev().compareTo(o1.getNev());
            case KOLTSEG:
                return miModon? o1.getKoltsegek() - o2.getKoltsegek() :
                         o2.getKoltsegek() - o1.getKoltsegek();
            default: return 0;
        }
    }

    public static void setValasztottSzempont(Szempont valasztottSzempont,
                            boolean miModon) {
        RendezoOsztaly.valasztottSzempont = valasztottSzempont;
        RendezoOsztaly.miModon = miModon;
    }
```

A frame (generált részek nélkül):

```
public class FoFrame extends javax.swing.JFrame {

    private final int SZELESSEG = 616;
    private final int MAGASSAG = 688;
    private final String CIM = "Gólyabál";

    public FoFrame() {
        initComponents();
        beallitas();
    }
```

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
     Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new FoFrame().start();
        }
    });
}

private void beallitas() {
    this.setSize(SZELESSEG, MAGASSAG);
    this.setTitle(CIM);
    this.setLocationRelativeTo(null);
}

private void start() {
    this.setVisible(true);
    Vezerlo vezerlo = new Vezerlo(golyaPanel1);
    golyaPanel1.setVezerlo(vezerlo);
    vezerlo.start();
}
```

A vezérlő osztály a `start()` metódusig:

```
public class Vezerlo {

    private final int FOGYASZTAS_HATAR = 1000;
    private final int ZSEBPENZ_HATAR = 10000;
    private final int MULATASI_CIKLUS_MERETE = 100;
    private final double KEDVEZMENY = 0.40;

    private final String BALOZOK_UTVONAl = "/adatok/balozok.txt";
    private final String ZENEK_UTVONAL = "/adatok/zenek.txt";

    private List<Balozo> balozok;
    private List<Zeneszam> zenek;

    private GolyaPanel golyaPanel;

    public Vezerlo(GolyaPanel balpanel) {
        this.golyaPanel = balpanel;
    }

    public void start() {
        statikusAdatok();
        adatBevitel();
        zsebpenzKiosztas();
    }
```

```
    private void statikusAdatok() {
        Golya.setKedvezmeny(KEDVEZMENY);
    }

    private void adatBevitel() {
        AdatBevitel adatBevitel = new FajlBevitel(BALOZOK_UTVONAl, ZENEK_UTVONAL);
        try {
            balozok = adatBevitel.balozoBevitelLista();
            abcSorrend();
            zenek = adatBevitel.zeneszamBevitelLista();
            golyaPanel.adatMegjelenites(balozok, zenek);
        } catch (Exception ex) {
            Logger.getLogger(Vezerlo.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public void abcSorrend() {
        RendezoOsztaly.setValasztottSzempont(RendezoOsztaly.Szempont.ABC,
                RendezoOsztaly.NOVEKVOEN);
        Collections.sort(balozok, new RendezoOsztaly());
    }

    private void zsebpenzKiosztas() {
        for (Balozo balozo : balozok) {
            balozo.setZsebpenz((int) (Math.random() * ZSEBPENZ_HATAR));
        }
    }
```

A panel hivatkozott metódusa (a szükséges inicializálásokkal együtt):

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

    private DefaultListModel<Balozo> balozoModell = new DefaultListModel<>();
    private DefaultListModel<Zeneszam> zeneszamModell
                                                = new DefaultListModel<>();
    private DefaultListModel<Zeneszam> valasztottZeneModell
                                                = new DefaultListModel<>();
    private DefaultListModel<Balozo> legtobbetTancolokModell
                                                = new DefaultListModel<>();

    private Vezerlo vezerlo;

    public void setVezerlo(Vezerlo vezerlo) {
        this.vezerlo = vezerlo;
    }
```

```
public GolyaPanel() {
    initComponents();
    lstBalozok.setModel(balozoModell);
    lstZenek.setModel(zeneszamModell);
    lstValasztottZenek.setModel(valasztottZeneModell);
    lstLegtobbetTancolok.setModel(legtobbetTancolokModell);
}

public void adatMegjelenites(List<Balozo> balozok, List<Zeneszam> zenek) {
    balozoModell.clear();
    zeneszamModell.clear();
    for (Balozo balozo : balozok) {
        balozoModell.addElement(balozo);
    }

    for (Zeneszam zeneszam : zenek) {
        zeneszamModell.addElement(zeneszam);
    }
}
```

A „Mulatás” gomb hatása:

a panelen:

```
private void btnMulatasActionPerformed(java.awt.event.ActionEvent evt) {
    vezerlo.mulatas();
}
```

a `Vezerlo` osztályban:

```
public void mulatas() {
    for (int i = 1; i <= MULATASI_CIKLUS_MERETE; i++) {
        balozok.get((int) (Math.random() * balozok.size())).tancol();
        balozok.get((int) (Math.random() * balozok.size())).
                fogyaszt((int) (Math.random() * FOGYASZTAS_HATAR));
    }
    csokkenoKoltsegSorrend();
    golyaPanel.balozokMegjelenitese(balozok);
    legtobbetTancolok();
}

public void csokkenoKoltsegSorrend() {
    RendezoOsztaly.setValasztottSzempont(RendezoOsztaly.Szempont.KOLTSEG,
            RendezoOsztaly.CSOKKENOEN);
    Collections.sort(balozok, new RendezoOsztaly());
}
```

```
public void legtobbetTancolok() {
    List<Balozo> legtobbetTancolokListaja = new ArrayList<>();
    int max = balozok.get(0).getTancokSzama();
    for (Balozo balozo : balozok) {
        if (balozo.getTancokSzama() > max) {
            max = balozo.getTancokSzama();
        }
    }
    for (Balozo balozo : balozok) {
        if (balozo.getTancokSzama() == max) {
            legtobbetTancolokListaja.add(balozo);
        }
    }
    golyaPanel.legtobbetTancoltakMegjelenitese(legtobbetTancolokListaja);
}
```

A panel hivatkozott metódusai:

```
public void balozokMegjelenitese(List<Balozo> balozok) {
    Balozo.setVoltMulatas(true);
    balozoModell.clear();
    for (Balozo balozo : balozok) {
        balozoModell.addElement(balozo);
    }
}

public void legtobbetTancoltakMegjelenitese(List<Balozo> legtobbetTancolok) {
    legtobbetTancolokModell.clear();
    for (Balozo balozo : legtobbetTancolok) {
        legtobbetTancolokModell.addElement(balozo);
    }
}
```

A bálozók listájára kattintás:

a panelen:

```
private void lstBalozokMouseClicked(java.awt.event.MouseEvent evt) {
    valasztottZeneModell.clear();
    Balozo balozo = (Balozo) lstBalozok.getSelectedValue();
    if(balozo != null) vezerlo.zenetValaszt(balozo);
}
```

a `Vezerlo` osztályban:

```
public void zenetValaszt(Balozo balozo) {
    if (balozo instanceof Golya) {
        ((Golya) balozo).kivalaszt(
                        zenek.get((int) (Math.random() * zenek.size())));

        golyaPanel.balozoNevKiirasa(balozo.getNev() + " kívánsága: ");
        golyaPanel.kedvencZenekMegjelenitese(
                ((Golya) balozo).getKivantZeneLista());
    } else {
        golyaPanel.balozoNevKiirasa(balozo.getNev() + " nem gólya ");
    }
}
```

A panel hivatkozott metódusai:

```
public void balozoNevKiirasa(String szoveg) {
    lblDiak.setText(szoveg);
}

public void kedvencZenekMegjelenitese(List<Zeneszam> kivantZeneLista) {
    for (Zeneszam zeneszam : kivantZeneLista) {
        valasztottZeneModell.addElement(zeneszam);
    }
}
```