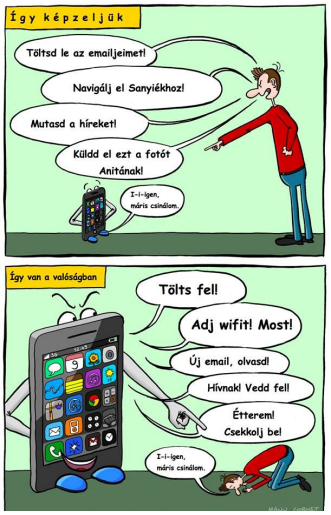


Mobilfónia

Több mint 25 éve vezették be Magyarországon a kereskedelmi mobilszolgáltatást. (Vagyis valószínűleg Ön nem is élt mobil nélküli világban.) Ennek kapcsán most azt „vizsgáljuk”, hogyan hatnak ezek a kütyük (mobil- és okos-telefon) az emberekre.

Egy **kütyü** a *típus*ával és egy *egyedi sorszám*mal azonosítható. Mindegyik tud *üzenetetküld()*eni, ekkor a kütyü által *küldött üzenetek mérete* megnövekszik a metódus paraméterében lévő karakter-számmal. Az üzenetek írása növeli az agy hüvelykujj irányítására szolgáló területét, a *hüvelykujjsejt()*ek száma (int) a küldött üzenetek méretétől függ, hasonlóan, de nem teljesen ugyanúgy számoljuk ki mobil- és okos-telefonok esetén.

Mobiltelefon esetén a hüvelykujjsejtek száma az üzenet mérete és egy, a mobiltelefonokra egyformán jellemző *billentyűerő* szorzata.

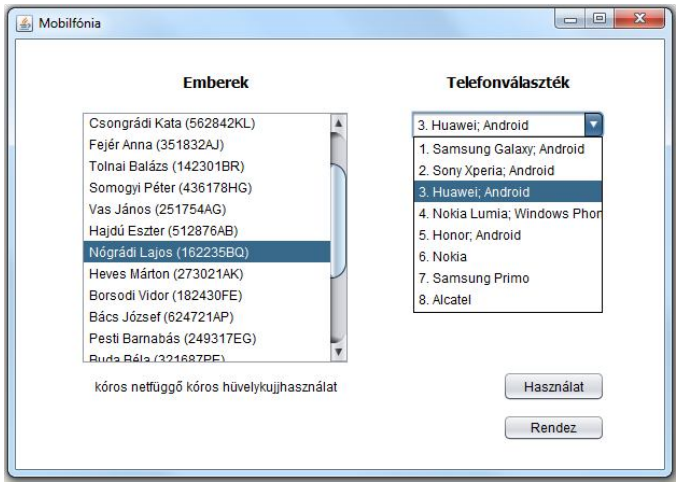
Okostelefon esetén ugyanígy számoljuk ki, csak a *billentyűerő* nagysága lesz más. Egy okos-telefon a típusa mellett az *operációs rendszere nevé*nek megadásával definiálható. Az eddigieken kívül esetenként még internetezni is lehet vele. Ez azon múlik, hogy *van WiFi*, vagy nincs. A *kapcsolodik()* metódus hatására lehet kapcsolódni, ekkor a van WiFi állítás igazzá válik, a *lekapcsolodik()* metódus hatására pedig hamissá. Amikor *internetezik()*, akkor, amennyiben van WiFi, a *netezéssel töltött idő* a metódus paraméterében lévő másodpercek (int) értékével növekszik.

Egy **ember** a *neve* és *személyigazolvány-száma* definiál. Amikor *kutyutvesz()*, akkor kütyüinek listájához hozzáadódik a metódus paraméterében lévő kütyü. (Persze, kétszer nem veheti meg ugyanazt.) Az emberek hüvelykujjhasználatát és net-függőségét vizsgálják, ezért a *huvelykujjero()* metódus eredményeként a kütyük hüvelykujjsejt()-jeinek összegét kapjuk, a *netidő()* eredménye pedig a netezéssel töltött idők összege.

Végül a *diagnozis()* szöveges formában visszaadja a diagnózist, vagyis ha a hüvelykujj-erő nagyobb, mint egy, az összes emberre egyformán érvényes *sejthatár*, akkor „kóros hüvelykujjhasználat”, ha a netezéssel töltött idő nagyobb, mint egy, szintén egyformán érvényes *függőségi határ*, akkor „kóros netfüggő”, egyébként „normális”.

Olvassa be az emberek és a kütyük adatait. A program indulásakor azonnal jelenjen meg az emberek listája, illetve egy comboBox-ban a telefonválaszték.

A comboBox-ra kattintva véletlen sokszor fusson le a következő: hozzunk létre egy ugyanolyan típusú (és esetleg operációs rendszerű) új kütyüt, amilyen a kiválasztott prototípus (ezt választjuk ki a comboBox-ból), és ezt vegye meg egy véletlenül kiválasztott ember. Természetesen többször is választhatunk telefont, és egy embernek több ugyanolyan telefonja is lehet (de nem ugyanaz).

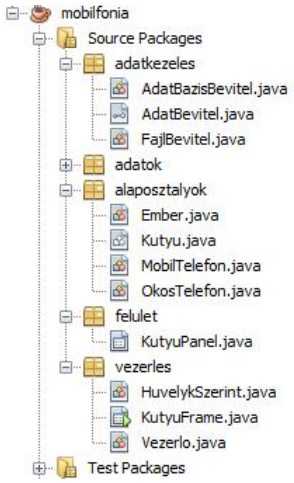


A Használat gomb hatása: valahányszor futtassuk le a következőt:

Válasszunk ki egy véletlen embert. Ha van kütyüje, akkor a kütyüi közül válasszunk ki véletlenszerűen egyet. Ez a kütyü küldjön egy véletlen hosszúságú üzenetet. Ha a kütyü történetesen okos-telefon, akkor valahány százalék eséllyel kapcsolódjon az internethez, egyébként kapcsolódjon le, majd véletlen hosszúságú ideig próbáljon internetezni.

A Rendez gomb hatása: hüvelykujj-erő szerint csökkenően rendezi az embereket. Az embereket tartalmazó listában most a név mellett a hüvelykerő jelenjen meg.

Néhány megoldásrészlet:



Az alaposztályok (getter/setter nélküli része):

```
public abstract class Kutyu {

    private String tipus;

    private static int utolsoSorszam;
    private int sorSzam;
    private int kuldottUzenetMeret;

    public Kutyu(String tipus) {
        this.tipus = tipus;
        utolsoSorszam++;
        sorSzam = utolsoSorszam;
    }

    @Override
    public String toString() {
        return sorSzam + ". " + tipus;
    }

    public abstract int huvelykUjjSejtSzam();

    public void uzenetetKuld(int karakterSzam){
        kuldottUzenetMeret += karakterSzam;
    }
```

```java
public class MobilTelefon extends Kutyu{

    private static double billentyuEro;

    public MobilTelefon(String tipus) {
        super(tipus);
    }

    @Override
    public int huvelykUjjSejtSzam(){
        return (int) (billentyuEro*this.getKuldottUzenetMeret());
    }



public class OkosTelefon extends Kutyu{

    private static double billentyuEro;
    private boolean vanWiFi;
    private int netezesiIdo;
    private String operaciosRendszerNeve;

    public OkosTelefon(String tipus, String operaciosRendszerNeve) {
        super(tipus);
        this.operaciosRendszerNeve = operaciosRendszerNeve;
    }

    @Override
    public int huvelykUjjSejtSzam(){
        return (int) (billentyuEro*this.getKuldottUzenetMeret());
    }

    public void kapcsolodik(){
        vanWiFi = true;
    }

    public void leKapcsolodik(){
        vanWiFi = false;
    }

    public void internetezik(int sec) {
        if (vanWiFi) {
            netezesiIdo += sec;
        }
    }

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

```java
public class Ember {

    private String nev;
    private String szigSzam;

    private List<Kutyu> kutyuk = new ArrayList<>();

    private static int huvelykSejtHatar;
    private static int netFuggosegHatar;
    private static boolean rendezve = false;

    public Ember(String nev, String szigSzam) {
        this.nev = nev;
        this.szigSzam = szigSzam;
    }

    public void kutyutVesz(Kutyu kutyu){
        if(!kutyuk.contains(kutyu)){
            kutyuk.add(kutyu);
        }
    }

    public int huvelyekUjjEro(){
        int ero = 0;
        for (Kutyu kutyu : kutyuk) {
            ero += kutyu.huvelykUjjSejtSzam();
        }
        return ero;
    }

    public int netIdo(){
        int ido=0;
        for (Kutyu kutyu : kutyuk) {
            if(kutyu instanceof OkosTelefon){
                ido += ((OkosTelefon)kutyu).getNetezesiIdo();
            }
        }
        return ido;
    }

    public String diagnozis(){
        String temp = "normális";
        String tempH = (huvelyekUjjEro()>huvelykSejtHatar)?
                                        "kóros hüvelykujjhasználat" : "";
        String tempF = (netIdo()> netFuggosegHatar)? "kóros netfüggő" : "";
        return ((tempH+tempF).isEmpty()) ? temp: tempF +" " + tempH;
    }

    @Override
    public String toString() {
        if(rendezve) return nev + ", hüvelykerő: " + huvelyekUjjEro();
        return nev + " (" + szigSzam + ')';
    }
```

Megjegyzés: Ha a diagnózis az, hogy megvan mindkét függősége, akkor a kiírásban a kettő közé vesszőt kellene írni. Ha igényes, akkor még ezt is oldja meg ☺.

Az adatbevitel, a vezérlő és a panel egymáshoz rendelése, illetve a rendező osztály a szokásos, nem részletezzük.

A combobox-hoz ugyanúgy, mint a többi Swing komponenshez, tartozik egy modell, ezt kell feltölteni és hozzárendelni a felülethez.

A panel megjelenítési metódusai:

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

    private Vezerlo vezerlo;

    private DefaultListModel<Ember> emberModell = new DefaultListModel<>() ;
    private DefaultComboBoxModel<Kutyu> kutyuBoxModell =
                                                new DefaultComboBoxModel<>();

    public KutyuPanel() {
        initComponents();
        lstEmberek.setModel(emberModell);
        cmbKutyuk.setModel(kutyuBoxModell);
    }

    public void megjelenitEmberek(List<Ember> emberLista) {
        emberModell.clear();
        for (Ember ember : emberLista) {
            emberModell.addElement(ember);
        }

        lblDiagnozis.setText("");
    }

    public void megjelenitKutyuk(List<Kutyu> kutyuLista) {
        kutyuBoxModell.removeAllElements();
        for (Kutyu kutyu : kutyuLista) {
            kutyuBoxModell.addElement(kutyu);
        }
    }
```

A diagnózis-label feliratának törlésére azért van szükség, mert rendezés után ismét meghívjuk ezt a metódust, és lehet, hogy ekkor van már rajta valamilyen szöveg. Hasonló okok miatt takarítjuk a modelleket is, ezzel azt érjük el, hogy a metódus minden újrahívásakor kizárólag a paraméterben szereplő lista elemei kerülnek a modellbe.

A combo-boksz-ra kattintás hatása:

A panelen:

```
    private void cmbKutyukActionPerformed(java.awt.event.ActionEvent evt) {
        Kutyu kutyu = (Kutyu) cmbKutyuk.getSelectedItem();
        if (kutyu != null) {
            vezerlo.kutyuVasarlas(kutyu);
        }
    }
```

A vezérlő osztályban:

```
    private List<Kutyu> kutyuLista = new ArrayList<>();
    private List<Ember> emberLista = new ArrayList<>();

    public void kutyuVasarlas(Kutyu kutyu) {
        int n = (int) (Math.random() * MAX_TELEFON_DB);
        int index;
        Kutyu ujKutyu;
        for (int i = 0; i < n; i++) {
            index = (int) (Math.random() * emberLista.size());
            if (kutyu instanceof OkosTelefon) {
                ujKutyu = new OkosTelefon(kutyu.getTipus(),
                        ((OkosTelefon) kutyu).getOperaciosRendszerNeve());
            } else {
                ujKutyu = new MobilTelefon(kutyu.getTipus());
            }
            emberLista.get(index).kutyutVesz(ujKutyu);
        }
    }
```

A Használat gomb:

A panelen:

```
    private void btnHasznalatActionPerformed(java.awt.event.ActionEvent evt) {
        vezerlo.hasznalat();
    }
```

A vezérlő osztályban:

```
public void hasznalat() {
    int emberIndex, kutyuIndex;
    int uzenet;
    Kutyu aktKutyu;
    for (int i = 0; i < HASZNALAT_SZAM; i++) {
        emberIndex = (int) (Math.random() * emberLista.size());
        if (emberLista.get(emberIndex).getKutyuk().size() > 0) {
            kutyuIndex = (int) (Math.random()*
                        emberLista.get(emberIndex).getKutyuk().size());
            uzenet = (int) (Math.random() * MAX_UZENETHOSSZ);
            aktKutyu = emberLista.get(emberIndex).getKutyuk().get(kutyuIndex);
            aktKutyu.uzenetetKuld(uzenet);
            if (aktKutyu instanceof OkosTelefon) {
                if (Math.random() < SZAZALEK) {
                    ((OkosTelefon) aktKutyu).kapcsolodik();
                } else {
                    ((OkosTelefon) aktKutyu).leKapcsolodik();
                }
                ((OkosTelefon) aktKutyu).internetezik((int) (Math.random() *
                                                    MAX_INTERNET_IDO));
            }
        }
    }
}
```

Az emberek listájára kattintva ismét célszerűbb az egérkattintás eseményt figyelni, hiszen előfordulhat, hogy a Használat gomb többszöri megnyomása után is mindig ugyanannak az embernek a diagnózisára vagyunk kíváncsiak.

```
private void lstEmberekMousePressed(java.awt.event.MouseEvent evt) {
    Ember ember = (Ember) lstEmberek.getSelectedValue();
    if (ember != null) {
        lblDiagnozis.setText(ember.diagnozis());
    }
}
```

A rendezés:

```
private void btnRendezActionPerformed(java.awt.event.ActionEvent evt) {
    vezerlo.rendezes();
}

public void rendezes() {
    Collections.sort(emberLista, new HuvelykSzerint());

    Ember.setRendezve(true);
    kutyuPanel.megjelenitEmberek(emberLista);
}
```