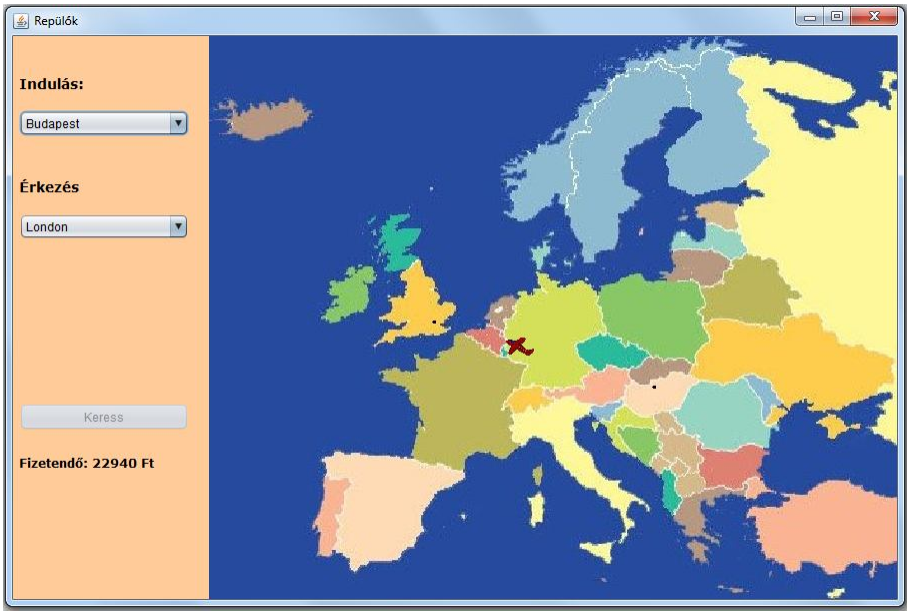
Repülőjáratok

Szimuláljon egy repülőjárat-keresést!

A 900x600 - as méretű felület jobboldalán legyen a 700x600 - as méretű térkép, a baloldalán pedig két combo-box – az egyikben az indulási, másikban az érkezési várost lehet kiválasztani.

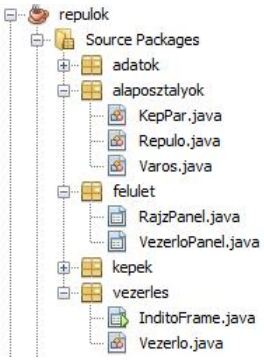


Ha kiválasztottunk egy várost (bármelyik combo-boxból), akkor jelenjen meg egy pont a város helyén, ha a Keress feliratú gombot is megnyomjuk, akkor az induló városból induljon el egy repülőgép a másik városba. Induláskor írja ki az útiköltséget is (távolság szorozva km-árral). Ha odaért, akkor egy csöpp várakozás után tűnjön el a kép is és a költség is. Ezután újabb városok között lehessen keresni. Mindig csak az aktuális városokat jelentő pontokat lehessen látni. A repülő orra mindig a célállomás irányába mutasson (elég csak jobbra vagy balra).

A szükséges adatok a *varosok.txt* fájlban találhatóak, de természetesen adatbázisból is dolgozhat. A repülőgép javasolt mérete: 30x20.

Néhány megoldásrészlet:

A felületkialakításban, adatbeolvasásban nincs semmi újdonság. Az adatok most combo-boxokba kerülnek, ezek kezeléséhez is modelleket használunk (`DefaultComboBoxModel`).



A `Varos` osztály a nevével és koordinátáival definiálható, illetve mivel itt írjuk le a kirajzolás módját is, ezért a várost jelző pötty sugarát is meg kell adnunk. (A pötty középpontja a város koordinátáival adható meg.)

Mivel a repülő mozog, ezért szálként kezeljük. Két pont között egyenes vonalban mozog, vagyis `run()` metódusa megírható a mozgásvariációkban mutatott változatok valamelyikével, de most egy negyedik mozgásváltozatot mutatunk meg.

Az osztály kódja:

```
public class Repulo extends Thread {

    private KepPar kepPar;
    private int kepSzelesseg, kepMagassag;

    private int kezdoX, kezdoY;
    private int vegX, vegY;
    private int aktX, aktY;
    private long kesleltetesiIdo;
    private long vegsoVarakozasiIdo;

    private Vezerlo vezerlo;

    private Image kep;

    public Repulo(KepPar kepPar, int kepSzelesseg, int kepMagassag,
                  int kezdoX, int kezdoY, int vegX, int vegY,
                  long kesleltetesiIdo, long vegsoVarakozasiIdo,
                  Vezerlo vezerlo) {
        this.kepPar = kepPar;
        this.kepSzelesseg = kepSzelesseg;
        this.kepMagassag = kepMagassag;
        this.kezdoX = kezdoX;
        this.kezdoY = kezdoY;
        this.vegX = vegX;
        this.vegY = vegY;
        this.kesleltetesiIdo = kesleltetesiIdo;
        this.vegsoVarakozasiIdo = vegsoVarakozasiIdo;
        this.vezerlo = vezerlo;
        this.kep = (kezdoX < kezdoY)? this.kepPar.getJobbKep() :
                                      this.kepPar.getBalKep();
    }

    public void rajzolas(Graphics g) {
        g.drawImage(kep, aktX, aktY, kepSzelesseg, kepMagassag, null);
    }
```

```
    @Override
    public void run() {

        double dx, meredekseg, tavX, tavY;

        aktX = kezdoX - kepSzelesseg / 2;
        aktY = kezdoY - kepMagassag / 2;
        tavX = 0;
        dx = (kezdoX < kezdoY) ? 1 : -1;
        meredekseg = (double) (vegY - kezdoY) / (double) (vegX - kezdoX);

        while (Math.abs(vegX - aktX - kepSzelesseg / 2) > 0) {

            pihen(kesleltetesiIdo);
            tavX += dx;
            tavY = meredekseg * tavX;
            aktX = (int) (kezdoX - kepSzelesseg / 2 + tavX);
            aktY = (int) (kezdoY - kepMagassag / 2 + tavY);
            vezerlo.frissit();
        }
        pihen(vegsoVarakozasiIdo);
        vezerlo.odaErt(this);
    }

    private void pihen(long ido) {
        try {
            Thread.sleep(ido);
        } catch (InterruptedException ex) {
            Logger.getLogger(Repulo.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

(A `KepPar` osztály együtt kezeli a balra-, illetve jobbra forduló repülőgép-képet.)

A vezérlőpanel eseményei:

```
    private void cmbIndulasItemStateChanged(java.awt.event.ItemEvent evt) {
        vezerlo.setInduloVaros((Varos) cmbIndulas.getSelectedItem());
    }

    private void cmbErkezesItemStateChanged(java.awt.event.ItemEvent evt) {
        vezerlo.setErkezoVaros((Varos) cmbErkezes.getSelectedItem());
    }

    private void btnKeresActionPerformed(java.awt.event.ActionEvent evt) {
        vezerlo.indit((Varos)induloModell.getSelectedItem(),
                      (Varos)erkezoModell.getSelectedItem());
    }
```

A `Vezerlo` osztály néhány részlete:

```
public void indit() {
    Varos.setSugar(VAROS_SUGAR);
    adatBevitel();
    vezerloPanel.comboBoxokbaIr(beolvasottVarosok);
}

public void rajzolas(Graphics g){
    if(induloVaros != null) induloVaros.rajzolas(g);
    if(erkezoVaros != null) erkezoVaros.rajzolas(g);
    if(induloVaros != null && erkezoVaros != null && repulo != null){
        repulo.rajzolas(g);
    }
}

public void frissit() {
    rajzPanel.repaint();
}

public void indit(Varos indulo, Varos erkezo) {
    this.induloVaros = indulo;
    this.erkezoVaros = erkezo;
    repulo = new Repulo(kepPar, REPULO_KEPSZELESSEG, REPULO_KEPMAGASSAG,
            induloVaros.getX(), induloVaros.getY(),
            erkezoVaros.getX(), erkezoVaros.getY(),
            REPULO_MOZGAS_KESLELTETO_IDO,
            REPULO_ERKEZES_UTANI_VARAKOZASI_IDO, this);
    repulo.start();

    vezerloPanel.gombAktivalas(false);
    arKalkulacio();
}

private void arKalkulacio() {
   double x = induloVaros.getX()- erkezoVaros.getX();
   double y = induloVaros.getY()- erkezoVaros.getY();
   int osszeg = (int)(Math.sqrt(x * x + y * y)* KM_EGYSEGAR);
   vezerloPanel.kiir("Fizetendő: " + osszeg + " Ft");
}

public void odaErt(Repulo aThis) {
    repulo = null;
    frissit();
    vezerloPanel.gombAktivalas(true);
    vezerloPanel.kiir("");
}

public void setInduloVaros(Varos induloVaros) {
    this.induloVaros = induloVaros;
    frissit();
}
```

Az `erkezoVaros` nevű változó settere ugyanilyen. Az `indit()` metódust a frame-ről hívjuk meg. A vezérlőpanel `comboBoxbaIr()` metódusa rakja be a modellekbe a beolvasott adatokat, a `gombAktivalas()` metódusa a Keress feliratú gomb aktivitását változtatja, a `kiir()` metódus pedig az eredmény-label feliratát adja meg.