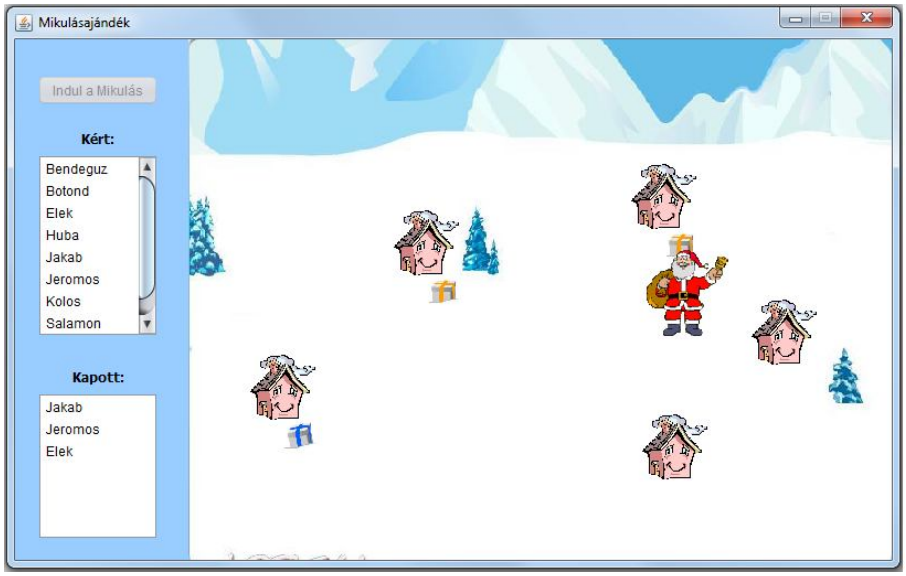
Mikulás 1.

Szimulálja a Mikulás munkáját egy Java programmal. A felület baloldala egy 150x500-as vezérlő felület, mellette egy 650x500-as havas táj látható. A program indulásakor már megjelennek a felső listában azok a nevek, akik potenciálisan a Mikulás „ügyfelei”. (Egy fájlból vagy adatbázisból olvassa be őket.) Ekkor házakat még nem lehet látni. Azok kattintással kerülnek a felületre, de csak akkor, ha a hegyvonulat alatt kattintunk, és még nem indult útjára a Mikulás. A kattintás helye a házikó-kép bal alsó sarkát jelöli ki.



Az indító gombra kattintva indul útjára a Mikulás. (Indulás után a gomb inaktívvá válik.)

A Mikulás megy házról házra. Mindegyiknél vár valamennyit (annyit, amennyi idő alatt odaért egyikből a másikba), majd lerak a ház mellé egy kis csomagot (a lehetséges csomagok közül véletlenül választja, hogy melyiket). Miután letette a csomagot, a baloldali alsó listában megjelenik a megajándékozott neve. (A nevet a fönti listából választjuk, szintén véletlenszerűen, és mindenki csak egyszer kaphat ajándékot.)

Ha az utolsó háznál is lerakta az ajándékot, akkor eltűnik, de a csomagok maradnak.

A csomag a ház jobb alsó sarkához kerül, a mikuláskép középpontja azonos lehet a házat kijelölő kattintás helyével.

Javasolt képméretek: csomag: 30x30, ház 60x60, Mikulás 80x80. A látóhatár mondjuk, 100 pixelnyire van a kép tetejétől.

Néhány megoldási ötlet:

A pozíciók figyelembevételével az egyes rajzolások:

`Csomag`:

```
    // kx, ky bal fölső sarok
    public void rajzolas(Graphics g){
         g.drawImage(kep, kx, ky, szelesseg, magassag, null);
    }
```

`Haz`:

```
    // kx, ky : bal alsó sarok
    public void rajzolas(Graphics g){
         g.drawImage(kep, kx, ky-magassag, szelesseg, magassag, null);
    }
```

`Mikulas`:

```
    // aktX, aktY a kép közepe - a ház jobb alsó sarkánál
     public void rajzolas(Graphics g){
         g.drawImage(kep, aktX-szelesseg/2, aktY-magassag/2, szelesseg, magassag, null);
    }
```

A feladatban egyetlen szálat indítunk, annyi az újdonság benne, hogy most nem egy, hanem egymás után több egyenes vonalú mozgásrészletet kell megtennie a figurának. Ehhez ismernie kell a meglátogatásra váró házakat. Egymás utáni két ház között végez egyenes vonalú egyenletes mozgást. Ezt most úgy oldottuk meg, hogy fixen megadtuk, hány lépést tegyen meg két ház között. Így a házak melletti várakozási idő is könnyen számolható.

A `Mikulas` osztály kódja (a rajzolás nélkül):

```
public class Mikulas extends Thread {

    private Image kep;
    private int szelesseg;
    private int magassag;

    private int aktX, aktY;
    private List<Haz> hazak;
    private long pihenoIdo;
    private Vezerlo vezerlo;

    public Mikulas(Image kep, int szelesseg, int magassag, long pihenoIdo,
                   Vezerlo vezerlo) {
        this.kep = kep;
        this.szelesseg = szelesseg;
        this.magassag = magassag;
        this.pihenoIdo = pihenoIdo;
        this.vezerlo = vezerlo;
    }
```

```
    @Override
    public void run() {
        Haz haz;
        double kezdx, kezdy, vegx, vegy, dx, dy;
        int n = 100;
        for(int i=1; i< hazak.size(); i++){

            haz = hazak.get(i-1);
            kezdx = haz.getKx();
            kezdy = haz.getKy();

            pihen(n*pihenoIdo);
            vezerlo.ajandekoz(haz);

            haz = hazak.get(i);
            vegx = haz.getKx();
            vegy = haz.getKy();

            dx = (vegx-kezdx)/n;
            dy = (vegy-kezdy)/n;

            for(int j=0; j<=n; j++){
                aktX = (int)(kezdx + j*dx);
                aktY = (int)(kezdy + j*dy);
                pihen(pihenoIdo);
                vezerlo.frissit();
            }
        }
        pihen(n*pihenoIdo);
        vezerlo.ajandekoz(hazak.get(hazak.size()-1));
        vezerlo.vege();
    }

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

    public void setHazak(List<Haz> hazak) {
        this.hazak = hazak;
        aktX = hazak.get(0).getKx();
        aktY = hazak.get(0).getKy();
    }
}
```

A vezérlő hivatkozott metódusai (a frissítés egyértelmű, azt most nem közöljük):

```
private Image[] csomagKepek =
    {new ImageIcon(this.getClass().getResource("/kepek/csomag1.jpg")).getImage(),
    new ImageIcon(this.getClass().getResource("/kepek/csomag2.jpg")).getImage(),
    new ImageIcon(this.getClass().getResource("/kepek/csomag3.jpg")).getImage()};

public void ajandekoz(Haz haz) {
    int index = (int) (Math.random() * csomagKepek.length);
    csomagok.add(new Csomag(haz.getKx() + Haz.getSzelesseg() / 2,
            haz.getKy(), haz.getNev(), csomagKepek[index]));
    vezerloPanel.kiirKapo(haz.getNev());
}

public void vege() {
    miki = null;
    frissit();
}
```

(A vezérlőpanel `kiirKapo()` metódusa az ajándékot kapók modelljébe írja a házhoz tartozó nevet.)

Nyilván azt is a vezérlő mondja meg, hogy a rajzpanelnek mit kell rajzolnia:

```
private List<Csomag> csomagok = new ArrayList<Csomag>();
private List<Haz> hazak = new ArrayList<Haz>();

private boolean elindult = false;
private Mikulas miki;

public void rajzolas(Graphics g) {
     for(Haz haz: hazak){
         haz.rajzolas(g);
     }
     for(Csomag csomag: csomagok){
         csomag.rajzolas(g);
     }
     if(elindult && miki != null) {
         miki.rajzolas(g);
     }
}
```

A rajzpanelre való kattintáskor a vezérlő `kattintottak()` metódusát hívjuk meg. Ekkor a megadott nevek közül véletlenszerűen ki kell választani egy nevet (de minden név csak egyszer szerepelhet, ezért legegyszerűbb, ha utána töröljük a választottat), és létrehozni a hozzá tartozó házat – feltéve persze, hogy a látóhatár alá kattintottunk.

```
public void kattintottak(int x, int y) {
    String nev = valasztottNev();
    if (nev != null && y > LATOHATAR && !elindult) {
        hazak.add(new Haz(x, y, nev, hazKep));
        frissit();
    }
}

private String valasztottNev() {
    int index = (int)(Math.random()*nevek.size());
    String nev;
    if(index>=0) {
        nev = nevek.get(index);
        nevek.remove(nev);
    }else nev = null;
    return nev;
}
```

A vezérlőpanelen lévő gomb megnyomásakor a vezérlő mikulásindító metódusát hívjuk meg:

```
public void mikulastIndit() {
    miki = new Mikulas(mikulasKep, MIKULASKEP_SZELESSEG, MIKULASKEP_MAGASSAG,
                       MIKULAS_PIHENOIDO, this);
    miki.setHazak(hazak);
    miki.start();
    elindult = true;
    frissit();
}
```

A vezérlő példányt szokásos módon a frame hozza létre, és ez hívja meg a vezérlő indító metódusát. Ebben megadhatjuk a statikus adatokat (a házak, illetve a csomagok méretét kezelhetjük statikusan), és beolvassuk a megajándékozandók névsorát. Természetesen ezeket a neveket meg is jelenítjük a megfelelő listafelületen.