# Dogodkovno vodeno programiranje

# Strukturna razlika

Grafično podprti programi (GUI) se pomembno razlikujejo od tekstovno (console based) usmerjenih

Pri konzolnih programih programer aplikacije določa, kdaj bi prišlo do vnosa podatkov ali izpisa rezultatov. Govorimo o postopkovnem programiranju (procedural programming)

Pri grafično podprtih (GUI) lahko uporabnik izvaja akcije kadarkoli. Nemogoče je predvideti zaporedje akcij:

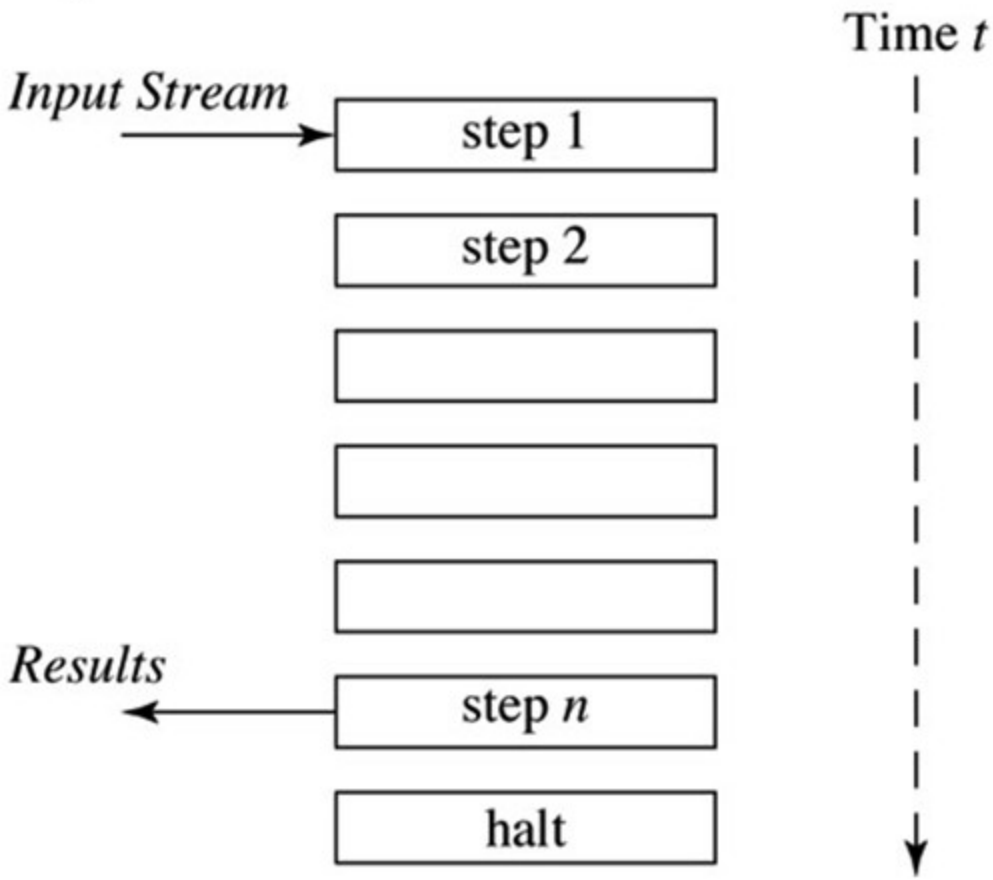
- Pritisk ,a gumb,
- Zapiranje okna
- Pomiki z miško,...

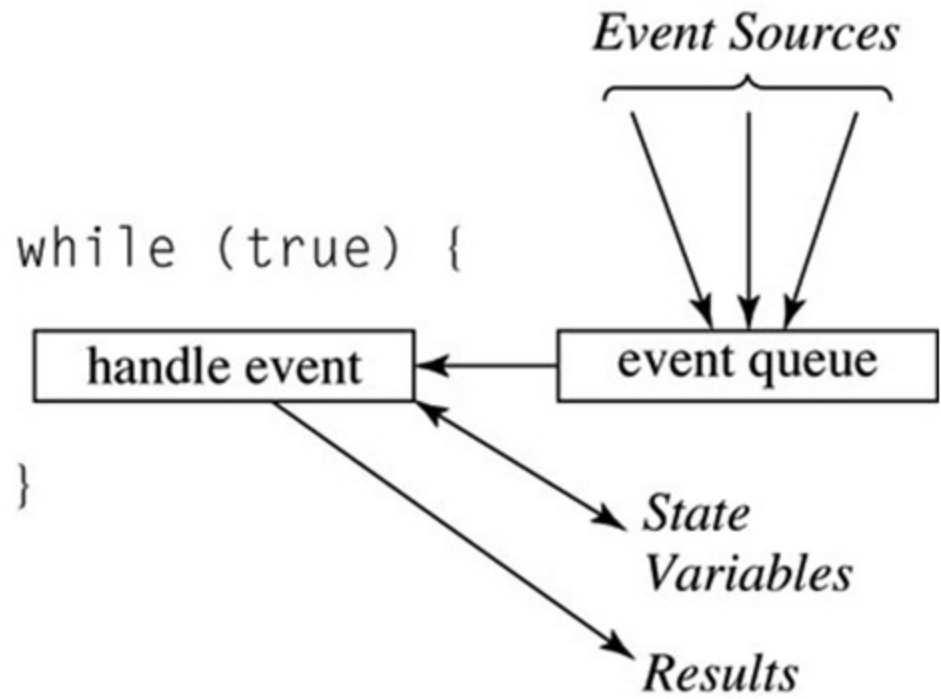
Taki programi so dogodkovno vodeni (event driven)

# Konceptualna razlika v programiranju

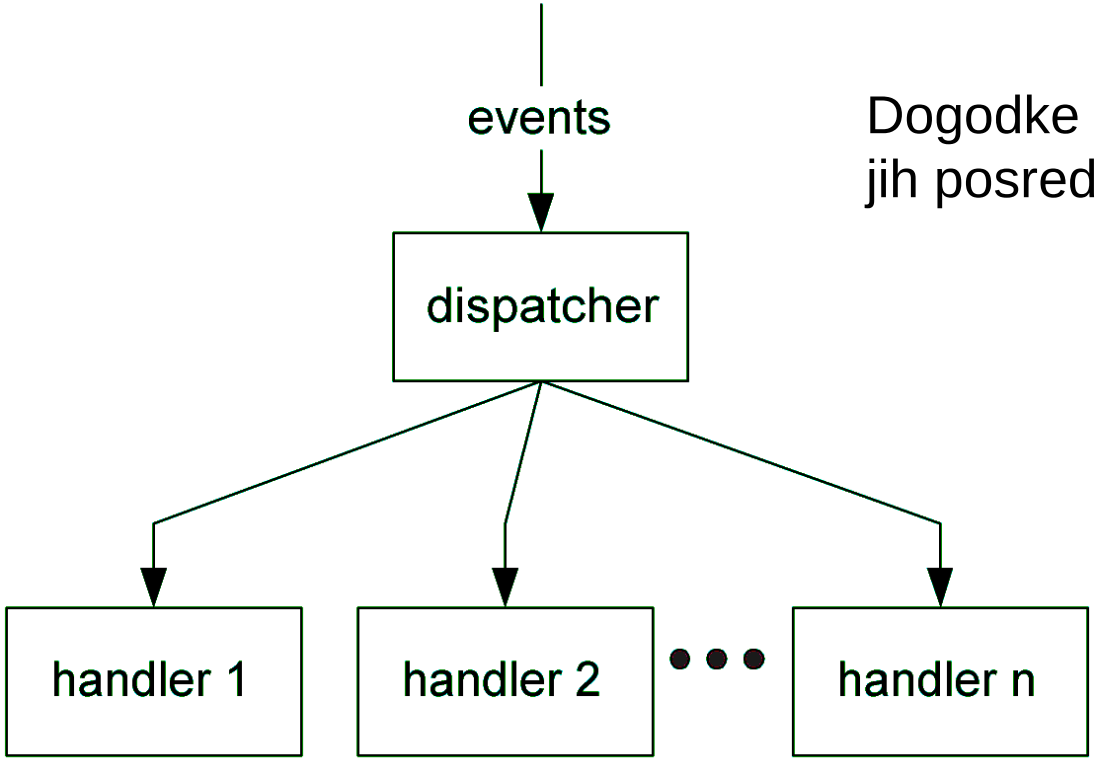
Proceduralen program

Dogodkovno voden program



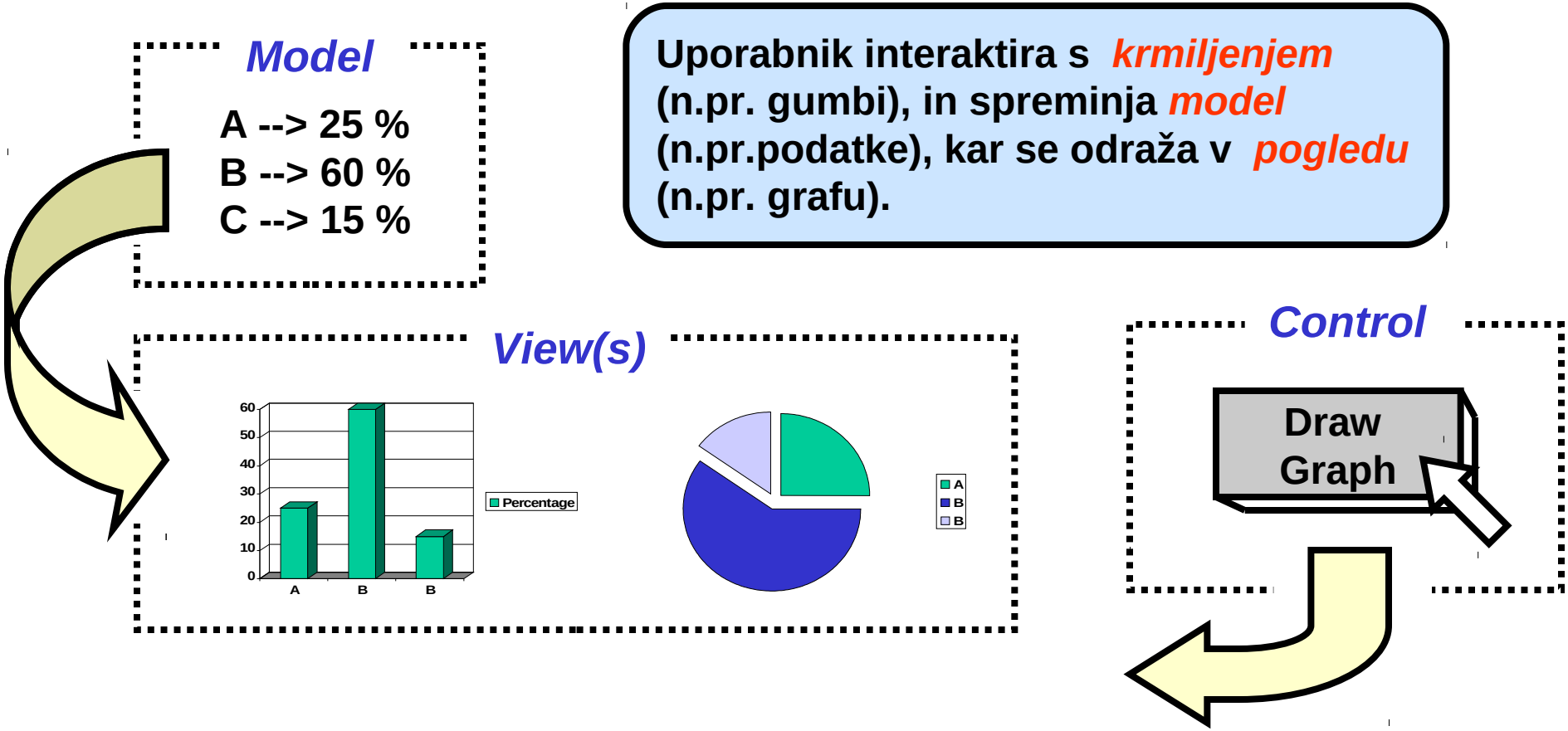
# Splošen koncept rokovalnikov




Dogodke prejema modul “dispečer” in jih posreduje ustreznemu “rokovalniku”

# Koncept MVC

*Koncept MVC (Model, View, Controller) razbija aplikacijo oziroma vmesnike na tri dele : model, pogled in krmiljenje.*



# Dogodkovno voden koncept programiranja

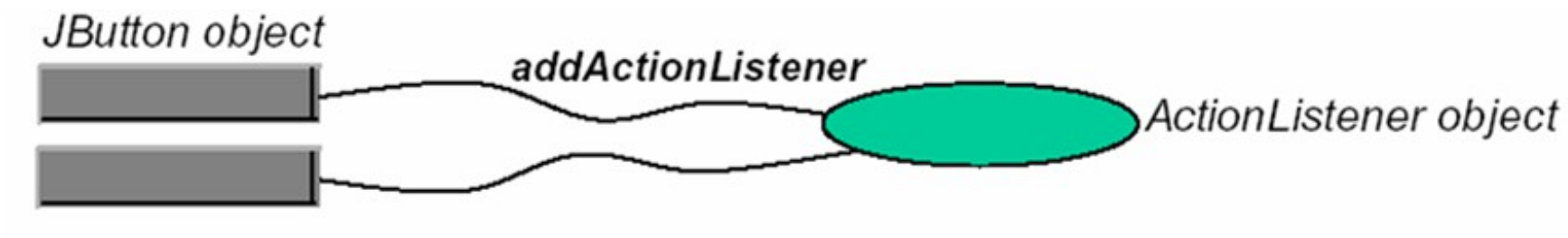
Pišemo le kodo, ki pomeni odziv na dogodke

Takim funkcijam pravimo včasih “message handlers”

V javi imamo za to poslušalce (listeners)

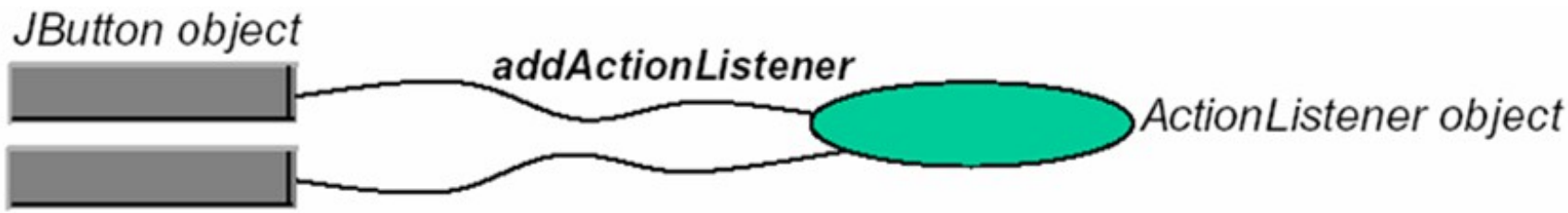
**Kako je to pri Javi?**

- ActionListener opazuje dogodke v zvezi z miško
- WindowListener opazuje dogodke v zvezi z okni



# Obravnavanje dogodkov

- Dogodke povzroča uporabnik, ko interaktira z nekim objektom GUI (na primer s klikom na gumb)
- Dogodek aktivira poslušalca ActionListener, ki je navezan na ta objekt GUI




- Sistem avtomatično kliče naslednjo metodo ActionListener:

```
actionPerformed (event e)
```

- Ta metoda mora razpoznati , GUI objekt je izvor dogodka, in dogodek obravnavati, na primer:

```
Object ob = e.getSource();
if (ob == myButton)
   ... koda, ki predstavlja odziv na klik na gumb
```

# Tipi dogodkov, odzivne funkcije

| Event Type | Event Source | Handler Required |
|---|---|---|
| Button selection | Button | actionPerformed |
| Text entry | TextField | actionPerformed |
| Menu selection | Choice | itemStateChanged |
| Mouse | | mousePressed<br>mouseReleased<br>mouseClicked<br>mouseExited<br>mouseEntered |
| Mouse motion | | mouseDragged<br>mouseMoved |

# Poslušalec dogodkov v ločeni datoteki

```
import java.applet.Applet;
import java.awt.*;

public class ClickReporter extends Applet {
    public void init() {
        setBackground(Color.YELLOW);
        addMouseListener(new ClickListener());
    }
}
```

```
import java.awt.event.*;
public class ClickListener extends MouseAdapter {
public void mousePressed(MouseEvent event) {
    System.out.println("Mouse pressed at (" +
        event.getX() + "," +
        event.getY() + ").");
    }
}
```

# Posplošitev primera

*Kaj, če “ClickListener” želi narisati krog, kjerkoli kliknemo z miško?*

*Zakaj ne bi kar klicali getGraphics in tako dobili objekt Graphics, s katerim naj bi risali?*

**Splošna rešitev:**

– Kličemo event.getSource in tako dobimo referenco okna ali grafične komponente, ki je generirala dogodek

– pretvorimo tip rezultata (cast) v skladu s potrebo

– Kličemo metode na tej referenci

# Posplošeni primer s poslušalcem

```
import java.applet.Applet;
import java.awt.*;
public class CircleDrawer1 extends Applet {
     public void init() {
         setForeground(Color.BLUE);
        addMouseListener(new CircleListener());
    }
}
```

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class CircleListener extends MouseAdapter {
    private int radius = 25;
    public void mousePressed(MouseEvent event) {
        Applet app = (Applet)event.getSource();
        Graphics g = app.getGraphics();
        g.fillOval(event.getX()-radius,
        event.getY()-radius, 2*radius, 2*radius);
    }
}
```

# Drug način:Implementacija poslušalca

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class CircleDrawer2 extends Applet implements MouseListener {
    private int radius = 25;
    public void init() {
        setForeground(Color.BLUE);
        addMouseListener(this);
    }
    public void mouseEntered(MouseEvent event) { }
    public void mouseExited(MouseEvent event) { }
    public void mouseReleased(MouseEvent event) { }
    public void mouseClicked(MouseEvent event) { }
    public void mousePressed(MouseEvent event) {
        Graphics g = getGraphics();
        g.fillOval(event.getX()-radius,
        event.getY()-radius,  2*radius,  2*radius);
     }
}
```

# Adapterji in vmesniki: primerjava

**Kaj, če se zmotimo pri podpisu metode?**

- public void mousepressed(MouseEvent e)
- public void mousePressed()

- **Vmesniki**
  - Prevajalnik ugotovi napako
- **Adapterji**
  - Med prevajanjem ne pride do napake, med izvajanjem na kliku z miško pa se ne bo zgodilo nič

# Tretji način (imenovani notranji razredi)

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class CircleDrawer3 extends Applet {
    public void init() {
        setForeground(Color.BLUE);
        addMouseListener(new CircleListener());
    }
    private class CircleListener extends MouseAdapter {
        private int radius = 25;
        public void mousePressed(MouseEvent event) {
            Graphics g = getGraphics();
            g.fillOval(event.getX()-radius,
              event.getY()-radius, 2*radius, 2*radius);
        }
    }
}
```

Razred znotraj razreda

# 4. način: neimenovani notranji razredi

```
public class CircleDrawer4 extends Applet {
    public void init() {
        setForeground(Color.BLUE);
        addMouseListener (new MouseAdapter() {
            private int radius = 25;
            public void mousePressed(MouseEvent event) {
                Graphics g = getGraphics();
                g.fillOval(event.getX()-radius, event.getY()-radius,
                    2*radius,  2*radius);
            }
        });
    }
}
```

# Prednosti in slabosti teh strategij

**Ločen poslušalec**

– Prednosti

- Adapter lahko podedujemo in zato lahko nepotrebne metode ignoriramo
- Lažje rokovanje z ločenimi razredi

– Slabost

- Za klic metod v glavnem oknu potrebujemo dodaten korak

- **Glavno okno implementira vmesnik**

– Prednost

- Ni potrebno dodatnih korakov za klic metod v glavnem oknu

– Slabost

- Implementirati moramo tudi metode, ki nas ne zanimajo

# Prednosti in slabosti (nadaljevanje)

**Imenovani notranji razredi**

– Prednosti
  - lahko podedujemo adapter in zato ignoriramo neuporabljene metode
  - Niso potrebni dodatni koraki za klic metod v glavnem oknu

– Slabost
  - težje razumnljivo

- **Anonimen notranji razred**

– Prednosti
  - Iste kot pri imenovanih notranjih razredih
  - Še krajše

– Slabost
  - Še težje razumljivo

# Popoln primer: Simple whiteboard (1)

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class SimpleWhiteboard extends Applet {
    protected int lastX=0, lastY=0;
    public void init() {
        setBackground(Color.WHITE);
        setForeground(Color.BLUE);
        addMouseListener(new PositionRecorder());
        addMouseMotionListener(new LineDrawer());
    }
    protected void record(int x, int y) {
        lastX = x; lastY = y;
    }
```

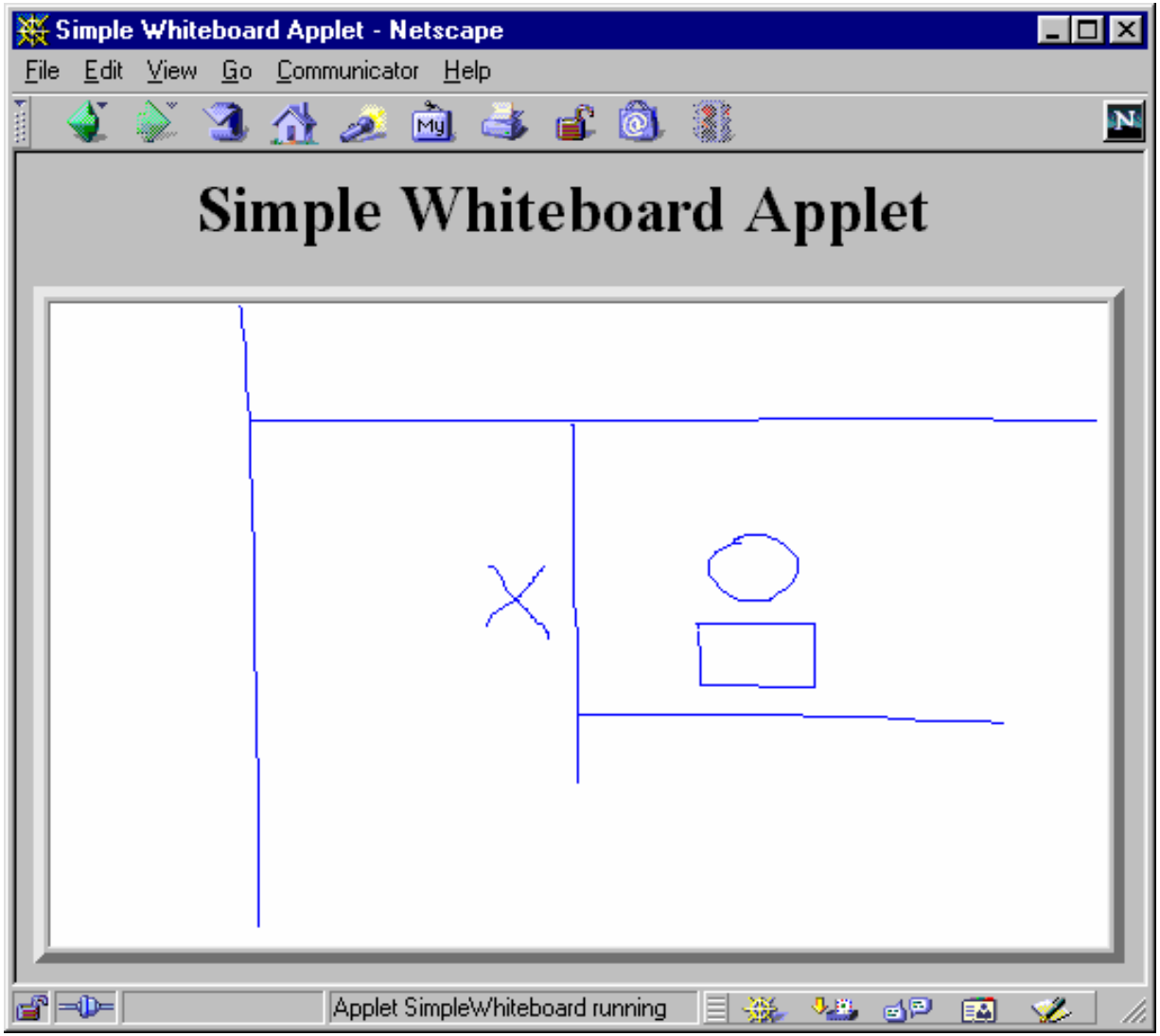
# Popoln primer: Simple whiteboard (2)

```
private class PositionRecorder extends MouseAdapter {

    public void mouseEntered(MouseEvent event) {
        requestFocus(); // Plan ahead for typing
        record(event.getX(), event.getY());
    }

    public void mousePressed(MouseEvent event) {
       record(event.getX(), event.getY());
    }
}
```

# Popoln primer: Simple whiteboard (3)

```
    private class LineDrawer extends MouseMotionAdapter {

    public void mouseDragged(MouseEvent event) {
            int x = event.getX();
            int y = event.getY();
            Graphics g = getGraphics();
            g.drawLine(lastX, lastY, x, y);
            record(x, y);
        }
    }
}
```

# Popoln primer: Simple whiteboard (4)



# Dodajmo dogodke s tipkovnico

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Whiteboard extends SimpleWhiteboard {
    protected FontMetrics fm;
    public void init() {
        super.init();
        Font font = new Font("Serif", Font.BOLD, 20);
        setFont(font);
        fm = getFontMetrics(font);
        addKeyListener(new CharDrawer());
}
```

# Dodajmo tipkovnico (2)

```
    private class CharDrawer extends KeyAdapter {
        // Ko pritisnemo znak, se ta narise , pozicija se premakne v desno.
        public void keyTyped(KeyEvent event) {
            String s = String.valueOf(event.getKeyChar());
            getGraphics().drawString(s, lastX, lastY);
            record(lastX + fm.stringWidth(s), lastY);
        }
    }
}
```