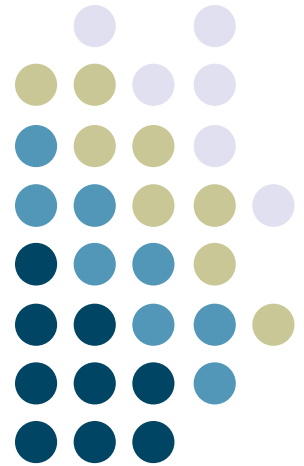


Java Swing GUI Programming 4



John Stasko

CS 6452

Prototyping Interactive Systems

Learning Objectives



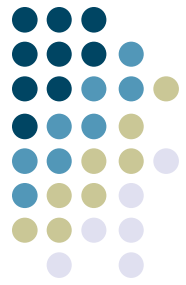
- New UI components
 - File chooser, Text area, Color chooser, Slider, Combo box (menu)
- Tooltips and short-cuts
- Mouse events
 - Dynamic drawing
- Key events
- Timer events and animation

Useful Components



- Let's examine some other UI components that come in handy

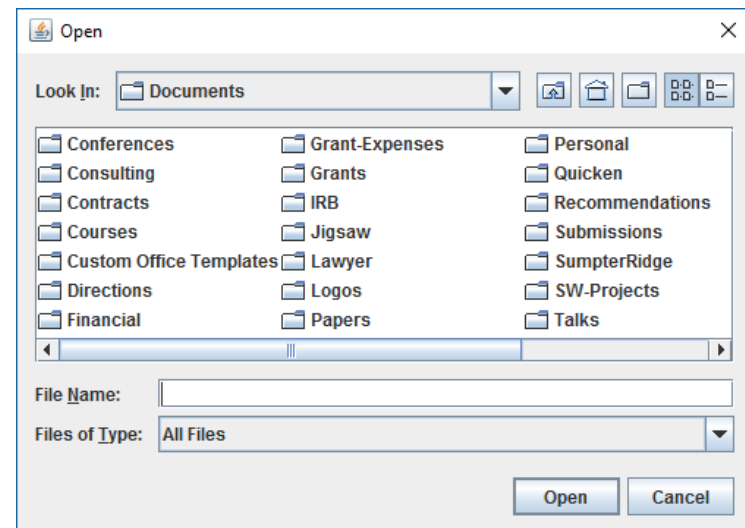
Choosing Files



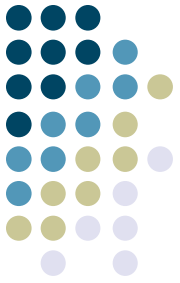
Convenient way to select files – `JFileChooser`

To use

1. Call constructor
 2. Call `showOpenDialog` method that displays the chooser
- Returns `int` (`JFileChooser.APPROVE_OPTION`)



Bigger Text Areas

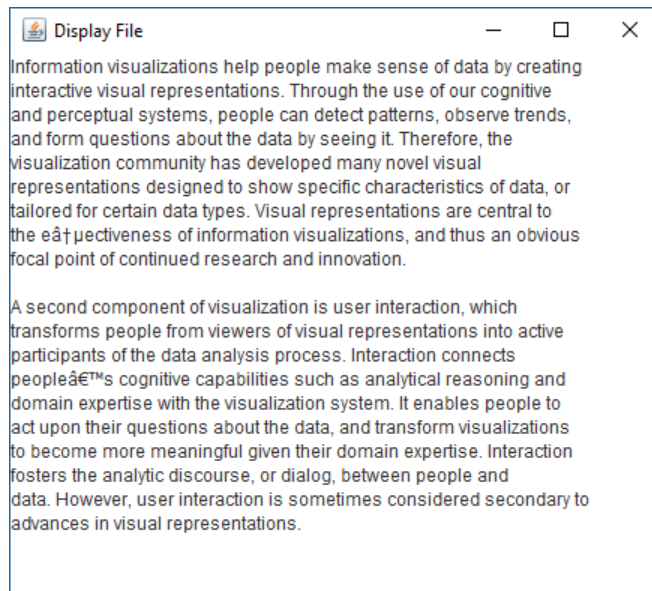


`JTextArea` – Multiple rows of text

Constructor `JTextArea(int rows, int cols)`

`setText` method puts text in there

By default, editable – Change via `.setEditable(false)`



DisplayFile program

Program

```
public class DisplayFile
{
    public static void main (String[] args) throws IOException
    {
        JFrame frame = new JFrame ("Display File");
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

        JTextArea ta = new JTextArea (20, 30);
        JFileChooser chooser = new JFileChooser();

        int status = chooser.showOpenDialog (null);

        if (status != JFileChooser.APPROVE_OPTION)
            ta.setText ("No File Chosen");
        else
        {
            File file = chooser.getSelectedFile();
            Scanner scan = new Scanner (file);

            String info = "";
            while (scan.hasNext())
                info += scan.nextLine() + "\n";

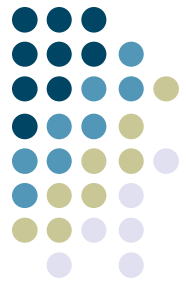
            ta.setText (info);
        }

        frame.getContentPane().add (ta);
        frame.pack();
        frame.setVisible(true);
    }
}
```



Uses both
components

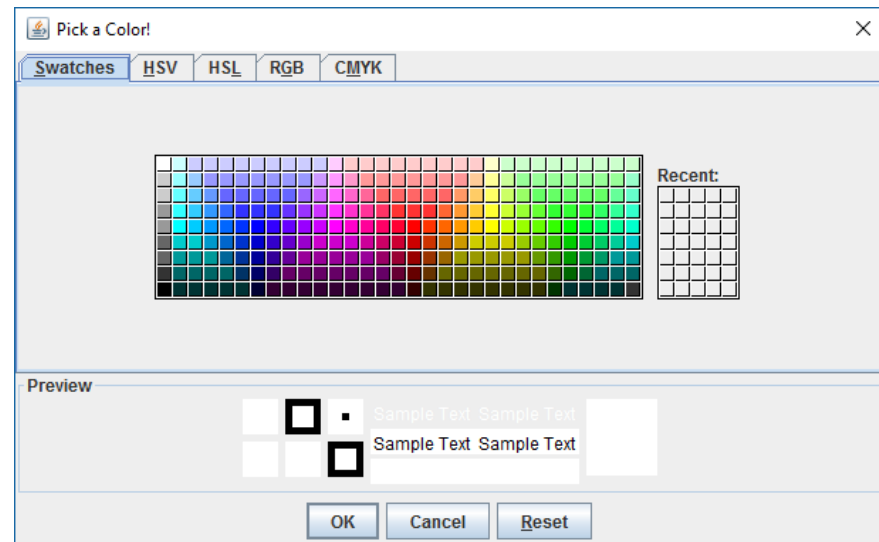
Choosing Colors



JColorChooser – Special color choice dialog

```
JColorChooser.showDialog(Component parent, String s, Color initCol)
```

Different method, just invoke static method rather than create an object



DisplayColor program

Program



```
public class DisplayColor
{
    public static void main (String[] args)
    {
        JFrame frame = new JFrame ("Display Color");
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

        JPanel colorPanel = new JPanel();
        colorPanel.setBackground (Color.white);
        colorPanel.setPreferredSize (new Dimension (300, 100));

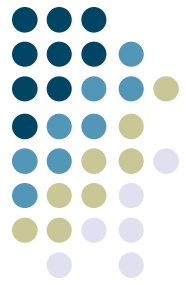
        frame.getContentPane().add (colorPanel);
        frame.pack();
        frame.setVisible(true);

        Color shade = Color.white;
        int again;

        do
        {
            shade = JColorChooser.showDialog (frame, "Pick a Color!", shade);
            colorPanel.setBackground (shade);

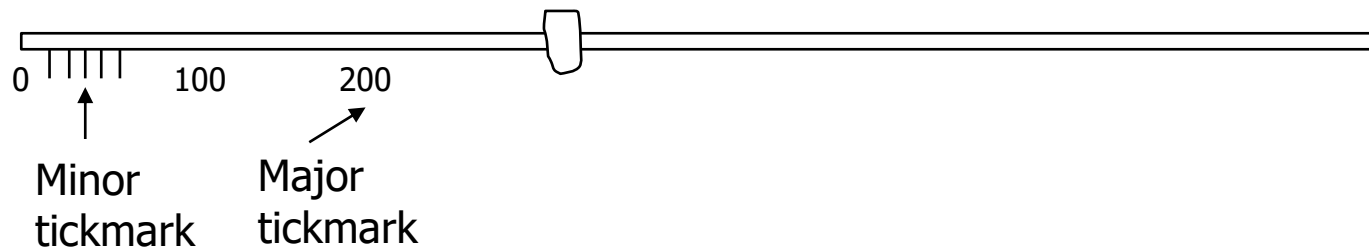
            again = JOptionPane.showConfirmDialog (null,
                "Display another color?");
        }
        while (again == JOptionPane.YES_OPTION);
    }
}
```


Choosing a Value

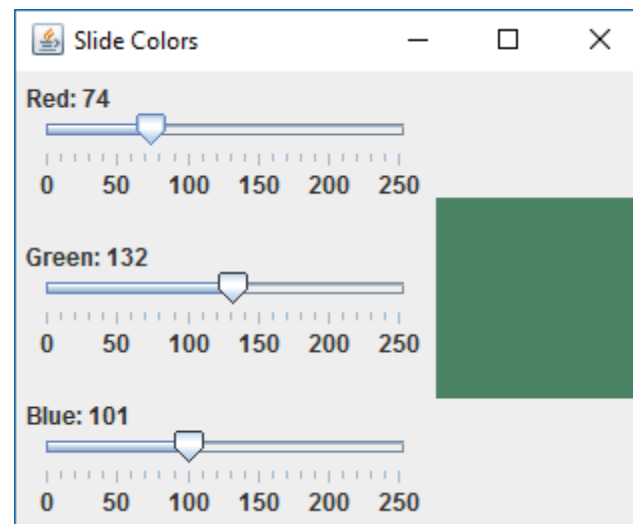


JSlider – Java scrollbar

```
JSlider(HorizOrVert, minval, maxval, startval)
```



Uses `ChangeListener` **interface** and `stateChanged()` **method**
Generates `ChangeEvent` **object**



Code in t-square

SlideColor program

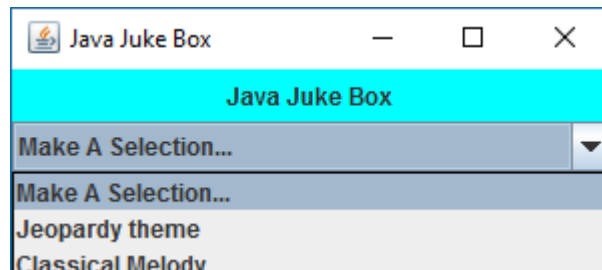
Pull-down Menus



`JComboBox` takes an array of strings (menu choices)

Menu choice triggers `ActionPerformed`

`combo.getSelectedIndex()` gets index of choice



JukeBox program

Key Code



```
URL url1, url2;
url1 = url2 = null;

// Obtain and store the audio clips to play
try
{
    url1 = new URL ("file", "localhost", "jeopardy.au");
    url2 = new URL ("file", "localhost", "classical.wav");
}
catch (Exception exception) {}

music = new AudioClip[7];
music[0] = null; // Corresponds to "Make a Selection..."
music[1] = JApplet.newAudioClip (url1);
music[2] = JApplet.newAudioClip (url2);

// Create the list of strings for the combo box options
String[] musicNames = {"Make A Selection...", "Jeopardy theme",
    "Classical Melody"};

musicCombo = new JComboBox (musicNames);
musicCombo.setAlignmentX (Component.CENTER_ALIGNMENT);

// Set up the buttons
playButton = new JButton ("Play", new ImageIcon ("play.gif"));
playButton.setBackground (Color.white);
playButton.setMnemonic ('p');

musicCombo.addActionListener (new ComboListener());
playButton.addActionListener (new ButtonListener());

current = null;
}
// continued..
```

```
//continuing..
```

```
private class ComboListener implements ActionListener
{
    public void actionPerformed (ActionEvent event)
    {
        if (current != null)
            current.stop();

        current = music[musicCombo.getSelectedIndex()];
    }
}

private class ButtonListener implements ActionListener
{
    public void actionPerformed (ActionEvent event)
    {
        if (current != null)
            current.stop();

        if (event.getSource() == playButton)
            if (current != null)
                current.play();
    }
}
}
```

JukeBox program

Neat Stuff



Add tooltips to buttons

```
JButton button = new JButton("Compute");  
button.setToolTipText("Calculates total cost");
```

Add a short-cut key method

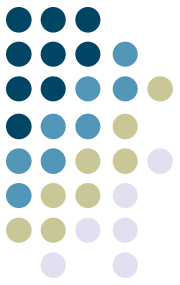
```
button.setMnemonic('C');
```

ALT-C activates it

Disable a button

```
button.setEnabled(false);
```

Mouse Events



Mouse events

Pressed

Released

Clicked – no movement in between

Entered (component)

Exited

Get (order):

press

release

click

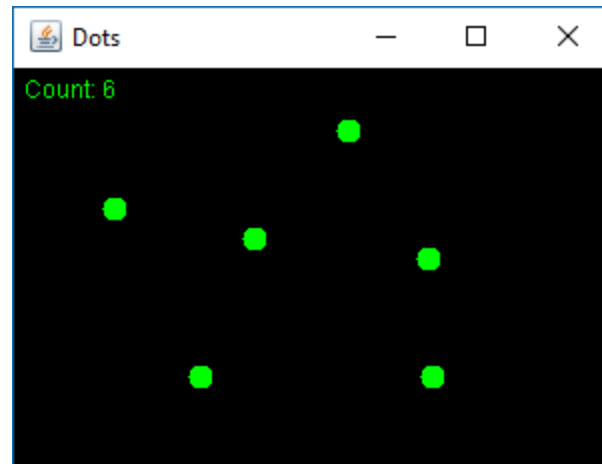
Mouse motion events

Moved (get lots of them)

Dragged

We have to decide what we want to listen for

Example Program I



Dots program

Example Program I



```
public class DotsPanel extends JPanel
{
    private final int SIZE = 6; // radius of each dot
    private ArrayList<Point> pointList;

    public DotsPanel()
    {
        pointList = new ArrayList<Point>();

        addMouseListener (new DotsListener());

        setBackground (Color.black);
        setPreferredSize (new Dimension(300, 200));
    }

    public void paintComponent (Graphics page)
    {
        super.paintComponent(page);

        page.setColor (Color.green);

        for (Point spot : pointList)
            page.fillOval (spot.x-SIZE, spot.y-SIZE, SIZE*2, SIZE*2);

        page.drawString ("Count: " + pointList.size(), 5, 15);
    }
}
// continued...
```

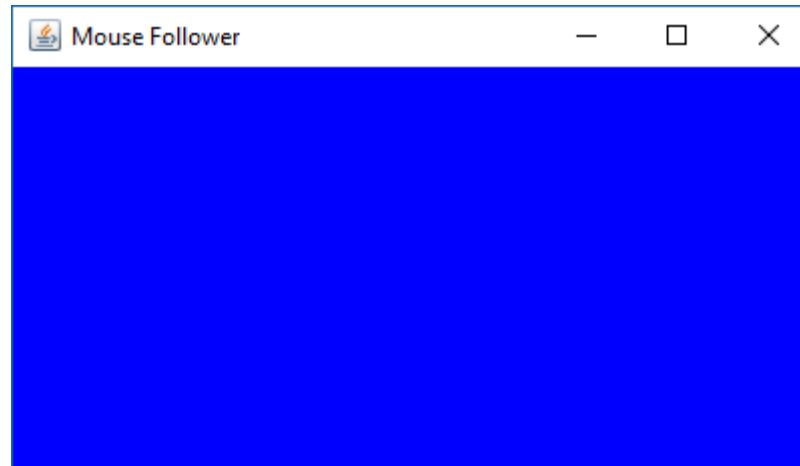
//continuing...

```
private class DotsListener implements MouseListener
{
    public void mousePressed (MouseEvent event)
    {
        pointList.add(event.getPoint());
        repaint();
    }
    public void mouseClicked (MouseEvent event) {}
    public void mouseReleased (MouseEvent event) {}
    public void mouseEntered (MouseEvent event) {}
    public void mouseExited (MouseEvent event) {}
}
}
```

Mouse listening has the
five events

Dots program

Example Program 2



(Color changes if in left or right)

MouseFollow program

Example Program 2



```
public class MouseFollowPanel extends JPanel
{
    private boolean left = true;

    public MouseFollowPanel()
    {
        LineListener listener = new LineListener();
        addMouseMotionListener (listener);

        setBackground (Color.black);
        setPreferredSize (new Dimension(400, 200));
    }

    public void paintComponent (Graphics page)
    {
        super.paintComponent(page);
        if (left == true)
            setBackground(Color.blue);
        else
            setBackground(Color.red);
    }
}

// continuing...
```

Two motion events

```
//continuing...

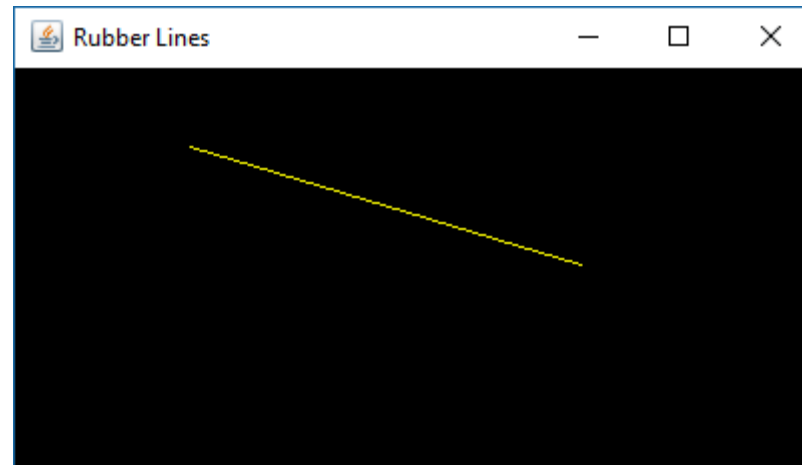
    private class LineListener implements
    MouseMotionListener
    {
        public void mouseDragged (MouseEvent event)
        {
            Point p;

            p = event.getPoint();
            if (p.getX() < 200)
                left = true;
            else
                left = false;
            repaint();
        }

        public void mouseMoved (MouseEvent event)
        {
            Point p;
            p = event.getPoint();
            if (p.getX() < 200)
                left = true;
            else
                left = false;
            repaint();
        }
    }
}
```

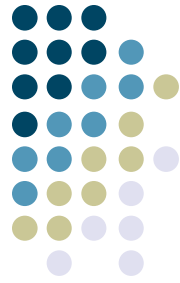
MouseFollow program

Example Program 3



RubberLines program

Example Program 3



```
public class RubberLinesPanel extends JPanel
{
    private Point point1 = null, point2 = null;

    public RubberLinesPanel()
    {
        LineListener listener = new LineListener();
        addMouseListener (listener);
        addMouseMotionListener (listener);

        setBackground (Color.black);
        setPreferredSize (new Dimension(400, 200));
    }

    public void paintComponent (Graphics page)
    {
        super.paintComponent (page);

        page.setColor (Color.yellow);
        if (point1 != null && point2 != null)
            page.drawLine (point1.x, point1.y, point2.x, point2.y);
    }
    // continues...
```

//continuing...

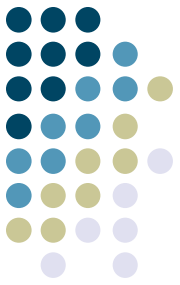
```
private class LineListener implements MouseListener,
MouseMotionListener
{
    public void mousePressed (MouseEvent event)
    {
        point1 = event.getPoint();
    }

    public void mouseDragged (MouseEvent event)
    {
        point2 = event.getPoint();
        repaint();
    }

    public void mouseClicked (MouseEvent event) {}
    public void mouseReleased (MouseEvent event) {}
    public void mouseEntered (MouseEvent event) {}
    public void mouseExited (MouseEvent event) {}
    public void mouseMoved (MouseEvent event) {}
}
}
```

RubberLines program

Things to Try



- Comment out `super.paintComponent()`
- Put `point1 = point2` in `dragged` first
 - That doesn't work – Why?
- Put `point1 = point2` in `repaint()`

Example Program 4



Draw program

Example Program 4



```
public class DrawPanel extends JPanel
{
    private Point point1 = null, point2 = null;

    public DrawPanel()
    {
        LineListener listener = new LineListener();
        addMouseListener (listener);
        addMouseMotionListener (listener);

        setBackground (Color.black);
        setPreferredSize (new Dimension(400, 200));
    }

    public void paintComponent (Graphics page)
    {
        //super.paintComponent (page);
        // This will wipe out the previous drawing

        page.setColor (Color.red);
        if (point1 != null && point2 != null)
            page.drawLine (point1.x, point1.y, point2.x, point2.y);
    }
}

// continued..
```

Note

//continuing...

```
private class LineListener implements MouseListener,
    MouseMotionListener
{
    public void mousePressed (MouseEvent event)
    {
        point1 = event.getPoint();
    }

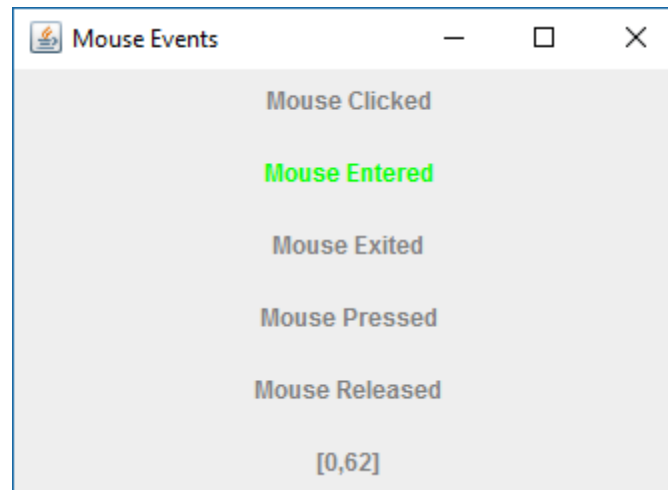
    public void mouseDragged (MouseEvent event)
    {
        point1 = point2;    // added this line to
                           // update old position

        point2 = event.getPoint();
        repaint();
    }

    public void mouseClicked (MouseEvent event) {}
    public void mouseReleased (MouseEvent event) {}
    public void mouseEntered (MouseEvent event) {}
    public void mouseExited (MouseEvent event) {}
    public void mouseMoved (MouseEvent event) {}
}
}
```

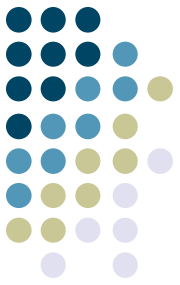
Draw program

Example Program 5



MouseExample program

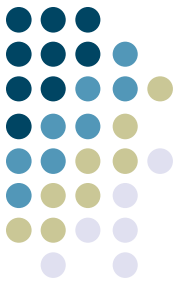
Code



- Communicates the different style of events
- Very different style of program
 - No Panel class way we've been doing it
- Try click down, then release outside

In t-square

Key Events



Generated when key is pressed

`KeyListener` – Interface for handling key events

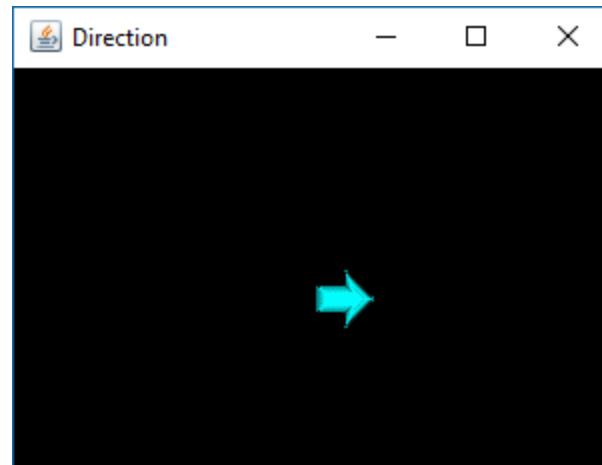
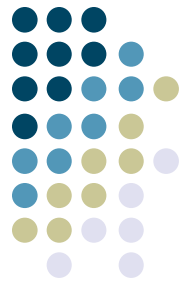
```
keyPressed(KeyEvent evt)  
keyReleased(KeyEvent evt)  
keyTyped(KeyEvent evt)
```

Three types of events



Example Program

Georgia
Tech



Direction program

Example Program



```
public class DirectionPanel extends JPanel
{
    private final int WIDTH = 300, HEIGHT = 200;
    private final int JUMP = 10; // increment for image movement

    private final int IMAGE_SIZE = 31;

    private ImageIcon up, down, right, left, currentImage;
    private int x, y;

    public DirectionPanel()
    {
        addKeyListener(new DirectionListener());

        x = WIDTH / 2;
        y = HEIGHT / 2;

        up = new ImageIcon("arrowUp.gif");
        down = new ImageIcon("arrowDown.gif");
        left = new ImageIcon("arrowLeft.gif");
        right = new ImageIcon("arrowRight.gif");

        currentImage = right;

        setBackground(Color.black);
        setPreferredSize(new Dimension(WIDTH, HEIGHT));
        setFocusable(true);
    }

    public void paintComponent(Graphics page)
    {
        super.paintComponent(page);
        currentImage.paintIcon(this, page, x, y);
    }
}
```

// continued...

//continuing...

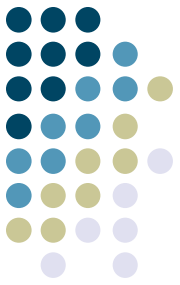
```
private class DirectionListener implements KeyListener
{
    public void keyPressed(KeyEvent event)
    {
        switch (event.getKeyCode())
        {
            case KeyEvent.VK_UP:
                currentImage = up;
                y -= JUMP;
                break;
            case KeyEvent.VK_DOWN:
                currentImage = down;
                y += JUMP;
                break;
            case KeyEvent.VK_LEFT:
                currentImage = left;
                x -= JUMP;
                break;
            case KeyEvent.VK_RIGHT:
                currentImage = right;
                x += JUMP;
                break;
        }

        repaint();
    }

    public void keyTyped(KeyEvent event) {}
    public void keyReleased(KeyEvent event) {}
}
```

Direction program

Animation



Redraw scene repeatedly with slight movement

`Timer` class – In Swing, generates an event at regular intervals

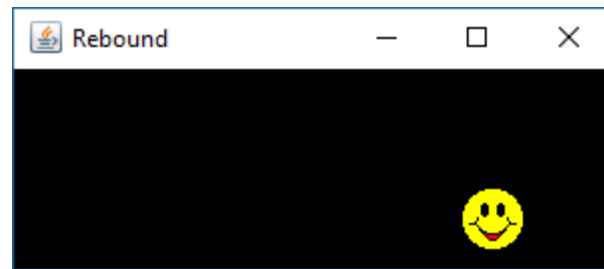
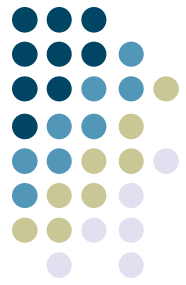
To do animation:

- Set up timer to generate events
- Provide right listener

Methods

```
Timer(int delay, ActionListener list) //delay is in msec  
void addActionListener(ActionListener list)  
boolean isRunning()  
void setDelay(int delay)  
void start()  
void stop()
```

Example Program



What determines velocity?

Rebound program

Example Program



```
public class ReboundPanel extends JPanel
{
    private final int WIDTH = 300, HEIGHT = 100;
    private final int DELAY = 20, IMAGE_SIZE = 35;

    private ImageIcon image;
    private Timer timer;
    private int x, y, moveX, moveY;

    public ReboundPanel()
    {
        timer = new Timer(DELAY, new ReboundListener());

        image = new ImageIcon ("happyFace.gif");

        x = 0;
        y = 40;
        moveX = moveY = 3;

        setPreferredSize (new Dimension(WIDTH, HEIGHT));
        setBackground (Color.black);
        timer.start();
    }

    public void paintComponent (Graphics page)
    {
        super.paintComponent (page);
        image.paintIcon (this, page, x, y);
    }
}
```

// continued..

//continuing..

```
private class ReboundListener implements ActionListener
{
    public void actionPerformed (ActionEvent event)
    {
        x += moveX;
        y += moveY;

        if (x <= 0 || x >= WIDTH-IMAGE_SIZE)
            moveX = moveX * -1;

        if (y <= 0 || y >= HEIGHT-IMAGE_SIZE)
            moveY = moveY * -1;

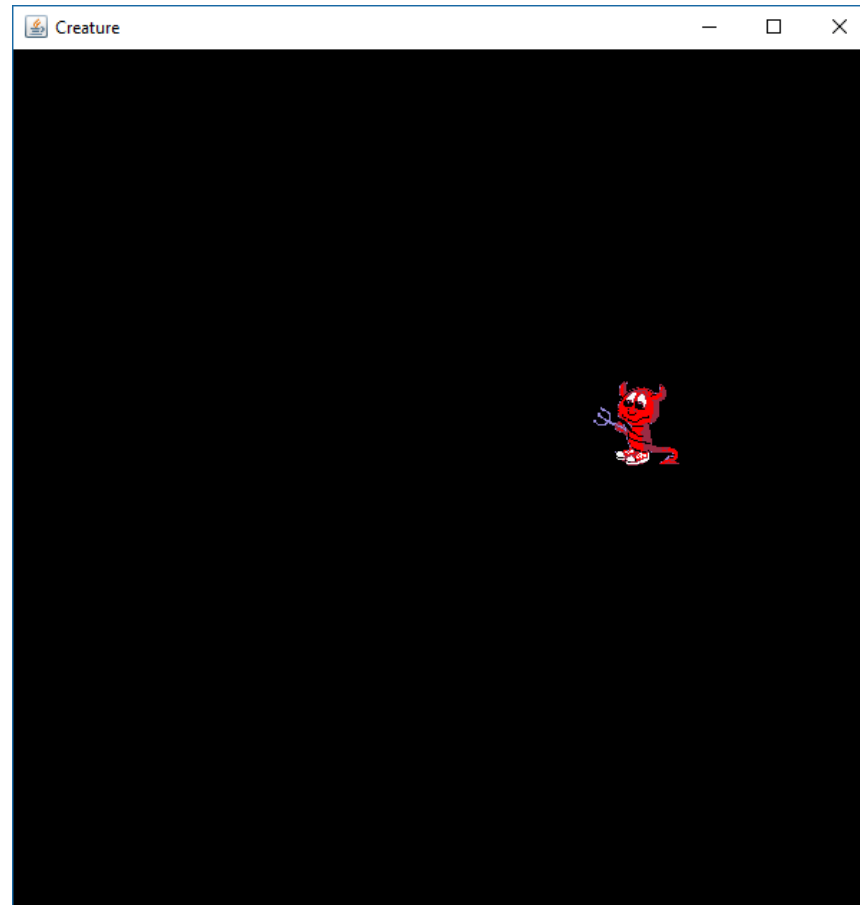
        repaint();
    }
}
```

Rebound program

Example Program



Game:
System moves the
image and you must
click in it by a preset
amount of time such
as one second



Creature program

Example Program



```
public class CreaturePanel extends JPanel
{
    private final int WIDTH=600, HEIGHT=600;
    private Timer timer;
    private ImageIcon image;
    private int iconWidth,iconHeight;
    private int x,y;
    private Random generator;
    private boolean chasing = false;
    private boolean inTime = false;
    private int tryCount = 0;
    private int hitCount = 0;
    private int hitX,hitY;
    private static final int DELAY =1000; // one second

    public CreaturePanel()
    {
        generator = new Random();

        ActionListener timerListener;
        timerListener = new TimerListener();
        timer = new Timer(DELAY,timerListener);
        MyMouseListener mouseListener = new MyMouseListener();
        addMouseListener (mouseListener);

        image = new ImageIcon("creature.gif");
        iconWidth = image.getIconWidth();
        iconHeight = image.getIconHeight();

        setBackground (Color.black);
        setPreferredSize(new Dimension(WIDTH,HEIGHT));
    }

    public void paintComponent (Graphics page)
    {
        super.paintComponent (page);

        image.paintIcon(this, page, x,y);
    }
}
// continued...
```

```
//continuing...
private class MyMouseListener implements MouseListener
{
    public void mousePressed (MouseEvent event)
    {
        Point point;

        if (chasing == false) {
            x = generator.nextInt(WIDTH-iconWidth) + 1;
            y = generator.nextInt(HEIGHT-iconHeight) + 1;
            timer.start();
            repaint();
            chasing = true;
            inTime = true;
        }
        else {
            tryCount++;
            if (inTime == true) {
                point = event.getPoint();
                hitX = point.x;
                hitY = point.y;
                if ((x <= hitX) && (hitX <= (x+iconWidth)) &&
                    (y <= hitY) && (hitY <= (y+iconHeight)))
                    hitCount++;
            }
            System.out.println(hitCount+" / "+tryCount);
            chasing = false;
            timer.stop();
        }
    }

    public void mouseClicked (MouseEvent event) {}
    public void mouseReleased (MouseEvent event) {}
    public void mouseEntered (MouseEvent event) {}
    public void mouseExited (MouseEvent event) {}
}

private class TimerListener implements ActionListener
{
    public void actionPerformed (ActionEvent event)
    {
        inTime = false;
    }
}
```

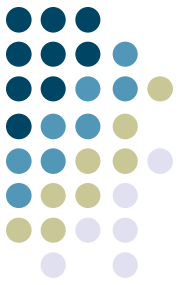
Creature program

Learning Objectives



- New UI components
 - File chooser, Text area, Color chooser, Slider, Combo box (menu)
- Tooltips and short-cuts
- Mouse events
 - Dynamic drawing
- Key events
- Timer events and animation

HW



- Something more involved with Swing GUIs

Next Time

- iOS app development for Macs
 - Ramik

