

**topics:**

- Intro to threads in Java
  - basic concepts
  - creating and using threads
  - GUI examples with threads

## multithreading

- programs can do several things at once - multitasking:
  - run multiple programs
  - play music
  - allow users to interact
- multithreaded programs extend the idea of multitasking by taking it one level lower:
  - individual programs carrying out several tasks in parallel.
  - if there aren't enough computational resources, programs will only appear to do multiple tasks at the same time.
- each task is usually called a *thread* – which is short for thread of control
- programs that can run more than one thread at once is said to be *multithreading*
- some other terms for multithreading are parallelism and concurrency

## multithreading (2)

- what is the difference between multiple processes and multiple threads?
  - processes have their own data
  - threads share the same data
- shared variables make communication between threads more efficient and easier to program than interprocess communication
- threads are more lightweight than processes
- computationally intensive processes may block user input in GUI applications, therefore good GUI applications contain different threads for GUI events and intensive or continuous computations such as animations.

## thread example: Blinking canvas

```
...  
  
public class BlinkDemo extends JFrame implements Runnable {  
    public static void main(String[] args) {  
        BlinkDemo demo = new BlinkDemo();  
        demo.setTitle("Blink Demo");  
        demo.setSize(500,400);  
        demo.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        demo.setVisible(true);  
  
        Thread t = new Thread(demo, "BlinkDemo");  
        t.start();  
    }  
  
    public BlinkDemo() {  
        canvas = new BlinkCanvas();  
        getContentPane().add(canvas);  
    }  
  
    public void run() {  
        while(true) {
```

```

        canvas.repaint();
        try {
            Thread.sleep(1000);
        }
        catch(InterruptedException e) {}
    }
}

private BlinkCanvas canvas;
}

class BlinkCanvas extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        Random r = new Random();
        int n = r.nextInt(3);

        Color c = Color.BLACK;
        switch(n){
            case 0:
                c = Color.red;
                break;
            case 1:

```

```

        c = Color.blue;
        break;
    case 2:
        c = Color.green;
        break;
    case 3:
        c = Color.black;
    }
    g2.setPaint(c);
    Rectangle2D rect = new Rectangle2D.Float(100,100,300,100);
    g2.fill(rect);
}
}

```

## Thread.sleep()

- class java.lang.Thread
- redraw the same image with two different colors (background and display color)
- class Thread provides 'delaying' mechanism
  - program waits by calling a library method Thread.sleep(), which has a parameter specifying how long the program waits
  - parameter passed to sleep() is the time in milliseconds that the program wants to sleep
  - exception-handling is required
- blinking rate and pattern can be controlled via the sleep intervals

## creating threads: Runnable interface

- although you will see code that does this, DO NOT INHERIT from Thread
- a simple procedure for running a task in a separate thread:
  - create a new class that implements the Runnable interface, and place the code for the task into the run method

```

class MyRunnable implements Runnable {
    public void run(){
        // task code goes here
    }
}

```

## Thread creation and running

- construct an object of your class that implements Runnable

```
Runnable r = new MyRunnable();
```

- construct a Thread object from the Runnable

```
Thread t = new Thread(r);
```

- start the thread

```
t.start();
```

## thread example: Controlled blinking

```
public class BlinkingLight implements ActionListener {  
  
    JFrame frame;  
    JButton startStop ;  
  
    public BlinkCanvas blinkCanvas ;  
  
    public BlinkingLight() {  
        frame = new JFrame();  
        startStop = new JButton("Start");  
        frame.getContentPane().add(startStop, BorderLayout.NORTH);  
        startStop.addActionListener(this);  
        blinkCanvas = new BlinkCanvas();  
        frame.getContentPane().add(blinkCanvas, BorderLayout.CENTER);  
        frame.setSize(500,400);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setVisible(true);  
    }  
  
    public static void main(String[] args) {  
        BlinkingLight bl = new BlinkingLight();  
    }  
}
```

```
Runnable blinkRunnable = new BlinkRunnable(bl.blinkCanvas);  
Thread t = new Thread(blinkRunnable);  
t.start();  
}  
  
public void actionPerformed(ActionEvent e){  
    blinkCanvas.setBlinking(!blinkCanvas.isBlinking());  
    if (blinkCanvas.isBlinking())  
        startStop.setLabel("Stop");  
    else  
        startStop.setLabel("Start");  
}  
}  
  
class BlinkRunnable implements Runnable {  
    public BlinkRunnable(BlinkCanvas blinkCanvas) {  
        this.blinkCanvas = blinkCanvas;  
    }  
  
    public void run() {  
        while(true) {  
            if(blinkCanvas.isBlinking()) {  
                blinkCanvas.toggleColor();  
            }  
        }  
    }  
}
```

```
    }  
    try {  
        Thread.sleep(1000);  
    } catch (InterruptedException e) {}  
    }  
}  
  
BlinkCanvas blinkCanvas;  
}  
  
class BlinkCanvas extends JPanel {  
    public BlinkCanvas(){  
        setSize(500,350);  
    }  
  
    public void paintComponent(Graphics g) {  
        g.setColor(theColor);  
        int rect_width = 50;  
        int rect_height = 50;  
        g.fillOval((getWidth()-rect_width)/2,  
                  (getHeight()-rect_height)/2,  
                  rect_width, rect_height);  
    }  
}
```

```
public boolean isBlinking() { return blinking; }

public void setBlinking(boolean blinking) {
    this.blinking = blinking;
}

public void toggleColor() {
    theColor = (theColor == Color.ORANGE) ? getBackground() : Color.ORANGE;
    repaint();
}

private Color theColor = Color.ORANGE;
private boolean blinking = false;
}
```