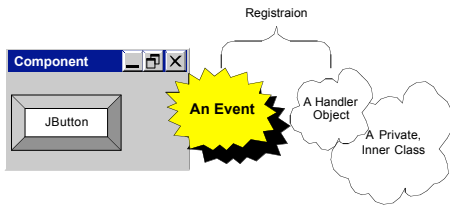


Handlers in General



A Handler Object

- Instantiate the Private, Inner Class

```
SomeHandler myHandler = new SomeHandler( );
```

- myHandler is an object
- can be registered to a component event

A Component

- JButton relies on user action
- an "event producer"

```
JButton helloButton = new JButton( "Hello");
```

Registration

- Tying a handler object to an “event producer”

```
helloButton.addActionListener( myHandler );
```

A Timer Event

- Imagine a clock “ticking” – at every tick, the computer generates an event.
- Computers have many internal clocks.
- Some with nano-second timing.
 - 10^{-9}
 - A nanosecond is to a second as a second is to 30 years
- The standard measure of computer time is the “millisecond” - 10^{-3}

Time

- second
- millisecond - $1/1000$ - 10^{-3}
- microsecond - $1/1,000,000$ - 10^{-6}
- nanosecond - $1/1,000,000,000$ - 10^{-9}

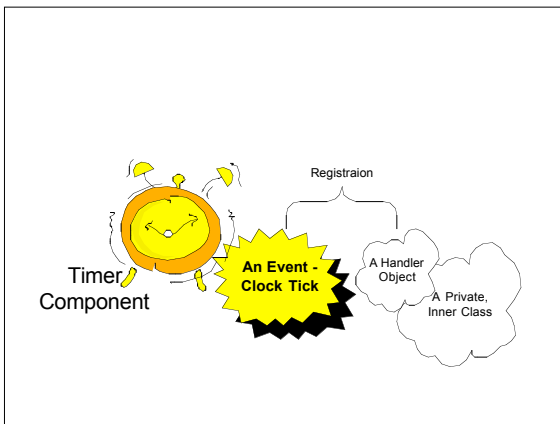
The Timer Component

```
SomeHandler myHandler = new SomeHandler( );
```

```
Timer tickTimer = new Timer(1000, myHandler);
```

How many milliseconds between events

what code runs every tick



Animation Coding Example

Main Class

```
public class Bouncer extends JFrame
    implements ActionListener
// note that the main class IS the handler class
// any registration will use "this"
{

// will contain an actionPerformed method
```

Main Class continued

```
{
private Timer timer; // timer must be known everywhere in the
// class
int ballX=0;
int ballY=0;
int diameter = 50;
static int width = 400;
static int height = 400;
int counter = 0;

// NO CONSTRUCTOR (everything must be in main method)
```

paint method

```
public void paint (Graphics g)
{
// background
g.setColor( Color.white );
g.fillRect( 0,0, width, height);
// ball
g.setColor( Color.red );
g.fillOval( ballX, ballY, diameter, diameter );
}
```

actionPerformed – every time the timer ticks

```
public void actionPerformed(ActionEvent e)
{
    ballY++;
    ballX++;
    repaint(); // like calling the paint method
    if ( counter++ == 400 ) // inc. and test together
    {
        timer.stop();
    }
}
```

timer gets a method all its own

```
public void initTimer()
{
    // note that handler is contained in this
    // class
    timer = new Timer( 5, this );
    timer.start();
}
```

main does what constructor usually does

```
public static void main(String[] args)
{
    Bouncer ball = new Bouncer();
    ball.setSize( width, height );
    ball.setVisible( true );
    ball.initTimer();
} // end main method

} // end main class
```

add a bounce

```
public void actionPerformed(ActionEvent e)
{
    ballY++;
    ballX++;
    repaint( );
    if(counter++ == 400)
    {
        timer.stop();
    }
    if(counter > 200)
    {
        ballY = ballY - 2 ; // net effect -1
    }
}
```
