# Lecture 26

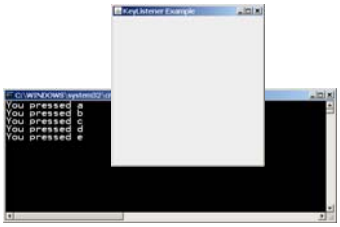
Keyboard Events

***Section 17.7***
Keyboard Events

# Introduction

- KeyEvents are generated when any key on the keyboard is pressed.

Let's write a program which simply prints out the key that was pressed




```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyJPanel extends JPanel {

	public MyJPanel() {


	}









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

	}
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyJPanel extends JPanel implements KeyListener {

	public MyJPanel() {


	}









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

	}
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyJPanel extends JPanel implements KeyListener {

	public MyJPanel() {


	}

	public void keyPressed(KeyEvent e) {

	}

	public void keyReleased(KeyEvent e) {}
	public void keyTyped(KeyEvent e) {}

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

	}
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyJPanel extends JPanel implements KeyListener {

	public MyJPanel() {

		addKeyListener(this);
	}

	public void keyPressed(KeyEvent e) {

	}

	public void keyReleased(KeyEvent e) {}
	public void keyTyped(KeyEvent e) {}

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

	}
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyJPanel extends JPanel implements KeyListener {

	public MyJPanel() {

		addKeyListener(this);
	}

	public void keyPressed(KeyEvent e) {
		System.out.println("You pressed " + e.getKeyChar());
	}

	public void keyReleased(KeyEvent e) {}
	public void keyTyped(KeyEvent e) {}

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

	}
}
```


```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyJPanel extends JPanel implements KeyListener {

	public MyJPanel() {

		addKeyListener(this);
	}

	public void keyPressed(KeyEvent e) {
		System.out.println("You pressed " + e.getKeyChar());
	}

	public void keyReleased(KeyEvent e) {}
	public void keyTyped(KeyEvent e) {}

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

	}
}
```


# Which key?

- To find out which key was pressed, there are two methods we can call on the KeyEvent parameter, e:

| *Method* | *Return type* | *Description* |
|---|---|---|
| **e.getKeyChar()** | **char** | Returns the character associated with the key |
| **e.getKeyCode()** | **int** | Returns the integer code associated with the key |

# getKeyChar() vs getKeyCode()

| **e.getKeyChar()** | **e.getKeyCode()** |
|---|---|
| Just compare with character literals:<br><br>**'a', 'b', 'c', ' ', ....** | Compare with public static constants defined in the KeyEvent class<br><br>**KeyEvent.VK_UP**<br>**KeyEvent.VK_LEFT**<br>**....** |
| EXAMPLE:<br>if (e.getKeyChar() == 'q') | EXAMPLE:<br>if (e.getKeyCode() == KeyEvent.VK_UP) |

# Focus

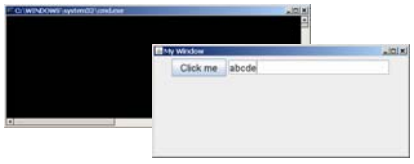
- At any time, only one component has "keyboard focus"
- If a component like a JTextField has keyboard focus, then any key presses will be handled by that component
- If we want out program to respond to key presses, then the JPanel must have keyboard focus


- Our JPanel class will initially have keyboard focus by default

# Focus

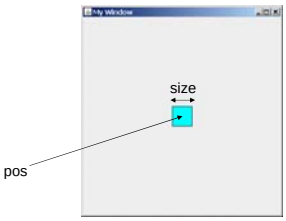
If the JPanel loses focus for some reason, we can request it again by calling the method:

```
requestFocusInWindow();
```



# Example: navigation

- Using the keyboard can be a useful way to navigate some object around the screen.




```
public class MyJPanel extends JPanel implements KeyListener {

	private Point pos;
	private int size;

	public MyJPanel() {

		addKeyListener(this);
	}

	public void keyPressed(KeyEvent e) {

	}

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


	}

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

Initialise the instance variables

Update the position of the square

Draw the square

```
public MyJPanel() {

	pos = new Point(100,80);
	size = 20;

	addKeyListener(this);
}
```

```
public void keyPressed(KeyEvent e) {

	if (e.getKeyCode() == KeyEvent.VK_UP)
		pos.y -= size;
	if (e.getKeyCode() == KeyEvent.VK_DOWN)
		pos.y += size;
	if (e.getKeyCode() == KeyEvent.VK_LEFT)
		pos.x -= size;
	if (e.getKeyCode() == KeyEvent.VK_RIGHT)
		pos.x += size;

	repaint();
}
```

```
  public void paintComponent(Graphics g){
	super.paintComponent(g);

	g.setColor(Color.cyan);
	g.fillRect(pos.x - size/2, pos.y - size/2, size, size);
	g.setColor(Color.black);
	g.drawRect(pos.x - size/2, pos.y - size/2, size, size);

}
```

# Handling multiple event types

- We can list multiple interfaces when we define the JPanel class:

```
public class MyJPanel extends JPanel
                implements MouseListener, ActionListener {
```