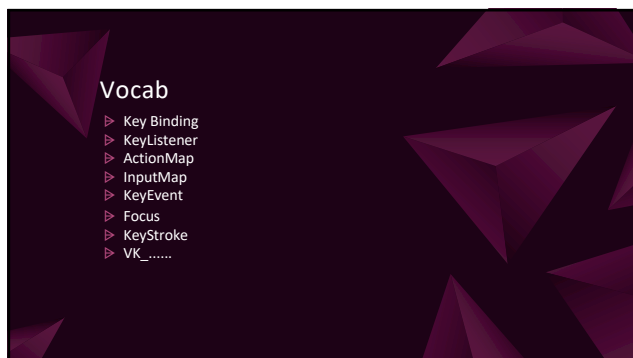
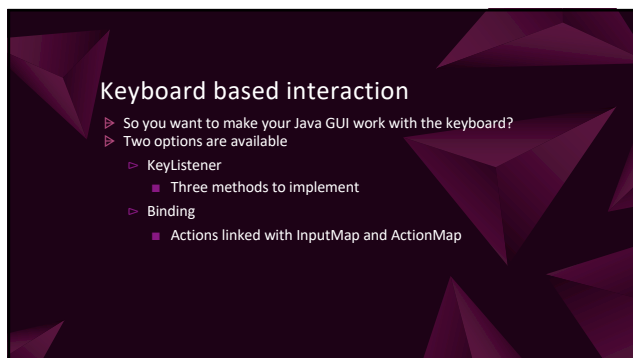




1



2



3

Listener

- ▷ Just like a `MouseListener` there are three methods so no lambdas
- ▷ Requires focus (not hard to do)
- ▷ All have a `KeyEvent` as a parameter
 - ▷ `keyPressed`
 - Finger on the key
 - ▷ `keyReleased`
 - Finger off the key
 - ▷ `keyTyped`
 - Only good for typing AKA do not use

4

Binding

- ▷ Need private classes that extend `AbstractAction`
 - ▷ Required method: `actionPerformed`
- ▷ Does not require focus
- ▷ First add the key to the `InputMap`
- ▷ Then add the action to the `ActionMap`

5

Basic Keys

- ▷ Each key on the keyboard can be accessed by the constant in the `KeyEvent` with a `VK_` prefix

6

Special Keys

- Alt, Shift, CTRL, and the meta key
- Function keys live here too (AKA the Fn#)
- Linked to binding either via name or with InputMask
- Allows for more complex keyboard interaction

7

KeyListener Example

```
this.addKeyListener(new KeyListener()
{
    public void keyTyped(KeyEvent e)
    {}
    public void keyPressed(KeyEvent e)
    {
        keys.add(e.getKeyCode());
        movement();
    }
    public void keyReleased(KeyEvent e)
    {
        keys.remove(e.getKeyCode());
    }
});
```

```
private void movement()
{
    for (Integer key : keys)
    {
        if (key == KeyEvent.VK_W || key == KeyEvent.VK_UP)
        {
            movementUp();
        }
        if (key == KeyEvent.VK_A || key == KeyEvent.VK_LEFT)
        {
            movementLeft();
        }
        if (key == KeyEvent.VK_S || key == KeyEvent.VK_DOWN)
        {
            movementDown();
        }
        if (key == KeyEvent.VK_D || key == KeyEvent.VK_RIGHT)
        {
            movementRight();
        }
    }
}
```

8

Bindings Example

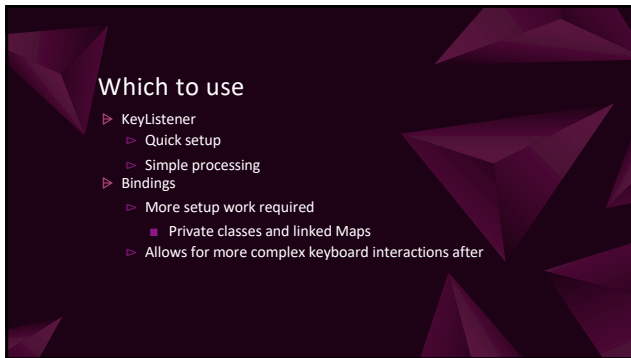
```
private class LeftAction extends AbstractAction
{
    public void actionPerformed(ActionEvent e)
    {
        movementLeft();
    }
}

private class DownAction extends AbstractAction
{
    public void actionPerformed(ActionEvent e)
    {
        movementDown();
    }
}

private class UpAction extends AbstractAction
{
    public void actionPerformed(ActionEvent e)
    {
        movementUp();
    }
}

//key bindings
this.getActionMap().put(KeyStroke.getKeyStroke("W"), new LeftAction());
this.getActionMap().put(KeyStroke.getKeyStroke("A"), new LeftAction());
this.getActionMap().put(KeyStroke.getKeyStroke("S"), new DownAction());
this.getActionMap().put(KeyStroke.getKeyStroke("D"), new DownAction());
this.getActionMap().put(KeyStroke.getKeyStroke("W"), new UpAction());
this.getActionMap().put(KeyStroke.getKeyStroke("A"), new UpAction());
this.getActionMap().put(KeyStroke.getKeyStroke("S"), new DownAction());
this.getActionMap().put(KeyStroke.getKeyStroke("D"), new DownAction());
```

9



Which to use

- ▷ KeyListener
 - ▷ Quick setup
 - ▷ Simple processing
- ▷ Bindings
 - ▷ More setup work required
 - Private classes and linked Maps
 - ▷ Allows for more complex keyboard interactions after

10
