

Implementing User Interfaces — Design Options

Note Title

① Ways to implement listener interfaces

- a. Direct implementation
- b. Extending adapters

② Relationship to the view class

- a. view listens to itself
- b. completely separate class
- c. inner class

③ Creating models that fire events

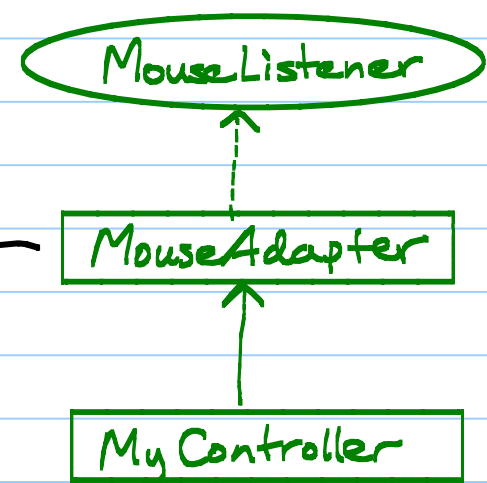
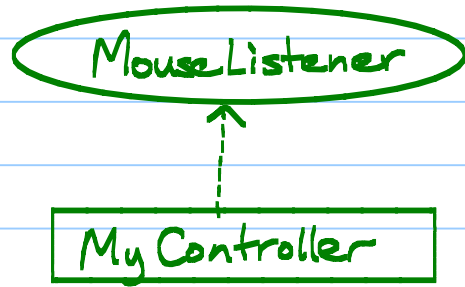
④ Handling exceptions

Ways to implement listener interfaces

Direct implementation

vs.

Extending adapters



```
mousePressed(--){ }  
mouseReleased(--){ }  
mouseClicked(--){ }  
;
```

```
override:  
mouseClicked(--){  
    interesting thing;  
}
```

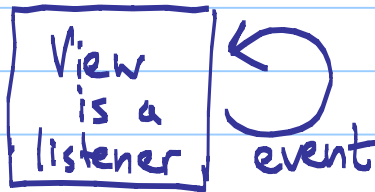
Adapters:

adv: only impl. methods of interest }

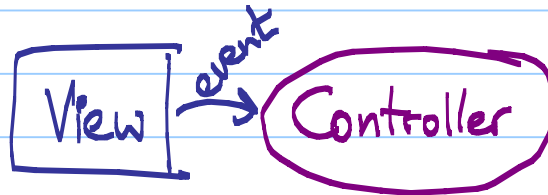
disadv: Java allows only single inheritance (one parent)

Relationship to the view class

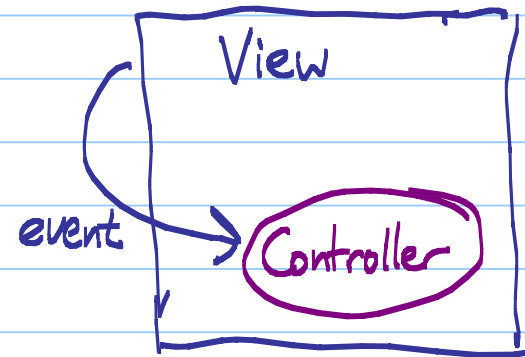
view listens to itself



completely separate class



inner class



Adv.	simple - put event handlers in the view	controller has a life of its own (inst. vars & methods) observer	behavior is all in one place
Disadv.	tightly coupled (poor reparation)	can be hard to "follow" program behavior spread over many classes	tightly coupled (poor reparation)

Named inner classes vs. anonymous classes

member / local

```
class Foo extends JPanel {
```

This is local like a method in a class

```
class MyListener extends MouseAdapter {  
    public void mouseClicked (MouseEvent me) {  
        ...  
    }  
}
```

```
// class name: Foo.MyListener
```

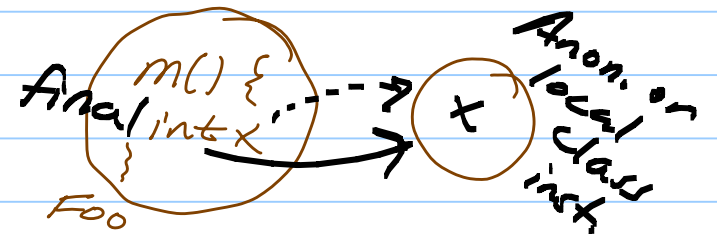
```
public void createListener() {  
    addMouseListener(new MouseAdapter() {  
        public void mouseClicked (MouseEvent me) {  
            ...  
        }  
    });  
}
```

compiler automatically passes a reference to the containing object to the inst. of the inner class

new MyListener()

Summary of Class Relationships to Class C

- Top-level class — no particular relationship
respect abstraction barrier
- Static nested class — like a top level class —
can't see instance vars & methods
— can create from outside: `new Foo.Bar()`
- Member class — has implicit ref. to containing object
- Local class — like member, but also can access
local variable if they are final
- Anonymous class



Creating a model that fires events

Example: DirectoryWatcher

