

COMP 401

MODEL-VIEW-CONTROLLER (MVC)

Instructor: Prasun Dewan



PREREQUISITES

- Interfaces
- Main Console Input
- Inheritance



GENERAL PROBLEM

- How to break up our program into multiple classes?



SEPARATION OF CONCERNS

History Semantics

History Display 1

History Semantics

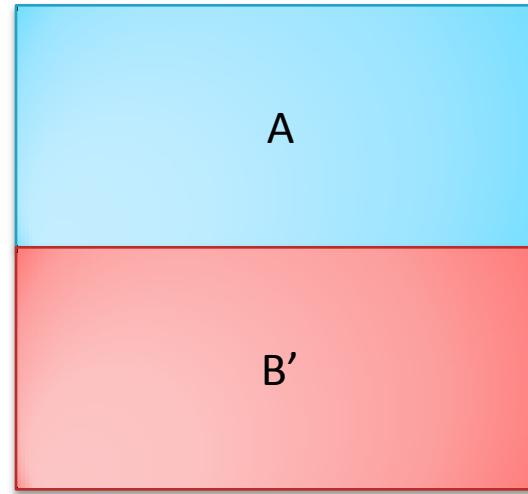
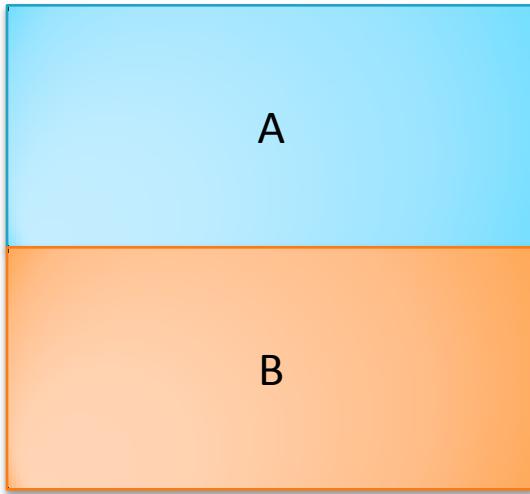
History Display 2

Can change display without changing other aspects of history

Display and semantics should go in different classes



SEPARATION OF CONCERNS



if a part A of a class can be changed without changing some other part B of the class, then refactor and put A and B in different classes

PATTERNS

- Recurring theme
- Bean, Vector pattern
 - Conventions for readability
- Loop patterns
 - Event-controlled
 - Counter controlled
- Design patterns
 - Helps identify the kind of classes our program should have

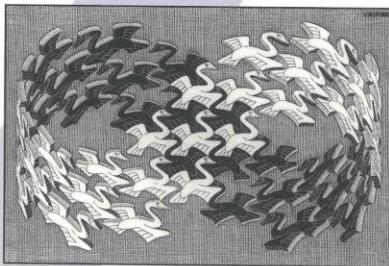


DESIGN PATTERN

Design Patterns

Elements of Reusable
Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



© 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch



- Reusable program decomposition pattern
- Not specific class or interface, infinite family of classes/interfaces implement this pattern.
- Usually involves multiple objects
- Language-independent
- Include architecture and frameworks
- Inspired by Architectural Pattern (Christopher Plummer)





SIMILAR

2nd story
porch
supported by
1st floor porch
columns

DIFFERENT

No outside stairs to
2nd story
Flat (not bay) window
Wide plank siding
Screened-in porch



MVC MOTIVATION

History Semantics

History Display 1

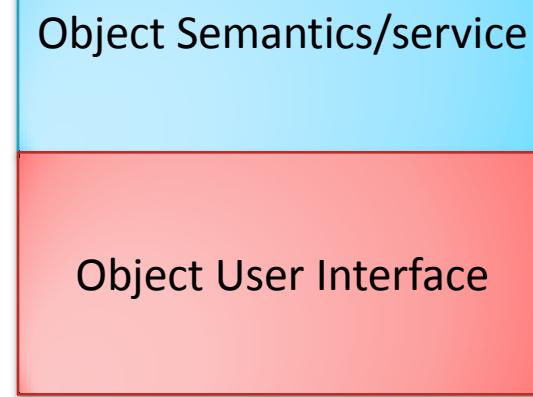
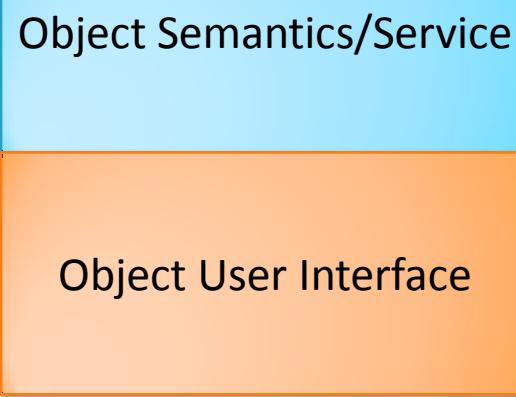
History Semantics

History Display 2

Can change display without changing other aspects of history

Display and semantics should go in different classes

MVC MOTIVATION



QUESTIONS

- How to reuse code among different interactive applications offering the same service?
- How to simultaneously create multiple user interfaces for same service?
 - Normal vs. Slide sorter
 - Shortcuts vs. menus vs. buttons

Slides Outline

IMPLEMENTING INTERACTIVE APPLICATIONS

IMPLEMENTING INTERACTIVE APPLICATIONS

Topics

- How to reuse code among different user interfaces?
- How to simultaneously create multiple user interfaces for same service?
- Normal
- Slide Sorter
- Shortcuts
- Menus
- Buttons

- How to reuse code among different user interfaces?
- How to simultaneously create multiple user interfaces for same service?
 - Normal
 - Slide sorter
 - Shortcuts
 - Menus
 - Buttons
- How to notify observers?

TOPICS

- How to reuse code among different user interfaces?
- How to simultaneously create multiple user interfaces for same service?
- Normal
- Slide Sorter
- Shortcuts
- Menus
- Buttons

00:09 7 02:14 8

COUNTER

- Can add arbitrary positive/negative value to an integer
- Different user interfaces

03:29 12 00:30 13

MONOLITHIC CONSOLE UI IMPLEMENTATION

```
public class Counter {  
    public void add(int amount) {  
        int counter = 0;  
        while (true) {  
            System.out.println("Counter " + counter);  
            if (amount > 0) counter++;  
            else if (amount < 0) counter--;  
            else break;  
        }  
    }  
}
```

02:09 17 08:15 18

MONOLITHIC MIXED UI IMPLEMENTATION

```
public class Counter {  
    public void add(int amount) {  
        int counter = 0;  
        while (true) {  
            System.out.println("Counter " + counter);  
            if (amount > 0) counter++;  
            else if (amount < 0) counter--;  
            else break;  
        }  
    }  
}
```

Data Aggregation

QUESTIONS

- How to simultaneously create multiple user interfaces for same service on different computers?
 - Facebook, email

The screenshot shows the Facebook desktop interface. At the top, there's a navigation bar with links for Home, Profile, Friends, and Inbox. Below the navigation is a search bar and a 'Log out' button. The main content area is divided into several sections:

- Recent Stories:** Shows updates from friends like Daniel, Jake, Kelvin, Oli, and Josh.
- Update status:** A button to post a status update.
- Requests:** A section for new friend requests, showing 1 farm town neighbour request.
- Suggestions:** A section for suggested pages like Bacon Butties, with 10 friends as fans.
- Events:** A section for upcoming events, including '1000 Miles To Go'.
- Pokes:** A section for pokes sent by friends like Sarah.
- Connect with friends:** A section for inviting friends to join Facebook and finding mutual friends.



QUESTIONS

- How to simultaneously create distributed user interfaces
 - multiple complete user interfaces for different users on different computers
 - Single user-interface on large computer controlled by multiple mobile devices

2008 NCAA Basketball Tournament Brackets

	A	B	C	D	E
1					
2					
3					
4					
5	First Round	Second Round		Regionals	
7	1 North Carolina				
9	16 Play-in				
11	Raleigh, NC				
12	8 Indiana				
13	9 Arkansas				

Google Docs

2008 NCAA Basketball Tournament Brackets Autosaved on Mar 15, 2008 8:35:50 AM PDT

File ▾ **Edit** **Sort** **Formulas** **Revisions**

Format **B** **I** **U** **More** **F** **T** **G** **Text** **Align** **Insert** **Delete**

A B C

	First Round	Second Round	Regionals
1	1 North Carolina		
2	16 Play-in		
3	Raleigh, NC		
4	8 Indiana		
5	9 Arizona		
6	12 Florida		
7	13 Michigan		
8	15 Wisconsin		
9	17 Ohio State		
10	4 Louisville		
11	5 Missouri		
12	6 Michigan State		
13	7 Kansas		
14	10 Kansas State		
15	11 Oklahoma		
16	14 Oklahoma State		
17	18 Butler		
18	21 Xavier		
19	22 Cincinnati		
20	23 Virginia Tech		
21	24 Wake Forest		
22	25 Temple		
23	26 Seton Hall		
24	27 Georgetown		
25	28 Boston College		
26	29 Villanova		
27	30 Marquette		
28	31 St. Louis		
29	32 Xavier		
30	33 Belmont		
31	34 Samford		
32	35 Loyola-Chicago		
33	36 Loyola Marymount		
34	37 Bucknell		
35	38 Saint Peter's		
36	39 Saint Louis		
37	40 Saint Joseph's		
38	41 Saint Mary's		
39	42 Saint Louis		
40	43 Saint Joseph's		
41	44 Saint Peter's		
42	45 Saint Louis		
43	46 Saint Joseph's		
44	47 Saint Peter's		
45	48 Saint Louis		
46	49 Saint Joseph's		
47	50 Saint Peter's		
48	51 Saint Louis		
49	52 Saint Joseph's		
50	53 Saint Peter's		
51	54 Saint Louis		
52	55 Saint Joseph's		
53	56 Saint Peter's		
54	57 Saint Louis		
55	58 Saint Joseph's		
56	59 Saint Peter's		
57	60 Saint Louis		
58	61 Saint Joseph's		
59	62 Saint Peter's		
60	63 Saint Louis		
61	64 Saint Joseph's		
62	65 Saint Peter's		
63	66 Saint Louis		
64	67 Saint Joseph's		
65	68 Saint Peter's		
66	69 Saint Louis		
67	70 Saint Joseph's		
68	71 Saint Peter's		
69	72 Saint Louis		
70	73 Saint Joseph's		
71	74 Saint Peter's		
72	75 Saint Louis		
73	76 Saint Joseph's		
74	77 Saint Peter's		
75	78 Saint Louis		
76	79 Saint Joseph's		
77	80 Saint Peter's		
78	81 Saint Louis		
79	82 Saint Joseph's		
80	83 Saint Peter's		
81	84 Saint Louis		
82	85 Saint Joseph's		
83	86 Saint Peter's		
84	87 Saint Louis		
85	88 Saint Joseph's		
86	89 Saint Peter's		
87	90 Saint Louis		
88	91 Saint Joseph's		
89	92 Saint Peter's		
90	93 Saint Louis		
91	94 Saint Joseph's		
92	95 Saint Peter's		
93	96 Saint Louis		
94	97 Saint Joseph's		
95	98 Saint Peter's		
96	99 Saint Louis		
97	100 Saint Joseph's		
98	101 Saint Peter's		
99	102 Saint Louis		
100	103 Saint Joseph's		
101	104 Saint Peter's		
102	105 Saint Louis		
103	106 Saint Joseph's		
104	107 Saint Peter's		
105	108 Saint Louis		
106	109 Saint Joseph's		
107	110 Saint Peter's		
108	111 Saint Louis		
109	112 Saint Joseph's		
110	113 Saint Peter's		
111	114 Saint Louis		
112	115 Saint Joseph's		
113	116 Saint Peter's		
114	117 Saint Louis		
115	118 Saint Joseph's		
116	119 Saint Peter's		
117	120 Saint Louis		
118	121 Saint Joseph's		
119	122 Saint Peter's		
120	123 Saint Louis		
121	124 Saint Joseph's		
122	125 Saint Peter's		
123	126 Saint Louis		
124	127 Saint Joseph's		
125	128 Saint Peter's		
126	129 Saint Louis		
127	130 Saint Joseph's		
128	131 Saint Peter's		
129	132 Saint Louis		
130	133 Saint Joseph's		
131	134 Saint Peter's		
132	135 Saint Louis		
133	136 Saint Joseph's		
134	137 Saint Peter's		
135	138 Saint Louis		
136	139 Saint Joseph's		
137	140 Saint Peter's		
138	141 Saint Louis		
139	142 Saint Joseph's		
140	143 Saint Peter's		
141	144 Saint Louis		
142	145 Saint Joseph's		
143	146 Saint Peter's		
144	147 Saint Louis		
145	148 Saint Joseph's		
146	149 Saint Peter's		
147	150 Saint Louis		
148	151 Saint Joseph's		
149	152 Saint Peter's		
150	153 Saint Louis		
151	154 Saint Joseph's		
152	155 Saint Peter's		
153	156 Saint Louis		
154	157 Saint Joseph's		
155	158 Saint Peter's		
156	159 Saint Louis		
157	160 Saint Joseph's		
158	161 Saint Peter's		
159	162 Saint Louis		
160	163 Saint Joseph's		
161	164 Saint Peter's		
162	165 Saint Louis		
163	166 Saint Joseph's		
164	167 Saint Peter's		
165	168 Saint Louis		
166	169 Saint Joseph's		
167	170 Saint Peter's		
168	171 Saint Louis		
169	172 Saint Joseph's		
170	173 Saint Peter's		
171	174 Saint Louis		
172	175 Saint Joseph's		
173	176 Saint Peter's		
174	177 Saint Louis		
175	178 Saint Joseph's		
176	179 Saint Peter's		
177	180 Saint Louis		
178	181 Saint Joseph's		
179	182 Saint Peter's		
180	183 Saint Louis		
181	184 Saint Joseph's		
182	185 Saint Peter's		
183	186 Saint Louis		
184	187 Saint Joseph's		
185	188 Saint Peter's		
186	189 Saint Louis		
187	190 Saint Joseph's		
188	191 Saint Peter's		
189	192 Saint Louis		
190	193 Saint Joseph's		
191	194 Saint Peter's		
192	195 Saint Louis		
193	196 Saint Joseph's		
194	197 Saint Peter's		
195	198 Saint Louis		
196	199 Saint Joseph's		
197	200 Saint Peter's		
198	201 Saint Louis		
199	202 Saint Joseph's		
200	203 Saint Peter's		
201	204 Saint Louis		
202	205 Saint Joseph's		
203	206 Saint Peter's		
204	207 Saint Louis		
205	208 Saint Joseph's		
206	209 Saint Peter's		
207	210 Saint Louis		
208	211 Saint Joseph's		
209	212 Saint Peter's		
210	213 Saint Louis		
211	214 Saint Joseph's		
212	215 Saint Peter's		
213	216 Saint Louis		
214	217 Saint Joseph's		
215	218 Saint Peter's		
216	219 Saint Louis		
217	220 Saint Joseph's		
218	221 Saint Peter's		
219	222 Saint Louis		
220	223 Saint Joseph's		
221	224 Saint Peter's		
222	225 Saint Louis		
223	226 Saint Joseph's		
224	227 Saint Peter's		
225	228 Saint Louis		
226	229 Saint Joseph's		
227	230 Saint Peter's		
228	231 Saint Louis		
229	232 Saint Joseph's		
230	233 Saint Peter's		
231	234 Saint Louis		
232	235 Saint Joseph's		
233	236 Saint Peter's		
234	237 Saint Louis		
235	238 Saint Joseph's		
236	239 Saint Peter's		
237	240 Saint Louis		
238	241 Saint Joseph's		
239	242 Saint Peter's		
240	243 Saint Louis		
241	244 Saint Joseph's		
242	245 Saint Peter's		
243	246 Saint Louis		
244	247 Saint Joseph's		
245	248 Saint Peter's		
246	249 Saint Louis		
247	250 Saint Joseph's		
248	251 Saint Peter's		
249	252 Saint Louis		
250	253 Saint Joseph's		
251	254 Saint Peter's		
252	255 Saint Louis		
253	256 Saint Joseph's		
254	257 Saint Peter's		
255	258 Saint Louis		
256	259 Saint Joseph's		
257	260 Saint Peter's		
258	261 Saint Louis		
259	262 Saint Joseph's		
260	263 Saint Peter's		
261	264 Saint Louis		
262	265 Saint Joseph's		
263	266 Saint Peter's		
264	267 Saint Louis		
265	268 Saint Joseph's		
266	269 Saint Peter's		
267	270 Saint Louis		
268	271 Saint Joseph's		
269	272 Saint Peter's		
270	273 Saint Louis		
271	274 Saint Joseph's		
272	275 Saint Peter's		
273	276 Saint Louis		
274	277 Saint Joseph's		
275	278 Saint Peter's		
276	279 Saint Louis		
277	280 Saint Joseph's		
278	281 Saint Peter's		
279	282 Saint Louis		
280	283 Saint Joseph's		
281	284 Saint Peter's		
282	285 Saint Louis		
283	286 Saint Joseph's		
284	287 Saint Peter's		
285	288 Saint Louis		
286	289 Saint Joseph's		
287	290 Saint Peter's		
288	291 Saint Louis		
289	292 Saint Joseph's		
290	293 Saint Peter's		
291	294 Saint Louis		
292	295 Saint Joseph's		
293	296 Saint Peter's		
294	297 Saint Louis		
295	298 Saint Joseph's		
296	299 Saint Peter's		
297	300 Saint Louis		
298	301 Saint Joseph's		
299	302 Saint Peter's		
300	303 Saint Louis		
301	304 Saint Joseph's		
302	305 Saint Peter's		
303	306 Saint Louis		
304	307 Saint Joseph's		
305	308 Saint Peter's		
306	309 Saint Louis		
307	310 Saint Joseph's		
308	311 Saint Peter's		
309	312 Saint Louis		
310	313 Saint Joseph's		
311	314 Saint Peter's		
312	315 Saint Louis		
313	316 Saint Joseph's		
314	317 Saint Peter's		
315	318 Saint Louis		
316	319 Saint Joseph's		
317	320 Saint Peter's		
318	321 Saint Louis		
319	322 Saint Joseph's		
320	323 Saint Peter's		
321	324 Saint Louis		
322	325 Saint Joseph's		
323	326 Saint Peter's		
324	327 Saint Louis		
325	328 Saint Joseph's		
326	329 Saint Peter's		
327	330 Saint Louis		
328	331 Saint Joseph's		
329	332 Saint Peter's		
330	333 Saint Louis		
331	334 Saint Joseph's		
332	335 Saint Peter's		
333	336 Saint Louis		
334	337 Saint Joseph's		
335	338 Saint Peter's		
336	339 Saint Louis		
337	340 Saint Joseph's		
338	341 Saint Peter's		
339	342 Saint Louis		
340	343 Saint Joseph's		
341	344 Saint Peter's		
342	345 Saint Louis		
343	346 Saint Joseph's		
344	347 Saint Peter's		
345	348 Saint Louis		
346	349 Saint Joseph's		
347	350 Saint Peter's		
348	351 Saint Louis		
349	352 Saint Joseph's		
350	353 Saint Peter's		
351	354 Saint Louis		
352	355 Saint Joseph's		
353	356 Saint Peter's		
354	357 Saint Louis		
355	358 Saint Joseph's		
356	359 Saint Peter's		
357	360 Saint Louis		
358	361 Saint Joseph's		
359	362 Saint Peter's		
360	363 Saint Louis		
361	364 Saint Joseph's		
362	365 Saint Peter's		
363	366 Saint Louis		
364	367 Saint Joseph's		
365	368 Saint Peter's		
366	369 Saint Louis		
367	370 Saint Joseph's		
368	371 Saint Peter's		
369	372 Saint Louis		
370	373 Saint Joseph's		
371	374 Saint Peter's		
372	375 Saint Louis		
373	376 Saint Joseph's		
374	377 Saint Peter's		
375	378 Saint Louis		
376	379 Saint Joseph's		
377	380 Saint Peter's		
378	381 Saint Louis		
379	382 Saint Joseph's		
380	383 Saint Peter's		
381	384 Saint Louis		
382	385 Saint Joseph's		
383	386 Saint Peter's		
384	387 Saint Louis		
385	388 Saint Joseph's		
386	389 Saint Peter's		
387	390 Saint Louis		
388	391 Saint Joseph's		
389	392 Saint Peter's		
390	393 Saint Louis		
391	394 Saint Joseph's		
3			

Slides **Outline** **X**

0 Implementing Direct Activity Selection

Implementation Options

Implementation Type	Implementation	Notes
Implementation A	Implementation A	Notes A
Implementation B	Implementation B	Notes B
Implementation C	Implementation C	Notes C

1 Implementing Direct Activity Selection

Implementation Options

Implementation Type	Implementation	Notes
Implementation A	Implementation A	Notes A
Implementation B	Implementation B	Notes B
Implementation C	Implementation C	Notes C

2 Topics

- How to reuse code with very different user interfaces?
- How to automatically generate code when you have a large number of similar objects?
- How to generate code from diagrams?

- How to reuse interfaces for slide sorting
 - How to notice



EXAMPLE: COUNTER

Can add
arbitrary
positive/negative
value to an
integer

Different user
interfaces

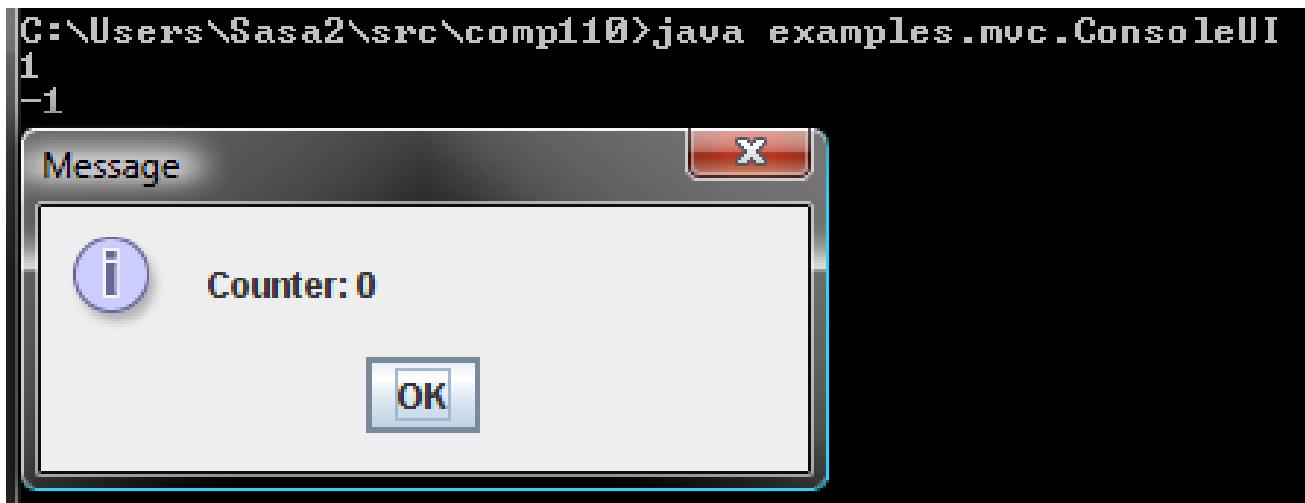
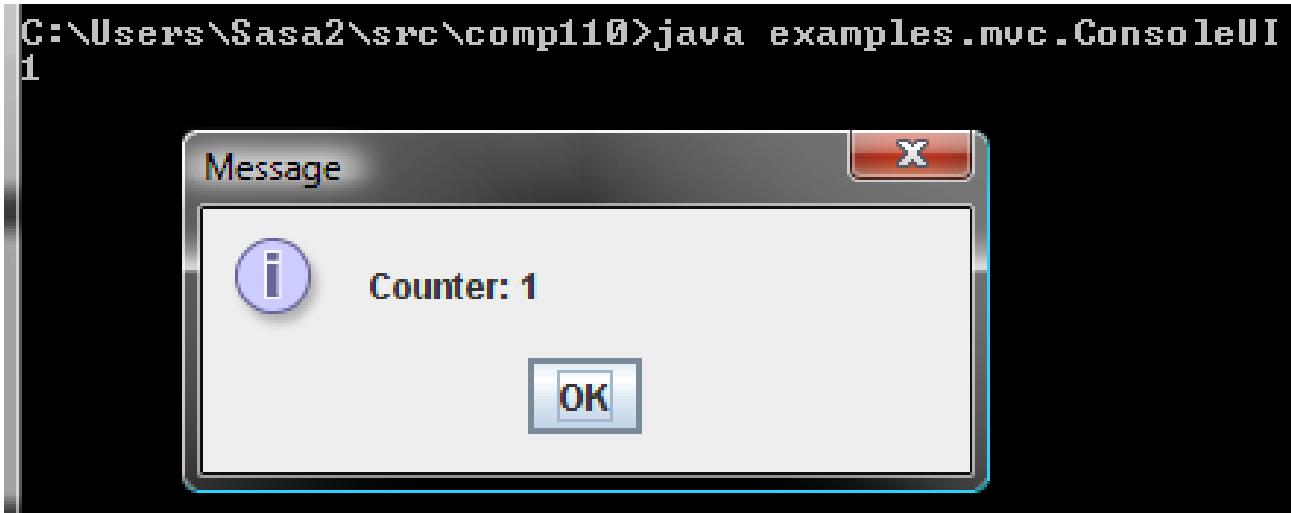


CONSOLE INPUT AND OUTPUT

```
C:\Users\Sasa2\src\comp10>java examples.mvc.ConsoleUI
Counter: 0
1
Counter: 1
-1
Counter: 0
5
Counter: 5
-
```



CONSOLE INPUT AND JOPTIONP INPUT



CONSOLE INPUT,OUTPUT AND JOptionPane OUTPUT



MONOLITHIC CONSOLE UI IMPLEMENTATION

```
public class MonolithicConsoleUI {  
    public static void main(String[] args) {  
        int counter = 0;  
        while (true) {  
            System.out.println("Counter: " + counter);  
            int nextInput = Console.readInt();  
            if (nextInput == 0) break;  
            counter += nextInput;  
        }  
    }  
}
```

```
C:\Users\Sasa2\src\comp110>java examples.mvc.ConsoleUI  
Counter: 0  
1  
Counter: 1  
-1  
Counter: 0  
5  
Counter: 5  
-
```

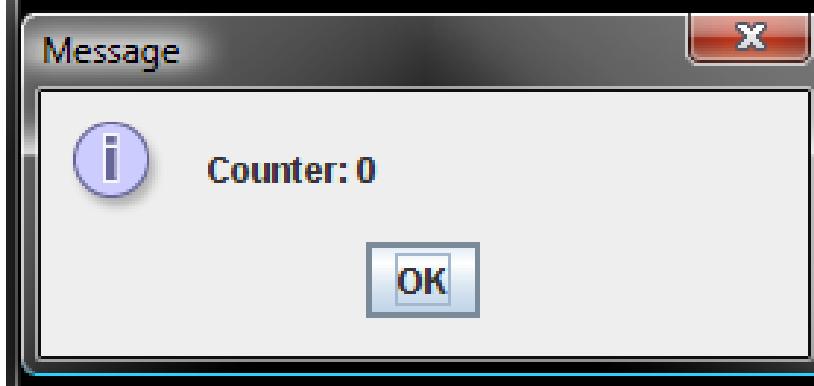


MONOLITHIC CONSOLE UI IMPLEMENTATION

```
public class MonolithicConsoleUI {  
    public static void main(String[] args) {  
        int counter = 0;  
        while (true) {  
            System.out.println("Counter: " + counter);  
            int nextInput = Console.readInt();  
            if (nextInput == 0) break;  
            counter += nextInput;  
        }  
    }  
}
```

C:\Users\Sasa2\src\comp110>java examples.mvc.ConsoleUI

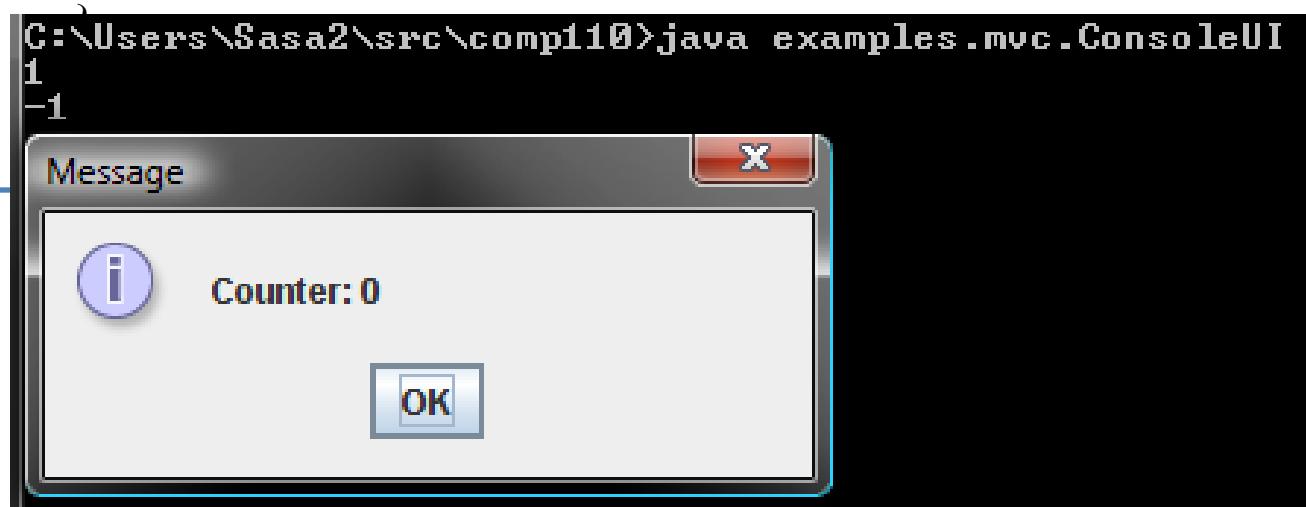
1
-1



MONOLITHIC MIXED UI IMPLEMENTATION

```
import javax.swing.JOptionPane;  
public class ConsoleUI {  
    public static void main(String[] args) {  
        int counter = 0;  
        while (true) {  
            JOptionPane.showMessageDialog(  
                null, "Counter: " + counter);  
            int nextInput = Console.readInt();  
            if (nextInput == 0) break;  
            counter += nextInput;  
        }  
    }  
}
```

Code duplication



MULTIPLE UIs?

```
import javax.swing.JOptionPane;
public class ConsoleUI {
    public static void main(String[] args) {
        int counter = 0;
        while (true) {
            JOptionPane.showMessageDialog(
                null, "Counter: " + counter);
            int nextInput = Console.readInt();
            if (nextInput == 0) break;
    } }
```



Cannot use both UIs simultaneously to update same counter



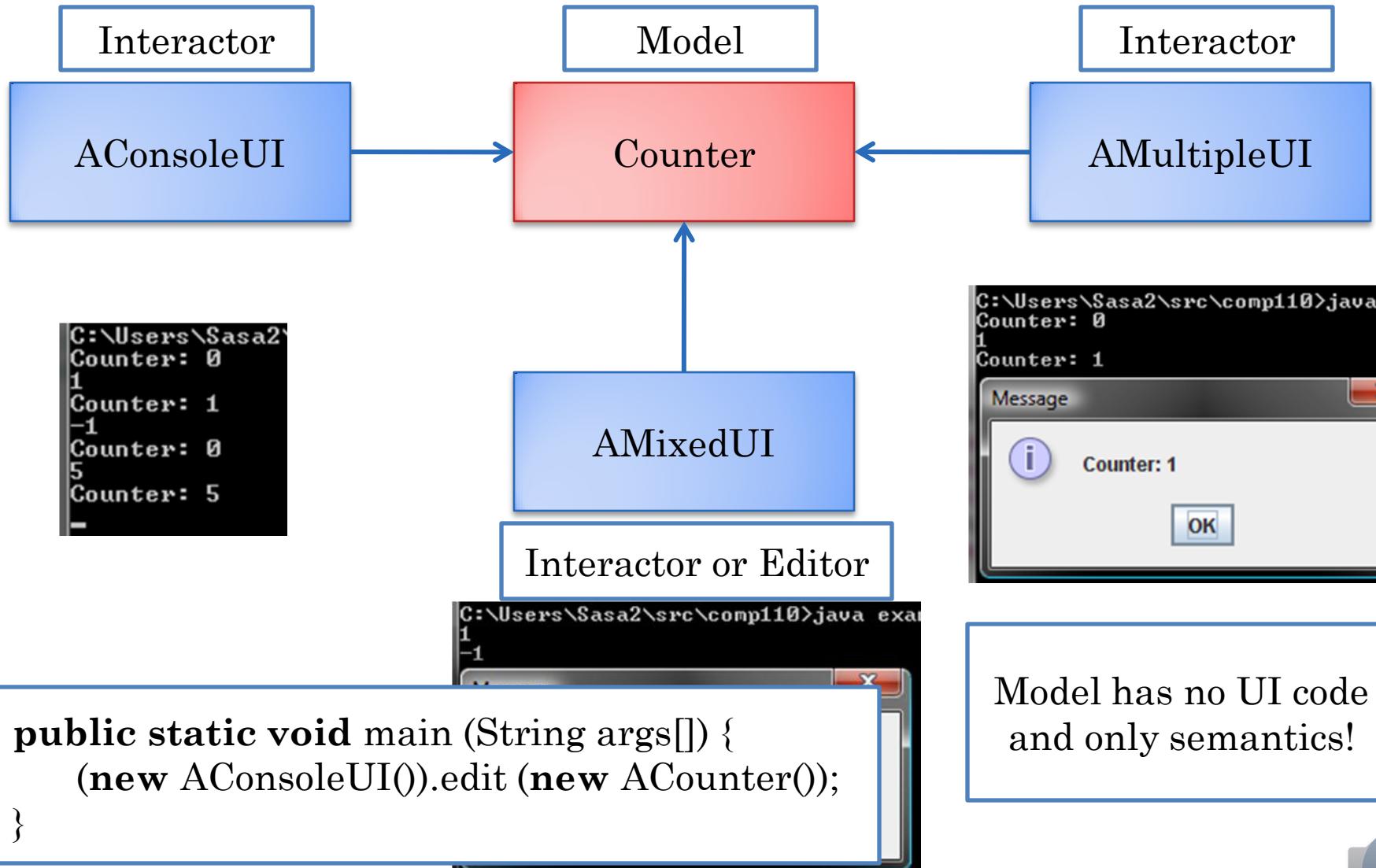
COUNTER MODEL

```
public class ACounter implements Counter {  
    int counter = 0;  
    public void add (int amount) {  
        counter += amount;  
    }  
    public int getValue() {  
        return counter;  
    }  
}
```

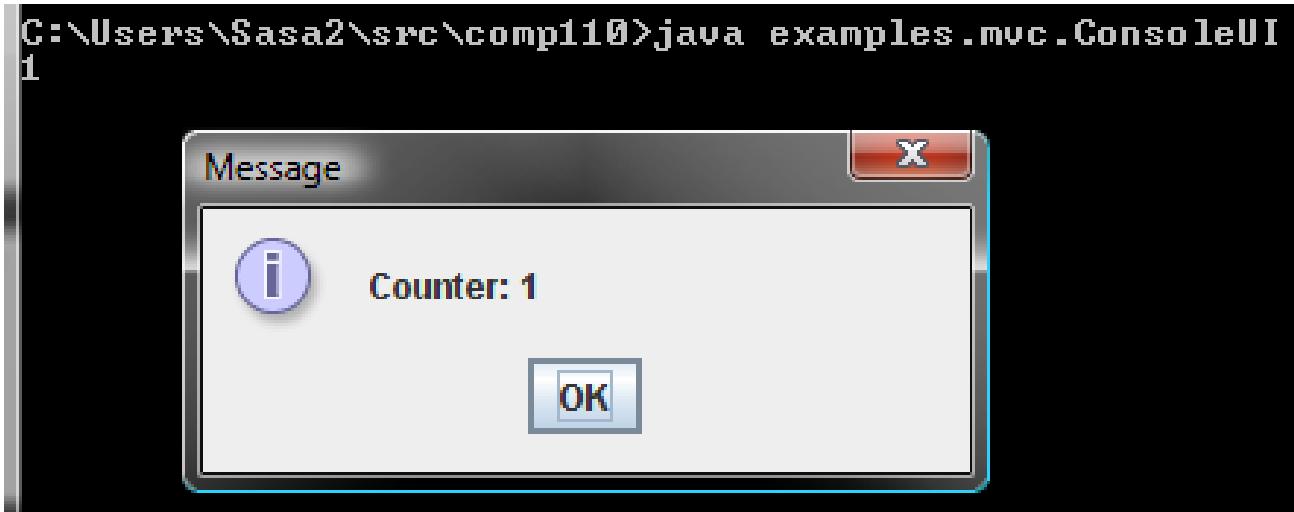
No input/output



MODEL/INTERACTOR(EDITOR) SEPARATION



MODEL?



COUNTER MODEL

```
public class ACounter implements Counter {  
    int counter = 0;  
    public void add (int amount) {  
        counter += amount;  
    }  
    public int getValue() {  
        return counter;  
    }  
}
```

No input/output



MONOLITHIC CONSOLE UI IMPLEMENTATION

```
public class ConsoleUI {  
    public static void main(String[] args) {  
        int counter = 0;  
        while (true) {  
            System.out.println("Counter: " + counter);  
            int nextInput = Console.readInt();  
            if (nextInput == 0) break;  
            counter += nextInput;  
        }  
    }  
}
```



CONSOLE INTERACTOR

```
public class AConsoleUIInteractor implements CounterInteractor {  
    public void edit(Counter counter) {  
        while (true) {  
            System.out.println("Counter: " + counter.getValue());  
            int nextInput = Console.readInt();  
            if (nextInput == 0) return;  
            counter.add(nextInput);  
        }  
    }  
}
```



MIXED INTERACTOR

```
public class AMixedUIInteractor implements CounterInteractor {  
    public void edit(Counter counter) {  
        while (true) {  
            JOptionPane.showMessageDialog(null  
                "Counter: " + counter.getValue());  
            int nextInput = Console.readInt();  
            if (nextInput == 0) break;  
            counter.add(nextInput);  
        }  
    }  
}
```

The diagram illustrates the decomposition of the provided Java code into four distinct functional components:

- Shared Model Code:** Represented by a blue box containing the line `counter.add(nextInput);`. A blue arrow points from this box to the original code.
- Output:** Represented by a blue box containing the lines `JOptionPane.showMessageDialog(null, "Counter: " + counter.getValue());` and `int nextInput = Console.readInt();`. A blue arrow points from this box to the original code.
- Input:** Represented by a blue box containing the line `if (nextInput == 0) break;`. A blue arrow points from this box to the original code.
- UI Implementation:** Represented by a blue box containing the line `public void edit(Counter counter) {`. A blue arrow points from this box to the original code.

I/O Code is
Duplicated

UI Implementation
is now monolithic



MULTIPLE UI INTERACTOR

```
public class AMultipleUI implements CounterInteractor {  
    public void edit(Counter counter) {  
        while (true) {  
            System.out.println("Counter: " + counter.getValue());  
            JOptionPane.showMessageDialog(null,  
                "Counter: " + counter.getValue());  
            int nextInput = Console.readInt();  
            if (nextInput == 0) break;  
            counter.add(nextInput);  
        }  
    }  
}
```

Shared Model Code

Output

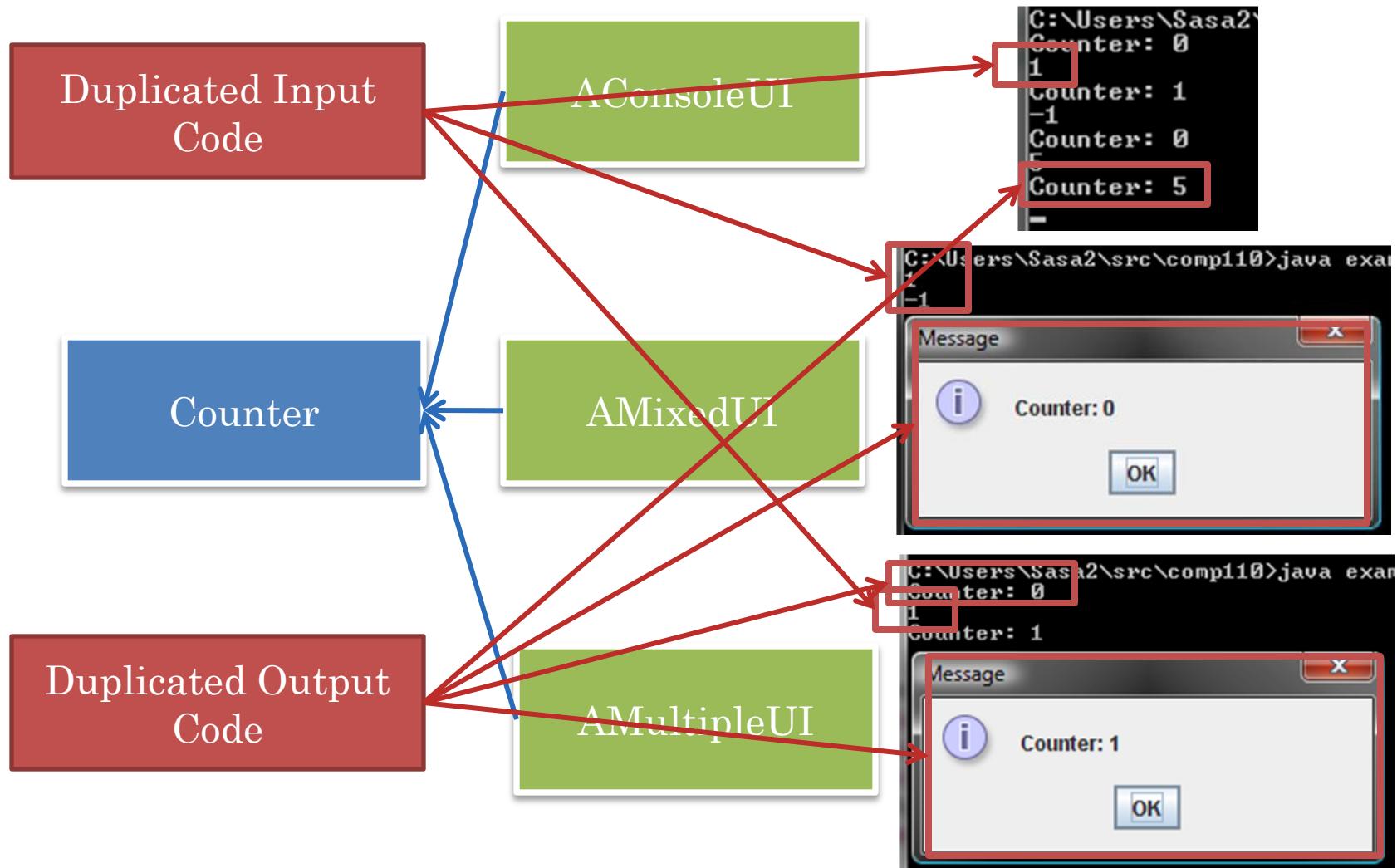
Input

I/O Code is
Duplicated

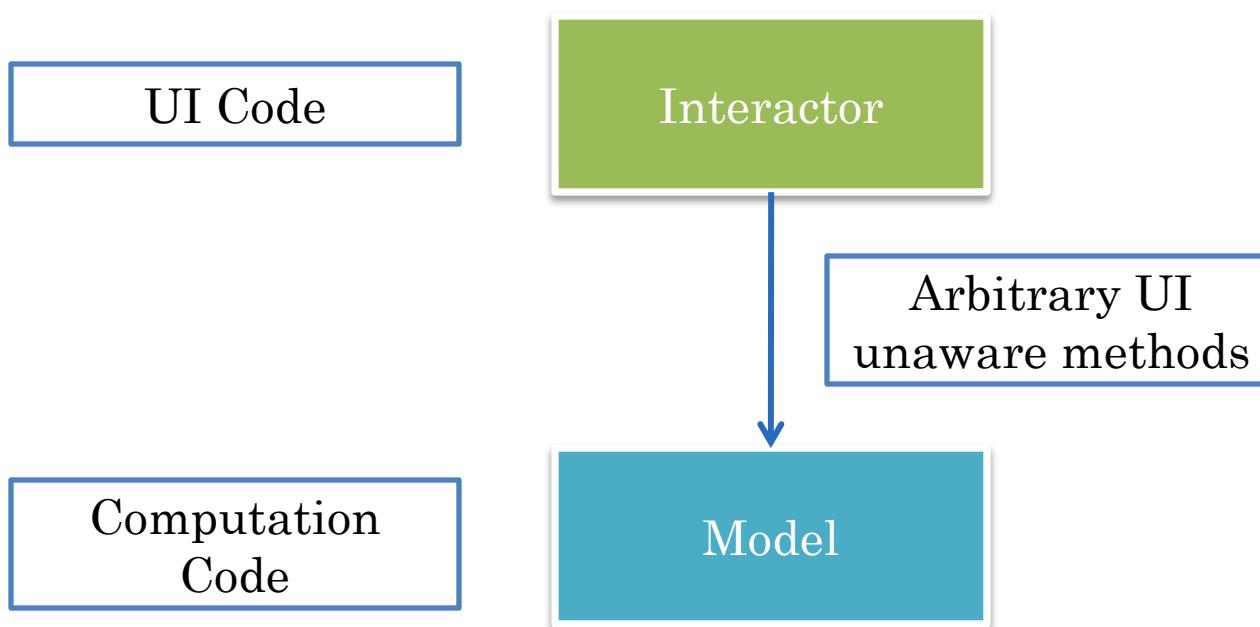
UI Implementation
is now monolithic



DRAWBACKS OF MONOLITHIC UI



MODEL/INTERACTOR PATTERN



MVC PATTERN



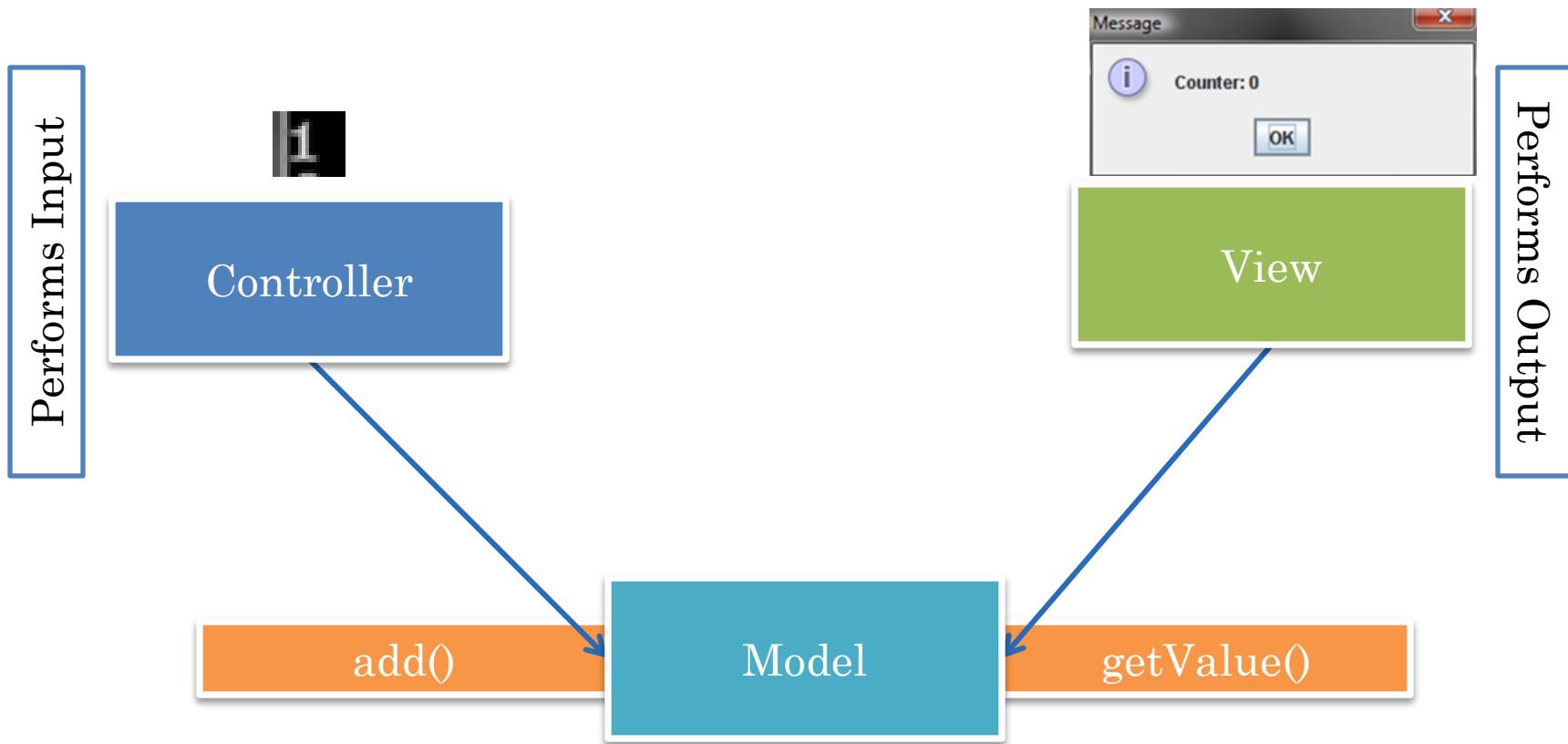
Write Methods

Model

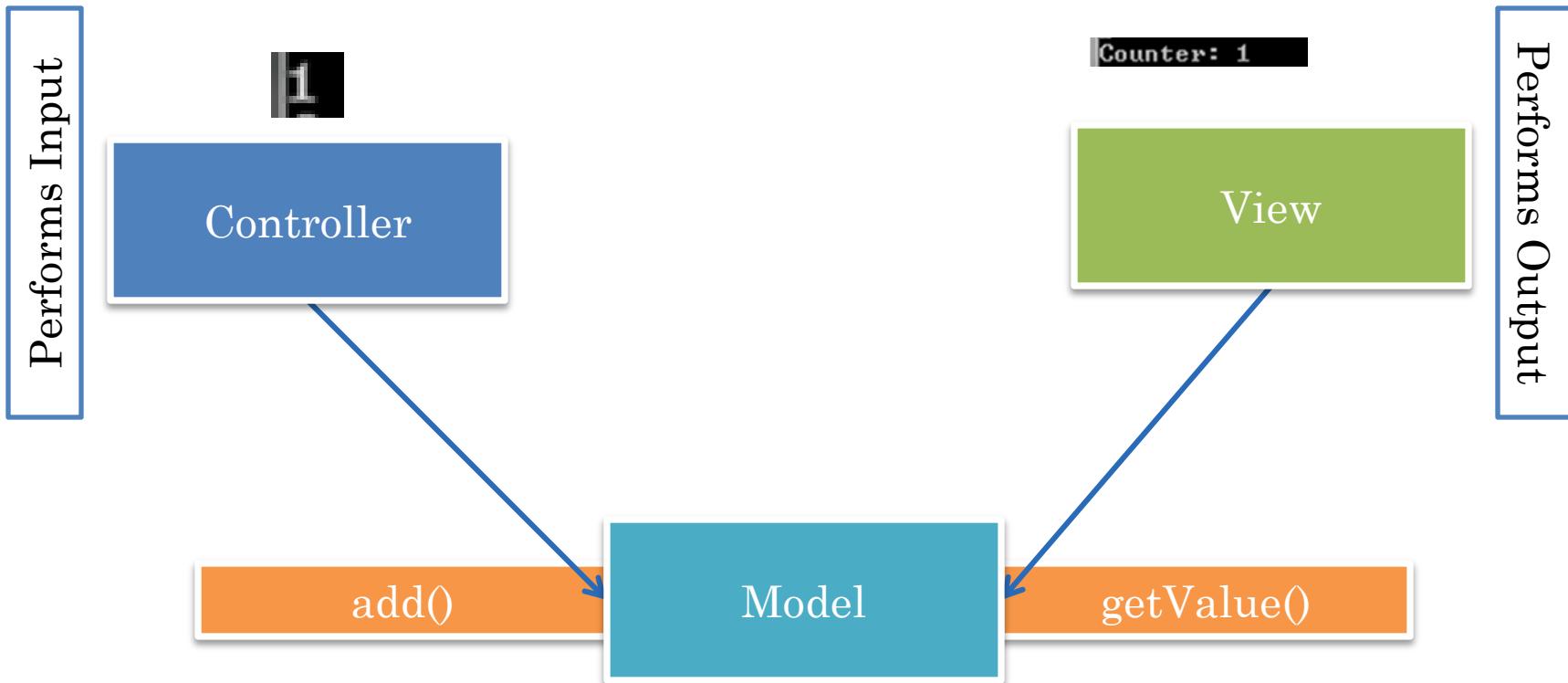
Read Methods

View can be on computer with
big screen and controller on
smart phone

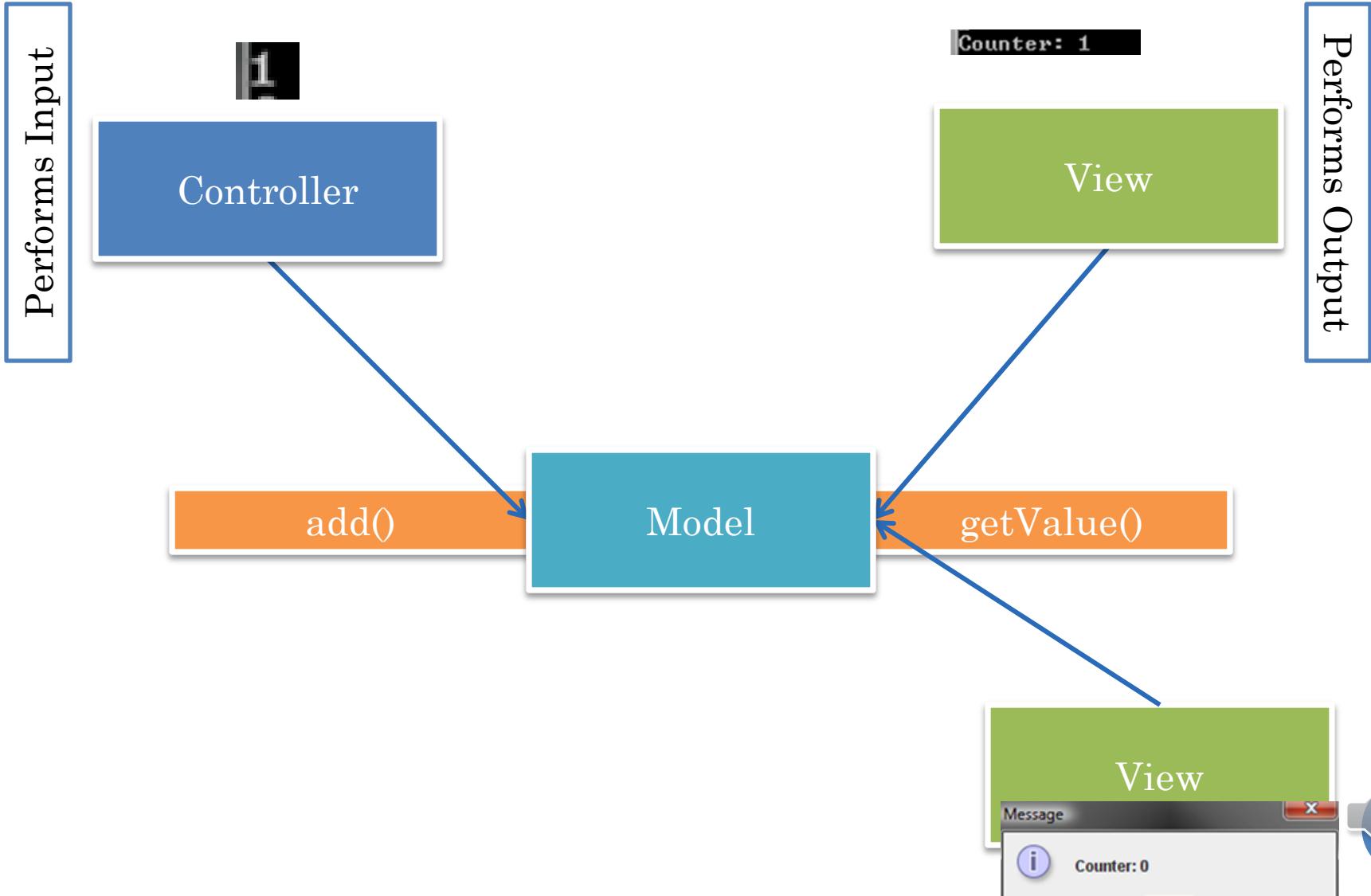
MVC PATTERN IN COUNTER



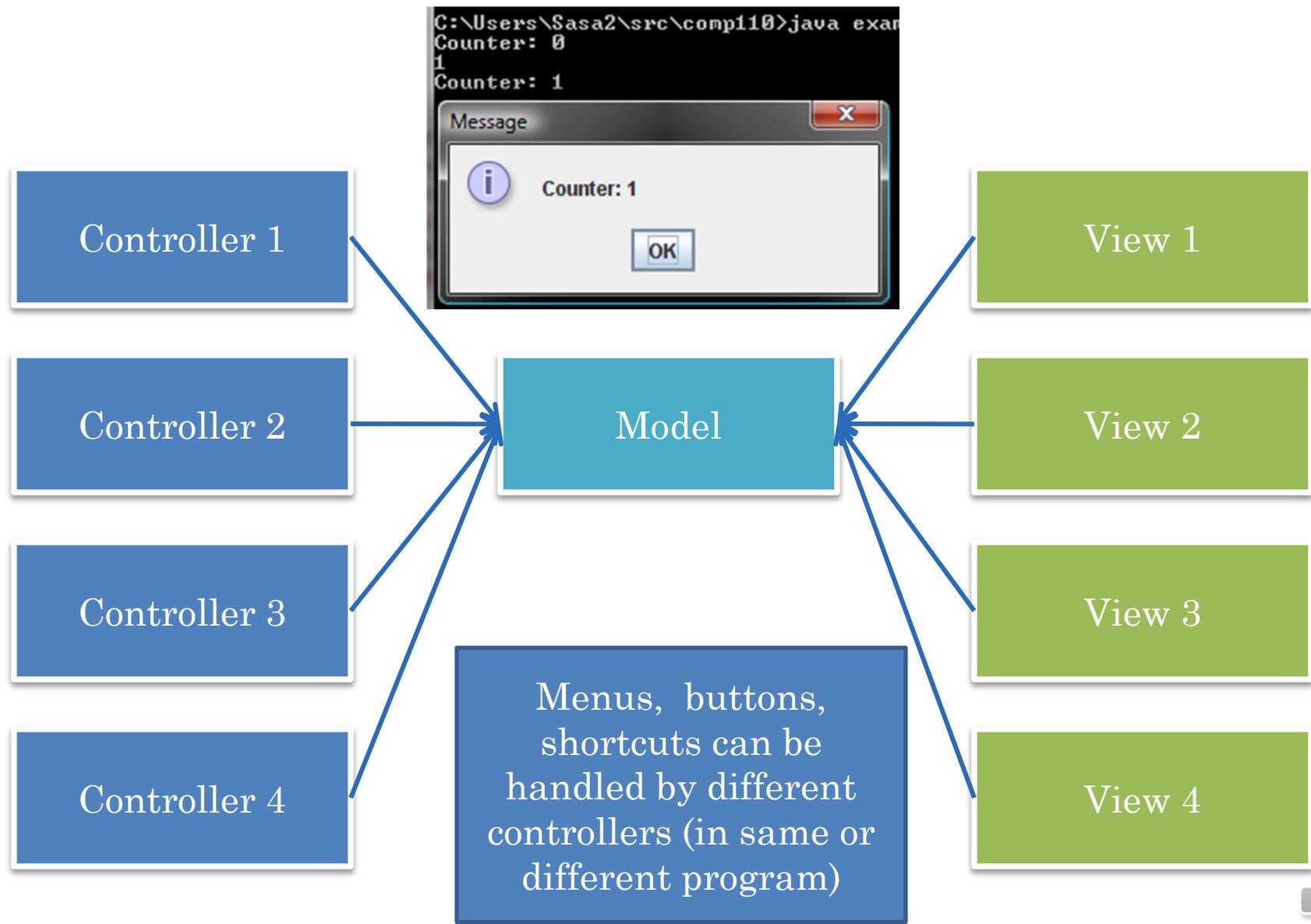
CHANGING TO CONSOLE VIEW



MULTIPLE VIEWS



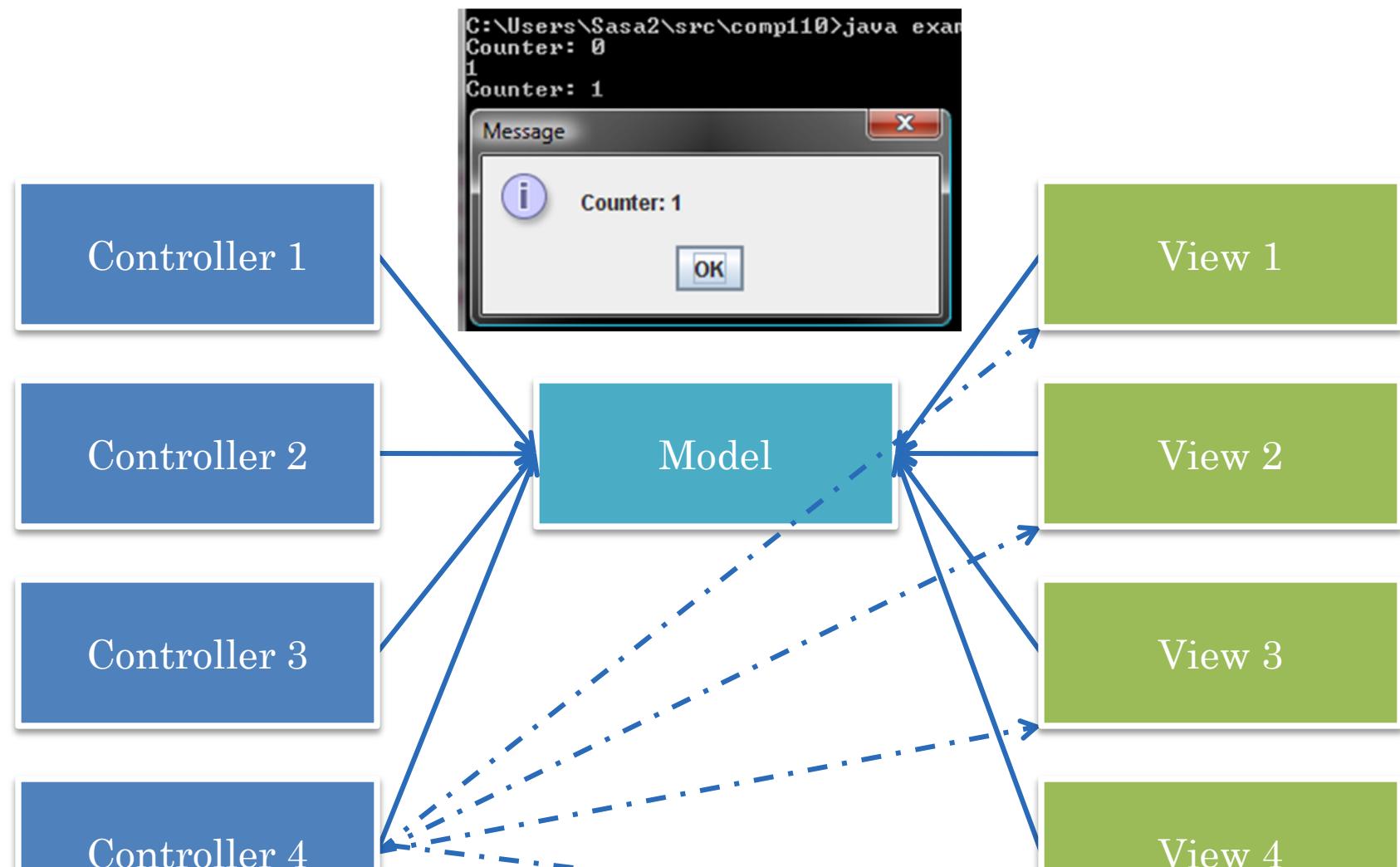
MULTIPLE VIEWS AND CONTROLLERS



```
C:\Users\Sasa2\src\comp110>java example  
Counter: 0  
1  
Counter: 1  
Message
```

A screenshot of a Java application window titled "Message". It displays the text "Counter: 1" next to an information icon. There is an "OK" button at the bottom.

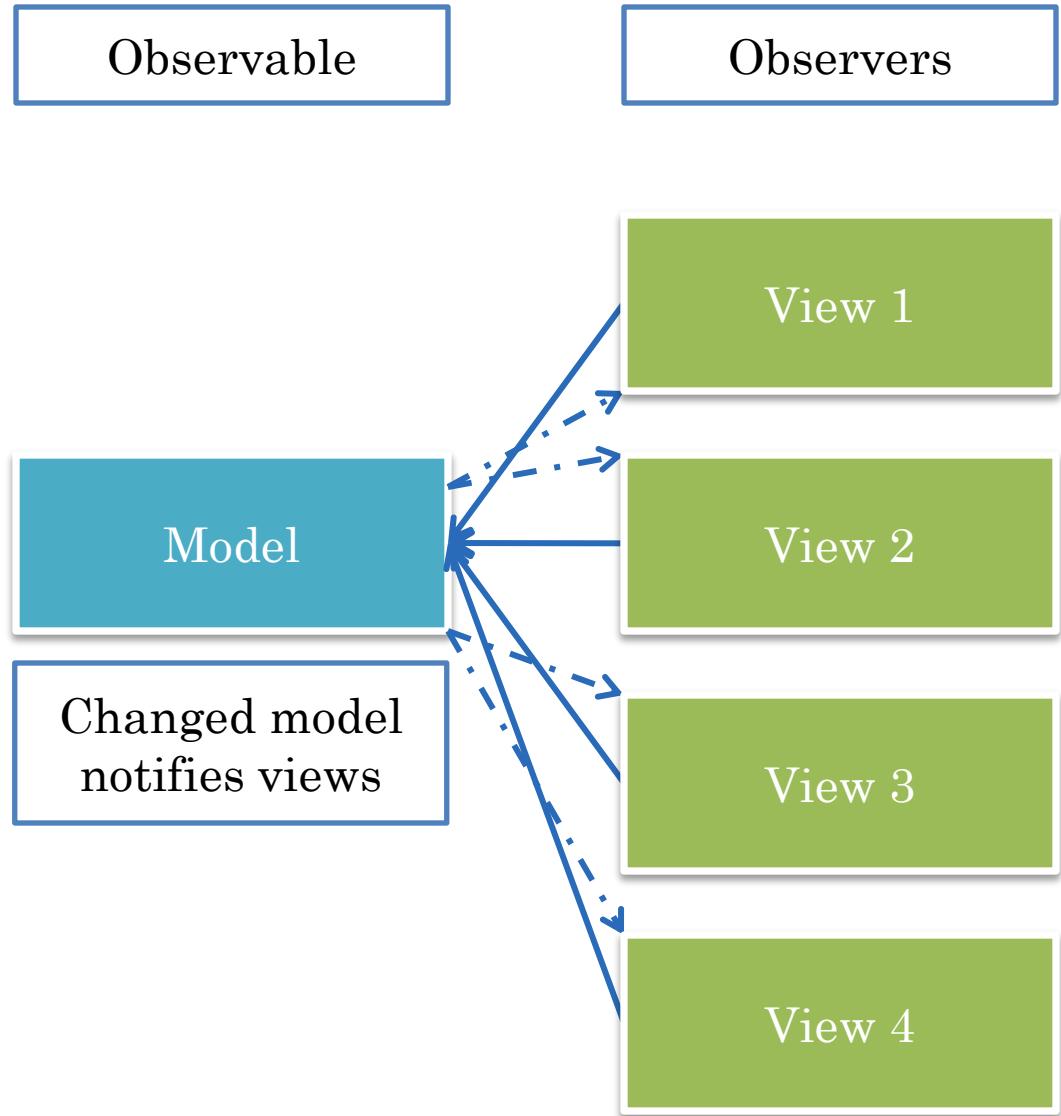
SYNCING CONTROLLERS & VIEW



In Http-based “MVC” a single view and controller exist in the browser and the model in the server. A Model cannot initiate actions in the browser so the controller directly communicates with the view



OBSERVER PATTERN



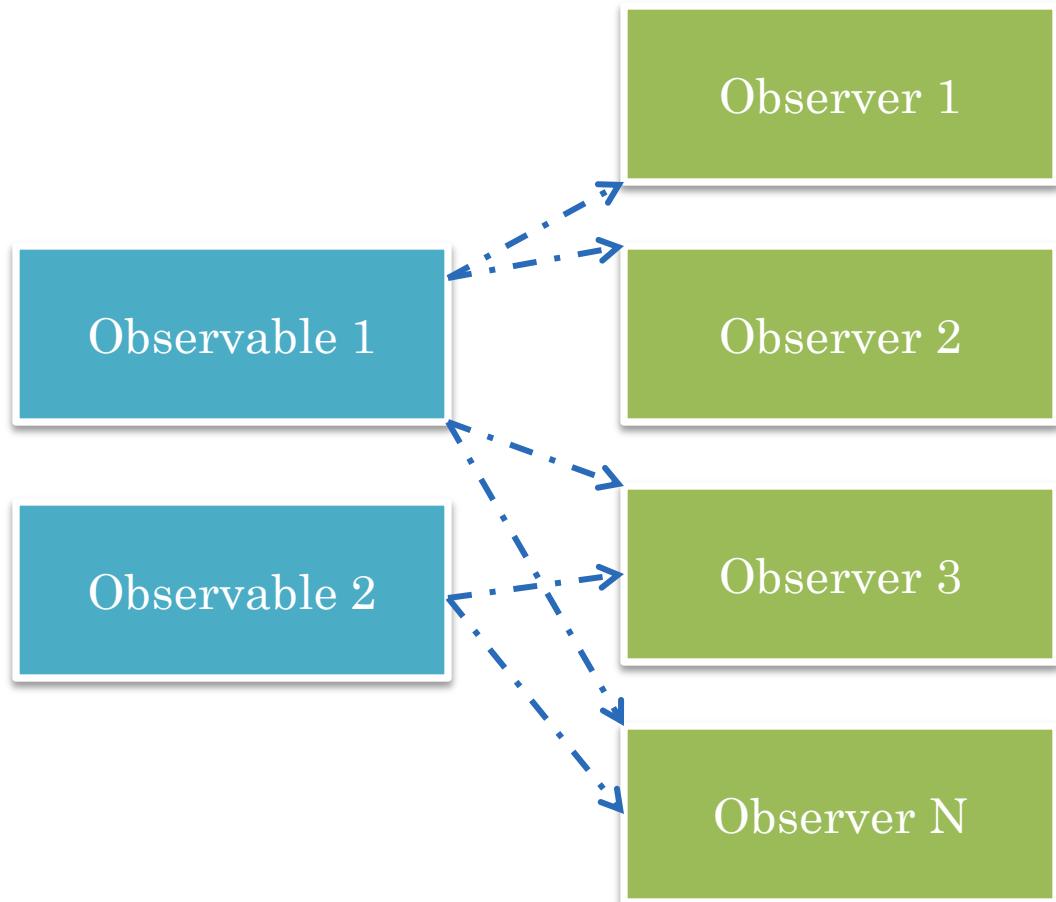
MULTIPLE OBSERVERS/OBSERVABLES

- A single battle simulation view observing

- Multiple planes
- Multiple tanks

How does observable know about its observers?

Observer registered with observable



NOTIFICATION SCHEME

Each observer is registered with observable

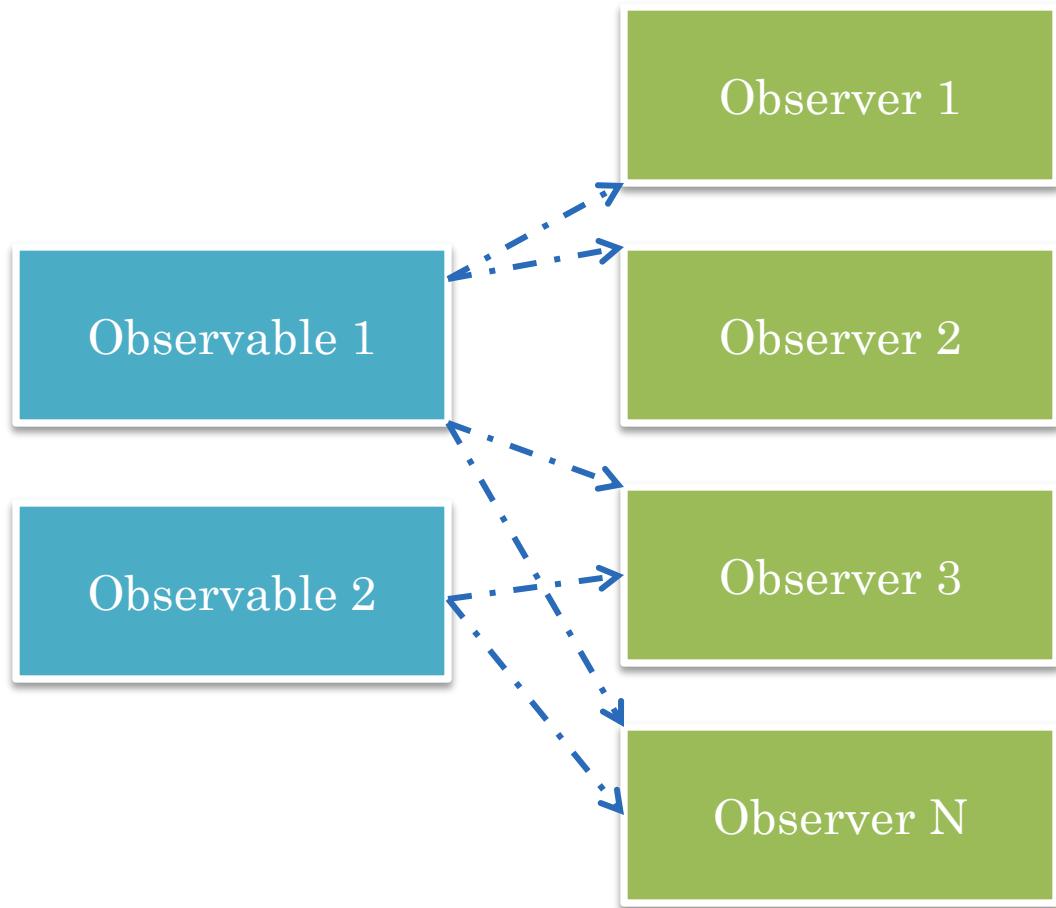
Each write method in observable calls a notification method in each observer

Notification method in observer reads model

Each student is registered with professor's listserv

When web page is updated mail sent to students

Student reads web page if mailed information is not sufficient

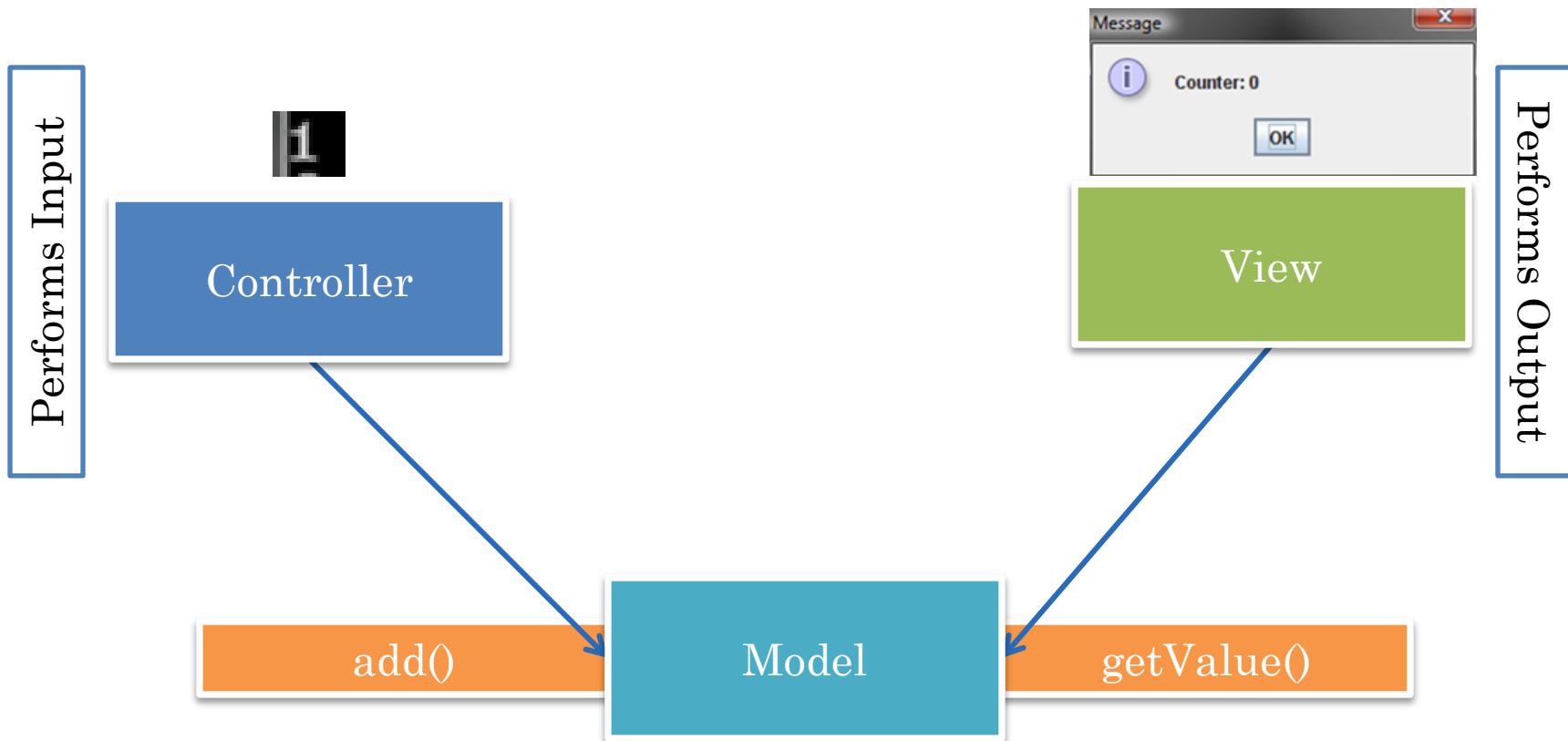


MVC PATTERN (REVIEW)



View can be on computer with
big screen and controller on
smart phone

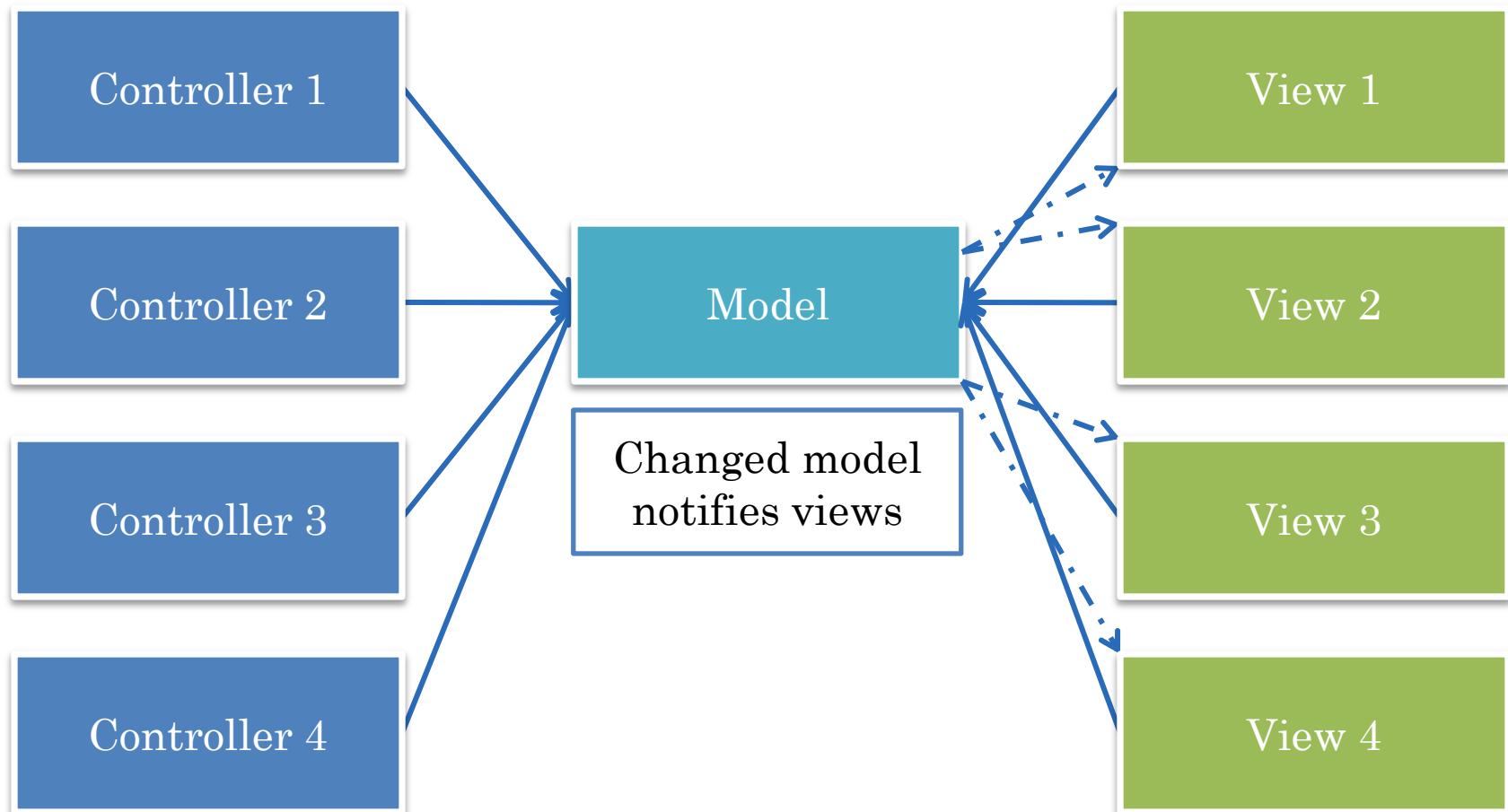
MVC PATTERN IN COUNTER (REVIEW)



OBSERVER PATTERN

Observable

Observers



NOTIFICATION SCHEME (REVIEW)

Each observer is registered with observable

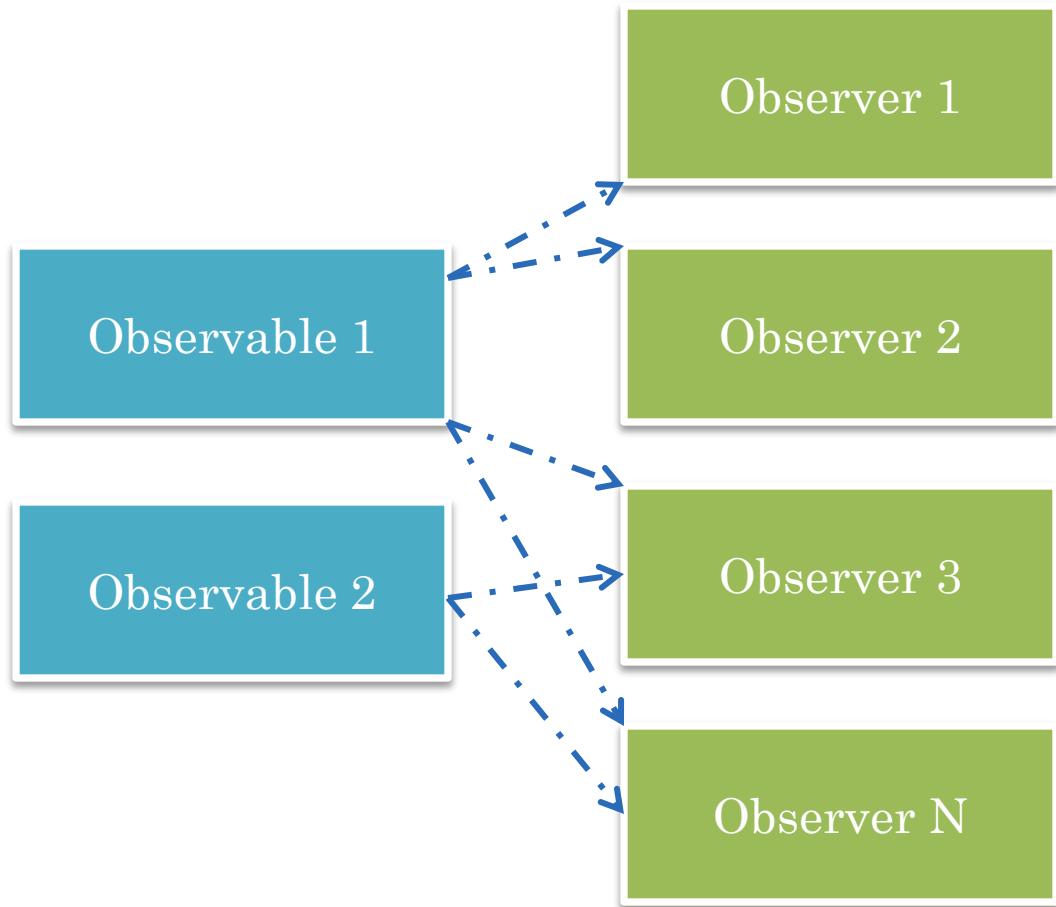
Each write method in observable calls a notification method in each observer

Notification method in observer reads model

Each student is registered with professor's listserv

When web page is updated mail sent to students

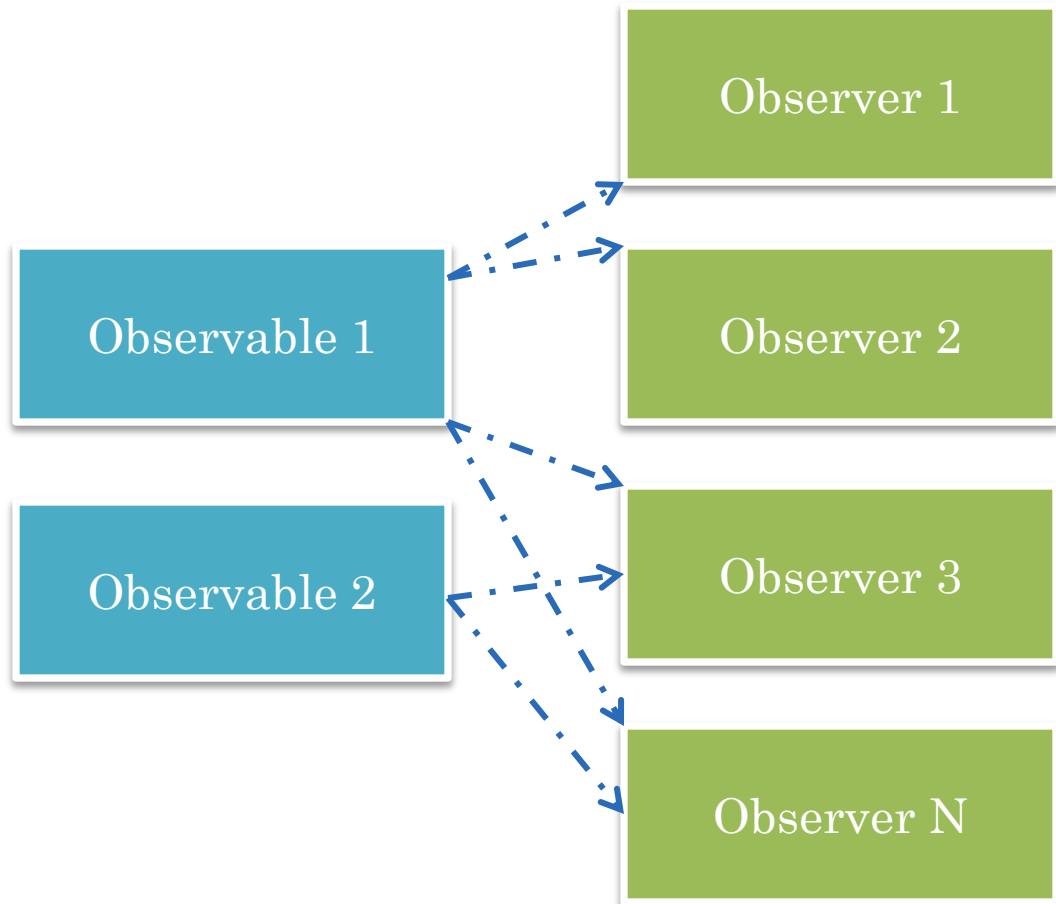
Student reads web page if mailed information is not sufficient



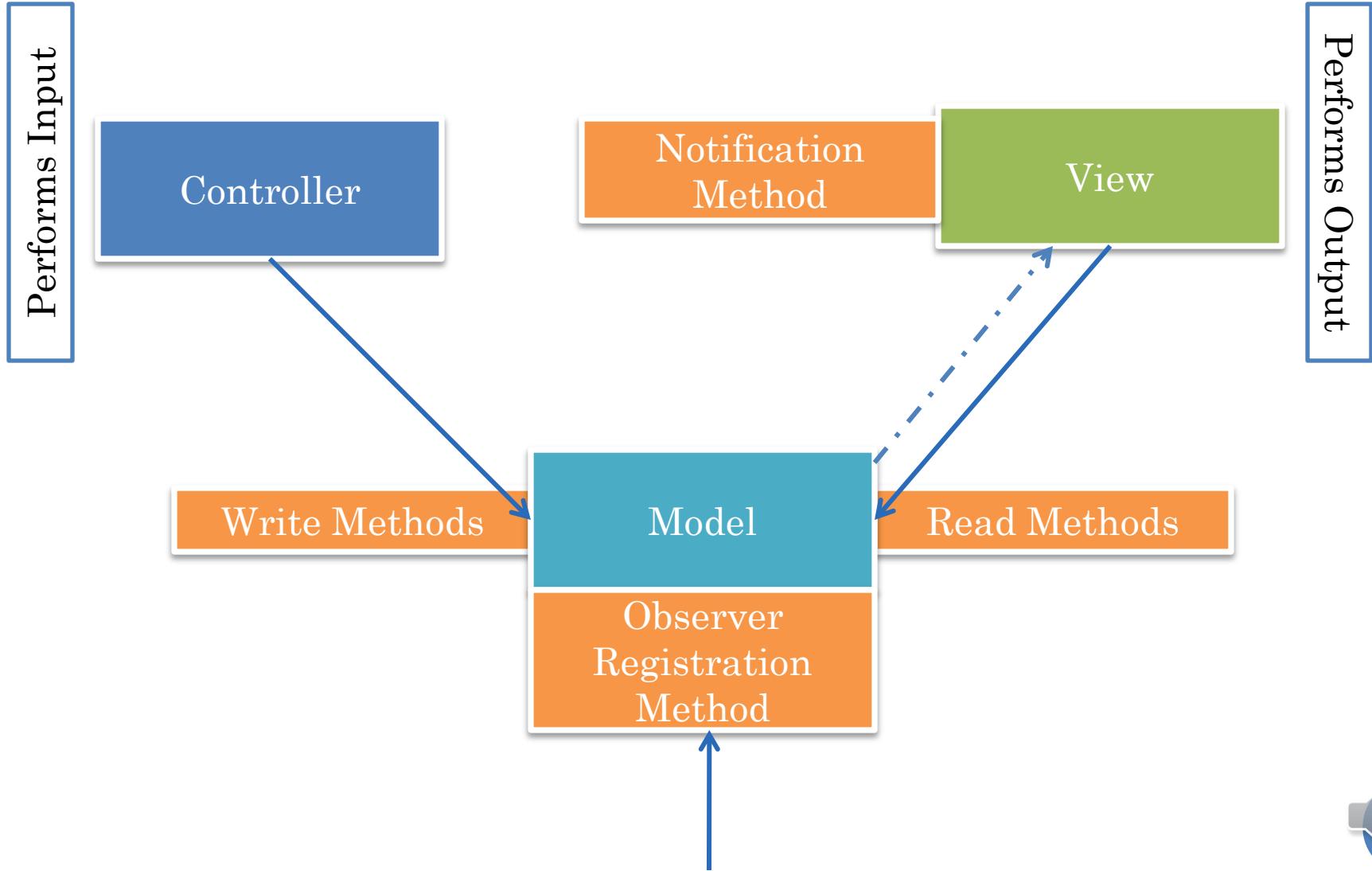
GENERAL NOTIFICATION SCHEME

Observers may have multiple observables with common notification method

Notification method parameter indicates which observable



NOTIFICATIONS IN MVC PATTERN



IMPLEMENTATION DEPENDENT ISSUES

How does controller know about model?

Who registers observer registered with observable?

Model connection method invoked on it

It registers itself if it knows about observable

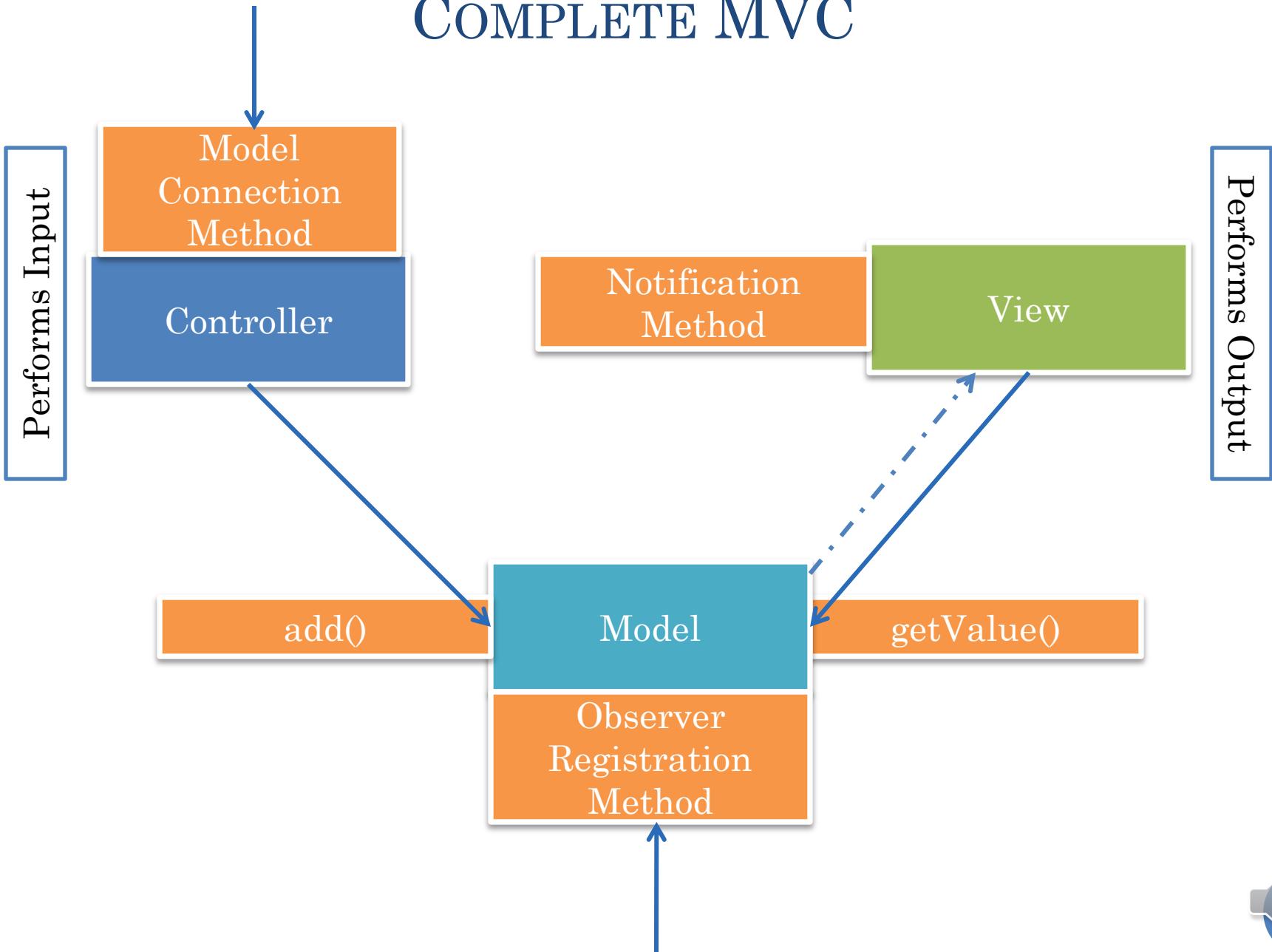
By model or some other program
• Main

Model registers it if it knows about observer

Some other code registers it
• Main



COMPLETE MVC



OBSERVABLE AND OBSERVER

Each observer is registered
with observable

Each write method in
observable calls a
notification method in each
observer

Notification method in
observer reads model



COUNTER OBSERVABLE AND OBSERVER

```
public interface ObservableCounter {  
    public void add (int amount) ;  
    public int getValue() ;  
    public void addObserver(CounterObserver observer);  
    public void removeObserver(CounterObserver observer);  
}
```

Console View,
JOptionPane View

```
public interface CounterObserver {  
    public void update(ObservableCounter counter);  
}
```

Called whenever
model is updated

Updated model



COUNTER MODEL

```
public class AnObservableCounter implements ObservableCounter {  
    int counter = 0;  
    ObserverList observers = new AnObserverList();  
    public void add (int amount) {  
        counter += amount;  
        notifyAllObservers();  
    }  
    public int getValue() {  
        return counter;  
    }  
    public void addObserver(CounterObserver observer) {  
        observers.addElement(observer);  
        observer.update(this);  
    }  
    public void removeObserver(CounterObserver observer) {  
        observers.removeElement(observer);  
    }  
    void notifyAllObservers() {  
        for (int observerNum = 0; observerNum < observers.size();  
             observerNum++)  
            observers.elementAt(observerNum).update(this);  
    }  
}
```

Give this observable initial value

Each write method notifies all!



CONSOLE VIEW

```
public class ACounterConsoleView implements CounterObserver {  
    public void update(ObservableCounter counter) {  
        System.out.println("Counter: " + counter.getValue());  
    }  
}
```



JOPTION VIEW

```
import javax.swing.JOptionPane;
public class ACounterJOptionView implements CounterObserver {
    public void update(ObservableCounter counter) {
        JOptionPane.showMessageDialog(
            null, "Counter: " + counter.getValue());
    }
}
```

CONSOLE CONTROLLER INTERFACE

```
public interface CounterController {  
    public void setModel(ObservableCounter theCounter);  
    public void processInput();  
}
```

CONSOLE CONTROLLER

```
public class ACounterController implements CounterController {  
    ObservableCounter counter;  
    public void setModel(ObservableCounter theCounter) {  
        counter = theCounter;  
    }  
    public void processInput() {  
        while (true) {  
            int nextInput = Console.readInt();  
            if (nextInput == 0) break;  
            counter.add(nextInput);  
        }  
    }  
}
```

CONSOLE MAIN

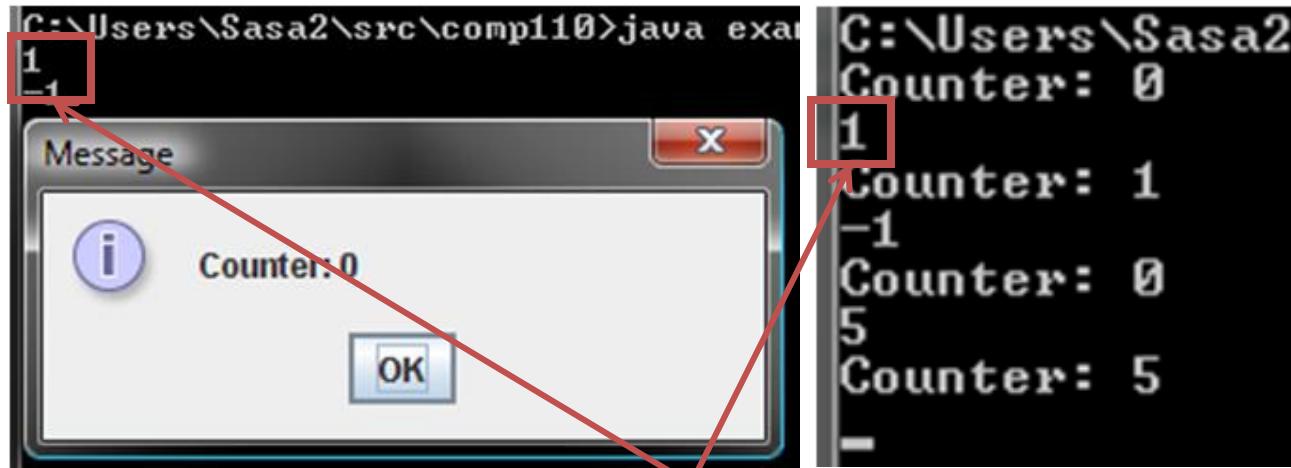
```
public static void main (String args[]) {  
    ObservableCounter model = new AnObservableCounter();  
    model.addObserver(new ACounterConsoleView());  
    CounterController controller = new ACounterController();  
    controller.setModel(model);  
    controller.processInput();  
}
```

```
C:\Users\Sasa2\  
Counter: 0  
1  
Counter: 1  
-1  
Counter: 0  
5  
Counter: 5  
-
```



CONSOLE AND JOPTION MAIN

```
public static void main (String args[]) {  
    ObservableCounter model = new AnObservableCounter();  
    model.addObserver (new ACounterJOptionView());  
    CounterController controller = new ACounterController();  
    controller.setModel(model);  
    controller.processInput();  
}
```

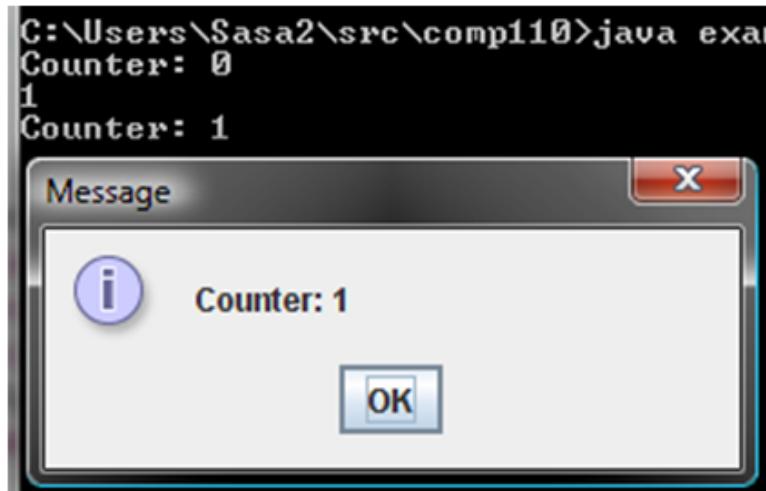


Shared input code



MIXED UI MAIN

```
public static void main (String args[]) {  
    ObservableCounter model = new AnObservableCounter();  
    model.addObserver(new ACounterJOptionView());  
    model.addObserver (new ACounterConsoleView());  
    CounterController controller = new ACounterController();  
    controller.setModel(model);  
    controller.processInput();  
}
```



EFFICIENCY

```
public interface CounterObserver {  
    public void update(ObservableCounter counter);  
}
```

```
public class ACounterConsoleView implements CounterObserver {  
    public void update(ObservableCounter counter) {  
        System.out.println("Counter: " + counter.getValue());  
    }  
}
```

What if observer is in USA and observable in China?

Update must make a “long distance” call to read method (getValue()) to update counter state



NOTIFICATION WITH CHANGE DESCRIPTION

No need to call read method after
notification

```
public interface CounterObserver {  
    public void update(ObservableCounter counter, int newCounterVal);  
}
```

```
public class ACounterConsoleView implements CounterObserver {  
    public void update(ObservableCounter counter, int  
newCounterVal) {  
        System.out.println("Counter: " + newCounterVal));  
    }  
}
```



OBJECTEDITOR UPDATE?

```
public interface CounterObserver {  
    public void update(ObservableCounter counter, int newCounterVal);  
}
```

Can ObjectEditor become a view of Counter so no need to call refresh?

ObjectEditor does not know about CounterObserver and cannot implement it.



JAVA.UTIL.OBSERVER AND OBSERVABLE

“Standard” observer interface talking arbitrary change Object argument

```
public interface java.util.Observer {  
    public void update(Observable o, Object arg);  
}
```

```
public class java.util.Observable {  
    public void addObserver(Observer o) { ... };  
    public void notifyObservers() { ... };  
}
```

Model must be subclass of Observable



EXTRA



CIRCULARITY

```
public interface ObservableCounter {  
    public void add (int amount) ;  
    public int getValue() ;  
    public void addObserver(CounterObserver observer);  
    public void removeObserver(CounterObserver observer);  
}
```

Cannot compile ObservableCounter
without CounterObserver and vice
versa

```
public interface CounterObserver {  
    public void update(ObservableCounter counter);  
}
```

BREAKING CIRCULARITY: MULTIPLE STAGES

```
public interface ObservableCounter {  
    public void add (int amount) ;  
    public int getValue() ;  
}
```

CounterObserver references compiled
ObservableCounter

```
public interface CounterObserver {  
    public void update(ObservableCounter counter);  
}
```

CIRCULARITY

```
public interface ObservableCounter {  
    public void add (int amount) ;  
    public int getValue() ;  
    public void addObserver(CounterObserver observer);  
    public void removeObserver(CounterObserver observer);  
}
```

Recompiled observable references
compiled CounterObserver

```
public interface CounterObserver {  
    public void update(ObservableCounter counter);  
}
```

CIRCULARITY AND BREAKING IT

Circularity

Two types reference each other

Neither can be compiled without the other

General approach to breaking it

Create both types as empty and compile them so they are known to Java

Next add references to each other

OBSERVERS THAT ARE NOT VIEWS

- Spreadsheet cell
 - observes cells on which it depends
- Monitoring of appliance usage
 - Each time I do setChannel() on TV event logged
- Eclipse quiz/activity plug-in
 - Observers Eclipse events
- Any big brother app!
- Counter observer?

OBSERVERS THAT ARE NOT VIEWS

- Spreadsheet cell
 - observes cells on which it depends
- Monitoring of appliance usage
 - Each time I do setChannel() on TV event logged
- Eclipse quiz/activity plug-in
 - Observers Eclipse events
- Any big brother app!
- Counter observer?

ROCKET OBSERVER

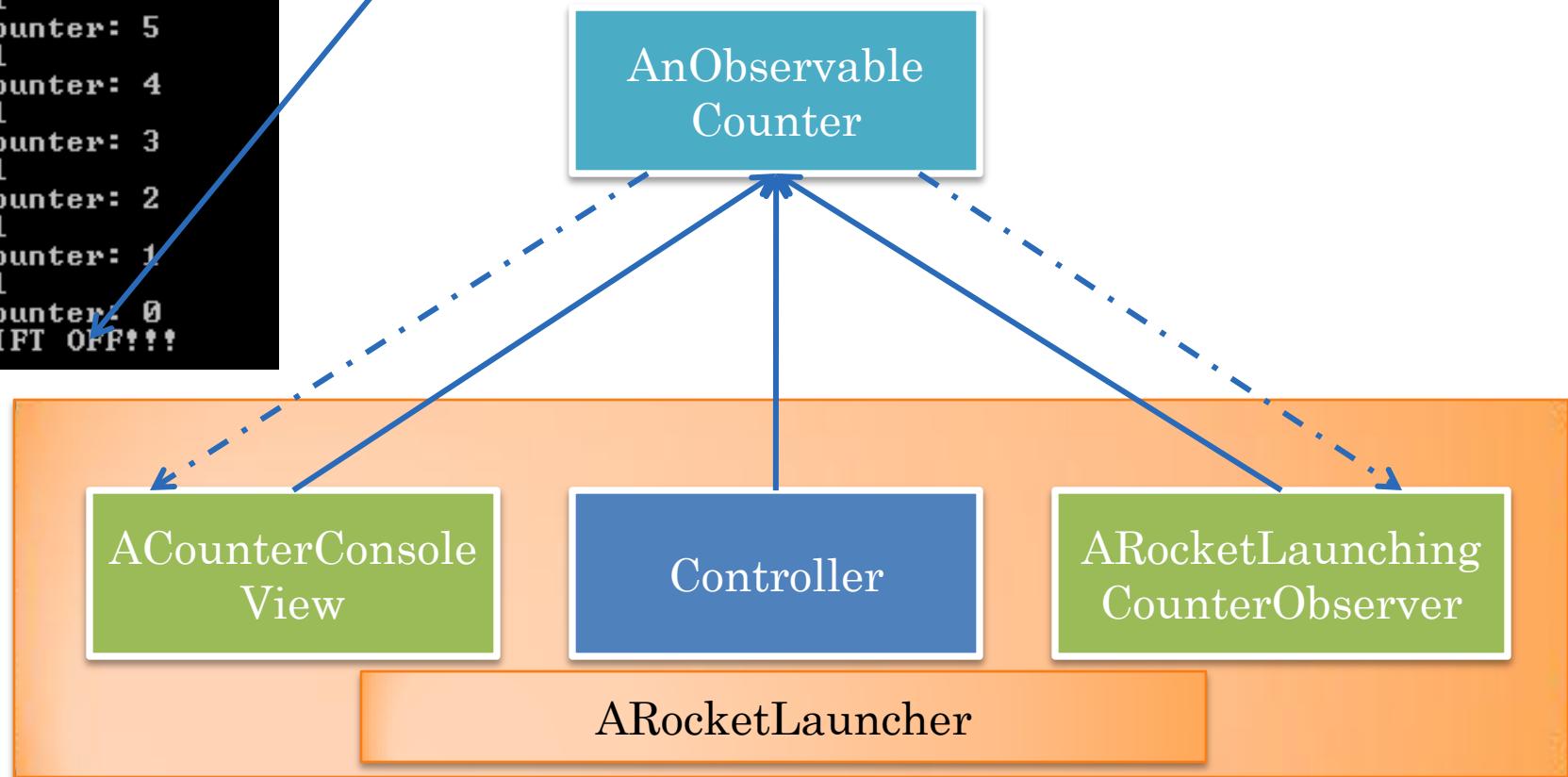
```
C:\Users\Sasa>
Counter: 10
-1
Counter: 9
-1
Counter: 8
-1
Counter: 7
-1
Counter: 6
-1
Counter: 5
-1
Counter: 4
-1
Counter: 3
-1
Counter: 2
-1
Counter: 1
-1
Counter: 0
LIFT OFF!!!
```

```
public class ARocketLaunchingCounterObserver
    implements CounterObserver {
    public void update(ObservableCounter counter) {
        if (counter.getValue() == 0)
            launch();
    }
    private void launch() {
        System.out.println("LIFT OFF!!!");
    }
}
```

INSTANCES CREATED AND COMPOSED

```
C:\Users\Sasa2\s:  
Counter: 10  
-1  
Counter: 9  
-1  
Counter: 8  
-1  
Counter: 7  
-1  
Counter: 6  
-1  
Counter: 5  
-1  
Counter: 4  
-1  
Counter: 3  
-1  
Counter: 2  
-1  
Counter: 1  
-1  
Counter: 0  
LIFT OFF!!!
```

Console View added before rocket launching observer



ROCKET LAUNCHING MAIN

```
public static void main (String args[]) {  
    ObservableCounter model = new AnObservableCounter();  
    model.addObserver (new ACounterConsoleView());  
    model.addObserver(new ARocketLaunchingCounterObserver());  
    CounterController controller = new ACounterController();  
    controller.setModel(model);  
    controller.processInput();  
}
```

BASIC NOTIFICATION

```
public interface CounterObserver {  
    public void update(ObservableCounter counter);  
}
```

Called when observer is updated

Updated Observable

IMPLICIT OBSERVER

```
public interface CounterObserver {  
    public void setObservable(ObservableCounter counter);  
    public void update();  
}
```

Updated Observable



Assuming observer has only one observable

DISTRIBUTION ISSUES

```
public interface CounterObserver {  
    public void update(ObservableCounter counter);  
}
```

What if observer is in USA and observable in China?

Update must make a “long distance” call to read method (getValue()) to update counter state

NOTIFICATION WITH CHANGE DESCRIPTION

No need to call read method after
notification

```
package models;  
public interface CounterObserver {  
    public void update(ObservableCounter counter, int newCounterVal);  
}
```

JAVA JAVA.UTIL.OBSERVER

```
public interface java.util.Observer {  
    public void update(Observable o, Object arg);  
}
```

“Standard” observer interface talking
arbitrary change Object argument

NOTIFICATION WITH CHANGED VALUE

New value of observable attribute

```
public interface CounterObserver {  
    public void update(ObservableCounter counter, int newCounterVal);  
}
```

NOTIFICATION WITH CHANGE

Difference between new and old value of observable attribute

```
public interface CounterObserver {  
    public void update(ObservableCounter counter,  
                      int counterIncrement);  
}
```

Observer may display change to user

Observer interested in change does not need to keep old value to determine change

Observer interested in absolute value must keep old value

NOTIFICATION WITH NEW AND OLD VALUE

Old and new value of observable attribute

```
public interface CounterObserver {  
    public void update (ObservableCounter counter,  
                      int oldCounterValue, int newCounterValue);  
}
```

Observer interested in change does not need to keep old value to determine change

Observer interested in absolute value need not keep old value

Makes observer harder to code

NOTIFICATION WITH SINGLE EVENT OBJECT

```
public interface CounterObserver {  
    public void update(  
        CounterChangeEvent event);  
}
```

```
public interface CounterChangeEvent {  
    ObservableCounter getCounter();  
    int getOldCounterValue();  
    int getNewCounterValue();  
}
```

- Easy to pass single object to different methods handling event
- Can make event info very elaborate
 - Time when event occurred
 - Unique ID for event
 -
- Callee does not have to declare parameters for event information fields not of interest
- Caller does not have to fill every value – can put null for object values such as counter and illegal values for primitives

JAVA ACTIONEVENT

```
import java.awt.Event;  
  
public interface java.awt.ActionListener {  
    public void actionPerformed(ActionEvent e);  
}
```

When you edit text and hit return this event sent by JTextField, TextField widget to its listeners such as ObjectEditor

When you press a button, this event sent by Button/Jbutton to its lsiteners such as ObjectEditor

OBSERVING MULTIPLE PROPERTIES

```
public interface BMISpreadsheet {  
    public double getHeight();  
    public void setHeight(int newVal);  
    public double getWeight();  
    public void setWeight(int newWeight);  
    public double getBMI();  
}
```

Observer Interface?

SINGLE COARSE-GRAINED UPDATE

```
public interface BMISpreadsheet {  
    public double getHeight();  
    public void setHeight(int newVal);  
    public double getWeight();  
    public void setWeight(int newWeight);  
    public double getBMI();  
    ....  
}
```

```
public interface BMIObserver {  
    public void update(  
        BMISpreadsheet bmiSpreadsheet);  
}
```

Coarse grained updated

Each setter sends the whole object

Observer must determine which property
changed

MULTIPLE FINE-GRAINED UPDATES

```
public interface BMISpreadsheet {  
    public double getHeight();  
    public void setHeight(int newVal);  
    public double getWeight();  
    public void setWeight(int newWeight);  
    public double getBMI();  
    ...  
}
```

```
public interface BMIObserver {  
    public void updateHeight(  
        BMISpreadsheet bmi, int oldHeight, int newHeight);  
    public void updateWeight(  
        BMISpreadsheet bmi, int oldWeight, int newWeight);  
    public void updateBMI(  
        BMISpreadsheet bmi, double oldBMI, double newBMI);  
}
```

SINGLE FINE-GRAINED UPDATE METHOD

```
public interface BMISpreadsheet {  
    public double getHeight();  
    public void setHeight(int newVal);  
    public double getWeight();  
    public void setWeight(int newWeight);  
    public double getBMI();  
    ...  
}
```

```
public interface BMIObserver {  
    public void update(  
        BMISpreadsheet bmi, String propertyName,  
        Object oldValue, Object newValue);  
}
```

“Wght”

“One”

New methods not needed as new properties added

Different setters calls the same update method with different types of values.

Can be used for arbitrary property values

Can make mistakes and must process property name to determine what changed

CUSTOM SINGLE FINE-GRAINED UPDATE METHOD

```
public void setHeight (int newVal) {  
    int oldVal = height;  
    height = newVal;  
    notifyAllObservers(this, "height", oldVal, newVal);  
}
```

Can make mistake

```
public void notifyAllObservers(BMISpreadsheet source, String  
    propertyName, Object oldValue, Object newValue) {  
    for (int index = 0; index < observers.size(); index++) {  
        observers.elementAt(index).update(source, propertyName,  
            oldValue, newValue);  
    }  
}
```

BMIObserver