House of the future

Complex communication involved

Too much *coupling* between components!

Mediator

Class responsible for *coordinating* interactions among group of *colleagues*

Colleagues refer to *mediator*, not each other

Can manage *complex* interaction protocols

Possible Design Skeleton

House of the future

Complex communication involved
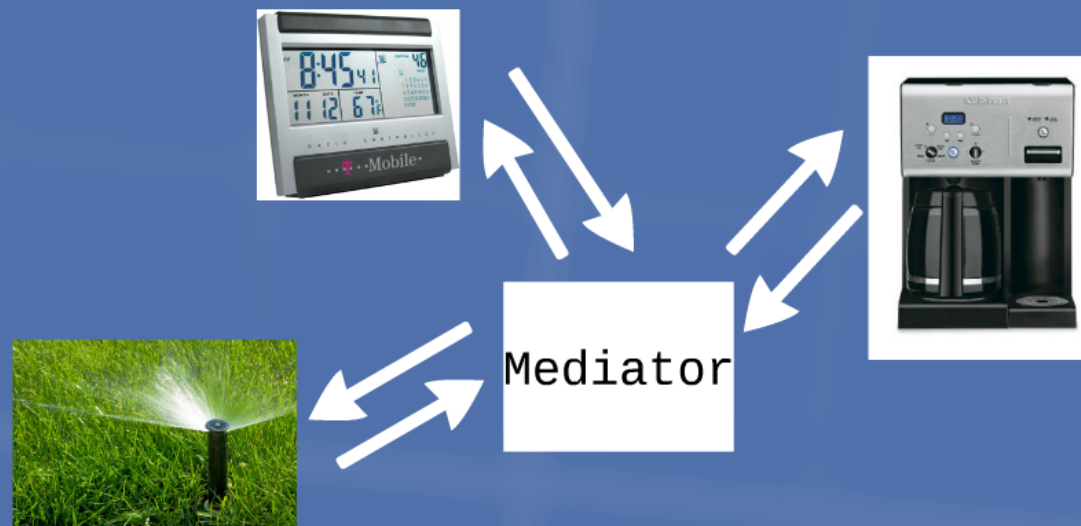
Too much *coupling* between components!

Mediator

Class responsible for *coordinating* interactions among group of *colleagues*

Colleagues refer to *mediator*, not each other

Can manage *complex* interaction protocols

Possible Design Skeleton

Possible Design Skeleton

<<Interface>>
Mediator

+ registerColleague(Colleague)
+ sendMessage(String msg, Colleague sender)

<<Interface>>
Colleague

+ notify(String msg, Colleague sender)
+ send(String msg)

ConcreteMediator

- colleagues: List<Colleague>

+ registerColleague(Colleague)
+ sendMessage(String msg, Colleague sender)

ColleagueA

- mediator: Mediator

+ ColleagueA(Mediator)
+ notify(String msg, Colleague sender)
+ send(String msg)

ColleagueB

- mediator: Mediator

+ ColleagueB(Mediator)
+ notify(String msg, Colleague sender)
+ send(String msg)

```
{
for each c in colleague {
   if(c != sender {
      c.notify(msg, sender);
   }
 }
}
```

```
{
   mediator.sendMessage(msg, this);
}
```