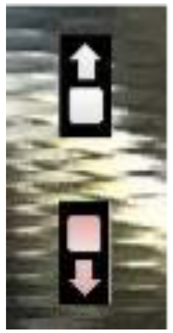
Lift Example

Problem Description

If you work for a lift manufacturer, you might have to code software for the operation of lifts. Optimizing the movements of lifts so as to minimize waiting times is not a small task. Knowing how many lifts are required in an office building depending on the number of people working in this building can also be challenging. One complicating factor is the need to model peak usage as people tend to use the lifts at the same time. Modelling and simulation can be used to do this, which takes us back to the roots of Object-Oriented programming.

Implementation

We may think of creating a lift class. First of all, it must know which floors it serves. Then, it has a state: it may be stopped (for maintenance sometimes) or moving, up or down, and won't change direction before it has reached an "extreme stop". Messages are of two kinds: an "up" or "down" call from the outside (floor is known) and a ”get me to floor" call when people press a button inside.



Lift Class

- Floors served
- Direction
- Moving/Stopped
- Stop request

This model works when there is a single lift, but not when you have an "elevator lobby" served by several lifts. When you call a lift, you care very little about which lift will stop at your floor as long as it goes into the right direction.

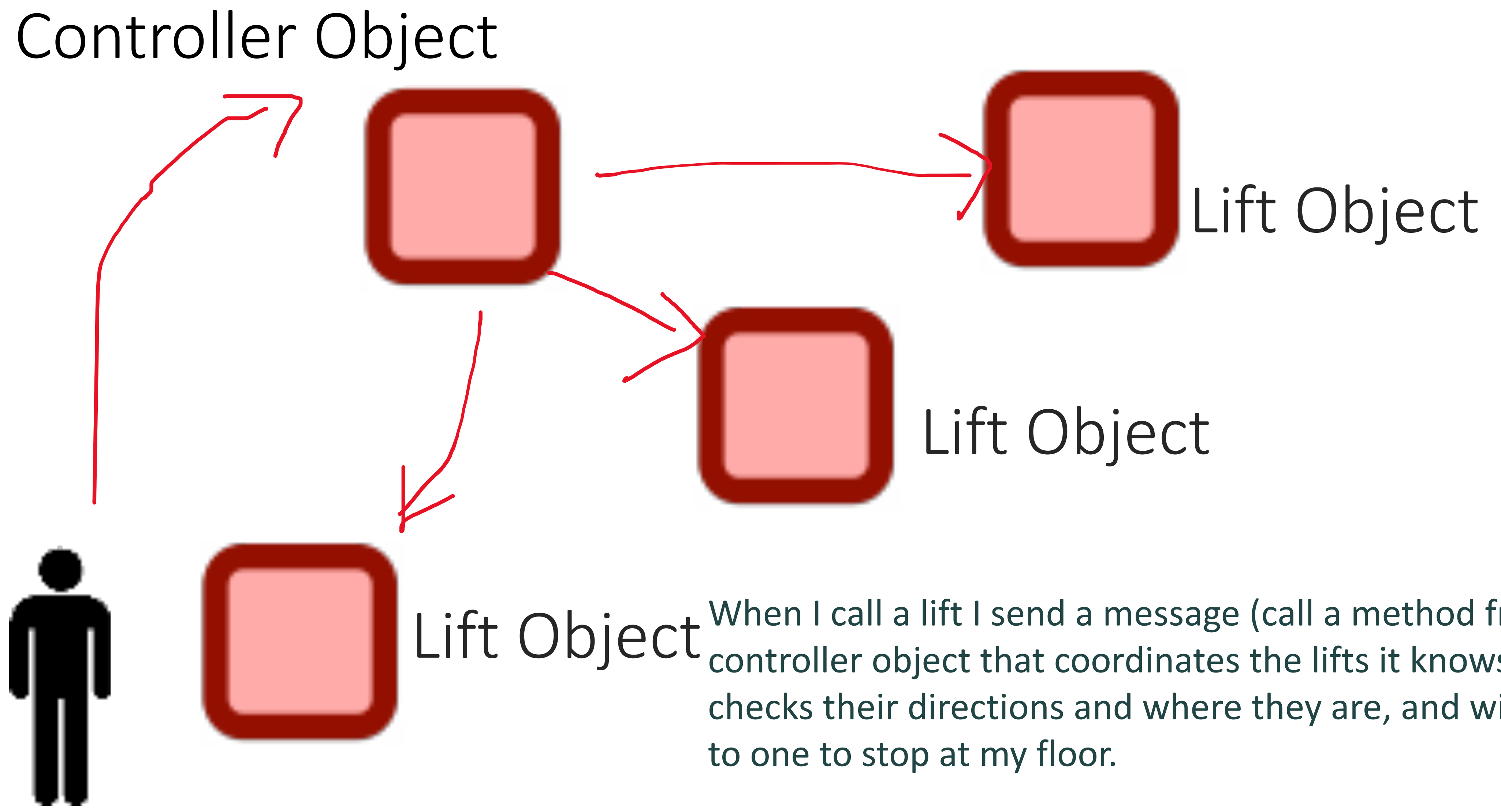
Controller Class

- Lift Requests

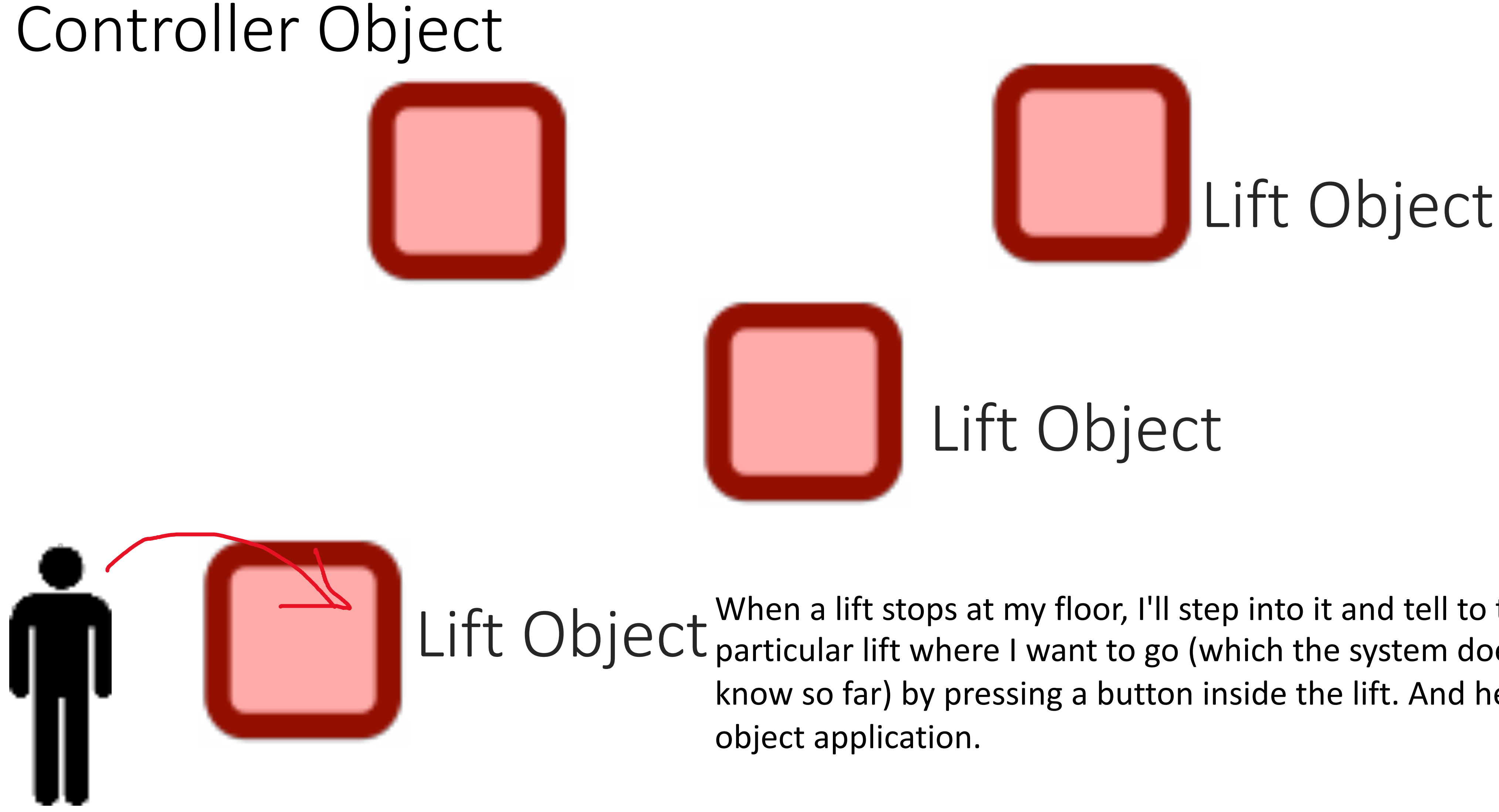
In that case we need a more abstract class, which I call "Controller class", for coordinating several lifts and taking call requests. Buttons inside the lift are still messages to a lift object that represents the lift you are in.

Lift Class

- Floors served
- Direction
- Moving/Stopped
- Stop request



When I call a lift I send a message (call a method from) a controller object that coordinates the lifts it knows, checks their directions and where they are, and will ask to one to stop at my floor.



When a lift stops at my floor, I'll step into it and tell to that particular lift where I want to go (which the system doesn't know so far) by pressing a button inside the lift. And here is my object application.

Creating an Object Oriented Application

- Identify necessary objects
- Define their role (what they do)
- Define the data they need
- Set up communications

the hard part

- **Maximize consistency and minimize coupling**
- Trade-off between one object that does everything and objects that send too many messages