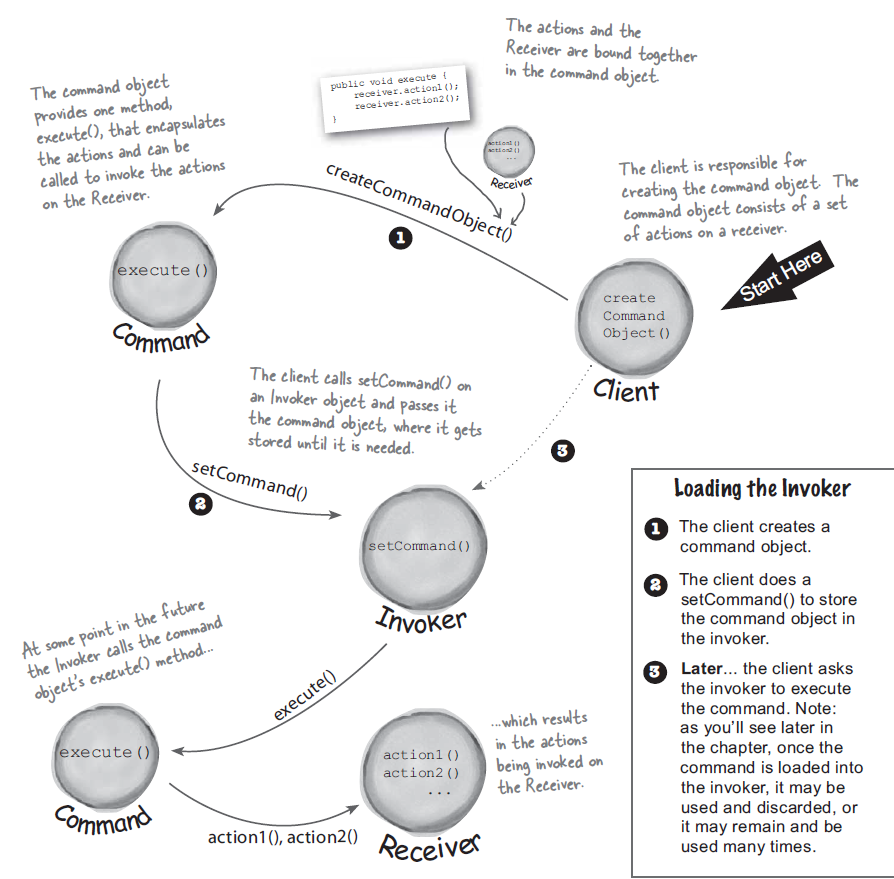
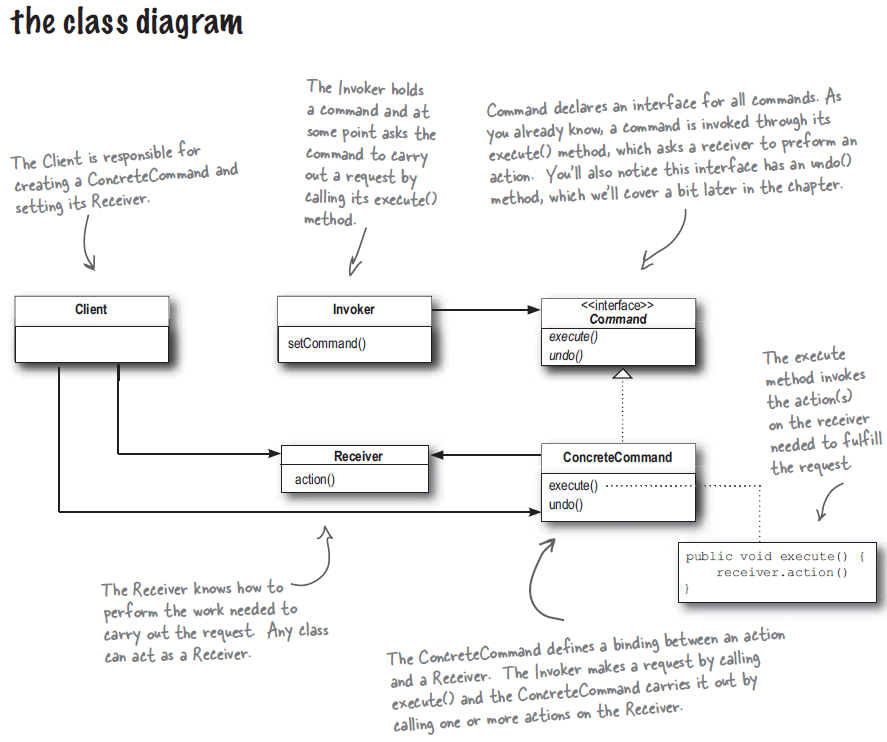
1. **Definition**: The Command Pattern encapsulates a request as an object thereby enabling parameterization of other objects with different requests.
2. Command pattern is about encapsulating invocation to objects.
3. Command pattern allows us to decouple requestor of an action from the object that performs the action. A command object encapsulates a request to do something on a specific object. The requestor just communicates with command objects with standard interface without knowing how each command object internally performs action on a specific object.

**Ex**: Requestor - Remote control; Performer - instance of vendor class.



1. Class diagram:



1. Importantly, the receiver of the request is bound to the command it’s encapsulating in.
2. Null object is useful when you don’t have meaningful object to return and yet you want to remove the responsibility of handling NULL from client.
3. In general, command object should be “dumb” which just invoke an action on a receiver. But a command object can be smart that implement logic needed to carry out the request
   1. We will have more coupling in this case.
   2. We won’t able to parameterize the command object with receivers to have a macro command.
4. This pattern ends up forcing a lot of Command classes that will make your design look cluttered - more operations being made possible leads to more command classes. Intelligence required of which Command to use and when leads to possible maintenance issues for the central controller.