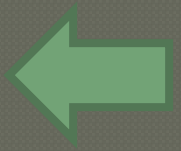
Design Patterns

- The Strategy Pattern
- The Factory Method
- Generics
- The Abstract Factory Pattern
- The State Pattern
- The Observer Pattern
- The Adapter Pattern
- The Composite Pattern
- The Iterator Pattern
- The Builder Pattern
- Fallen Patterns
  - The Singleton Pattern
  - The Visitor Pattern
- **Command Pattern**



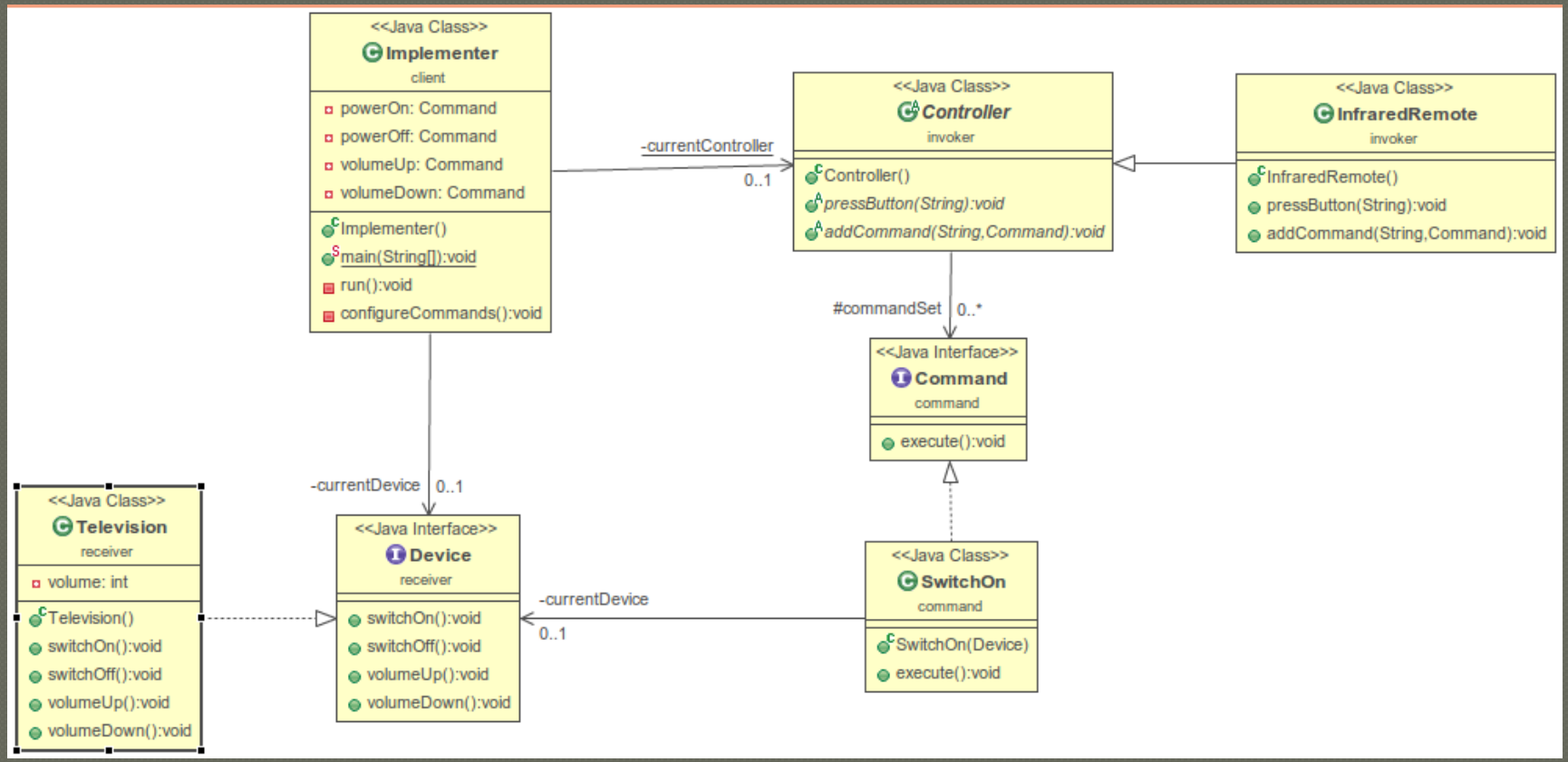
Patterns So Far

- Strategy
- Factory method
- Abstract Factory
- State
- Observer
- Adapter
- Composite
- Iterator
- Builder
- Singleton
- Visitor

The Last Pattern: Command

- Complex, but valuable to a wide variety of domains
- Encapsulates an action, to be performed at a later time
- Turns a "verb" into a "noun"
- Can be combined with the "memento" pattern to save state

Example: TV Remote



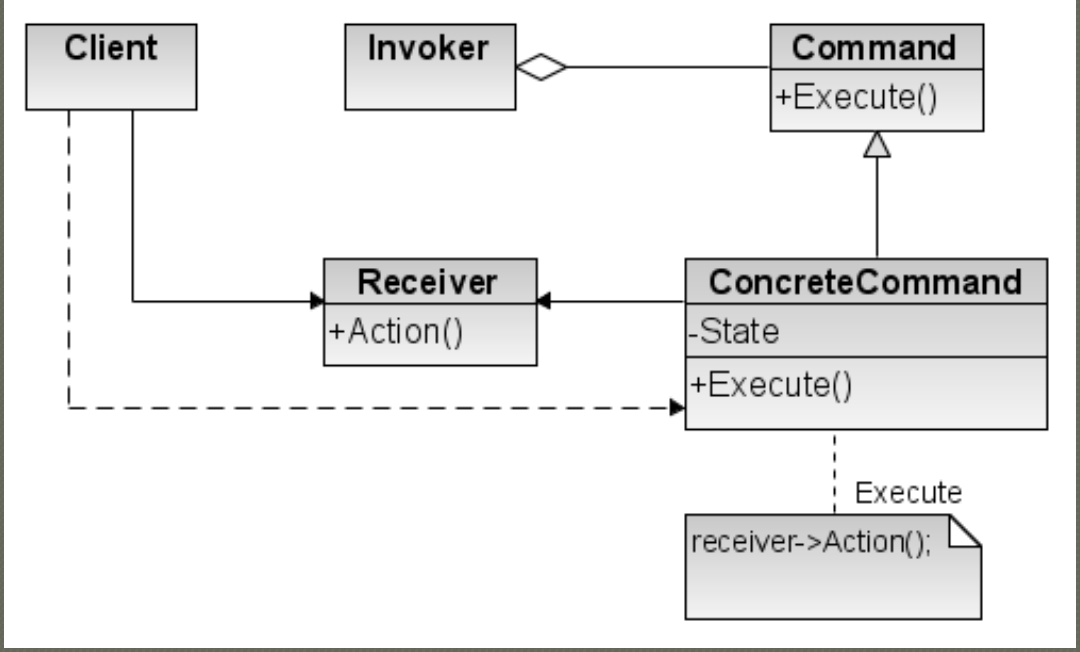
Code

```
interface Command {
  void Execute();
}

class SwitchOn implements Command {
  private Device currentDevice;

  public SwitchOn( Device d ) {
     currentDevice = d;
  }

  public void Execute() {
     currentDevice.switchOn();
  }
}
```

Why Bother?

- We can implement new kinds of controller (Bluetooth, WiFi, …)
- We can add functions and remap buttons
- We can now store a dictionary to do this if we want
- We can implement a new kind of device (like a stereo) and have the controller still work

Generic UML



- Note: the aggregation relationship between Invoker and Command can be delegation instead

Important Parts?

- Command: abstract command with "execute" method
- Invoker: the class/method calling "execute" on the command (the TV remote)
- ConcreteCommand: the implementor of the actual execute method. (PowerOn)
- Receiver: the object which is modified by "execute" (the TV)

How is Command Useful?

- You can buffer lists of commands to be sent via IP
- You can store lists of commands to a file
- You can implement undo/redo systems
- If commands don't conflict, they can be processed in parallel
- More examples: https://en.wikipedia.org/wiki/Command_pattern

More Examples

Online multiplayer: commands represent changes in game-state. They are buffered and sent to the server where they are executed.

Crypto-currencies: the blockchain is essentially a linked list of command blocks. Esp. currencies like Ethereum.

Best Example

- Redux uses command for state management: https://redux.js.org/

Alternatives to Command

- Command is not intrinsically object oriented, and can be implemented using switch-case statements:
  - The command is identified by an integer/enum
  - The state of the command can be stored as an arbitrary pointer or byte data
- Functionally: commands can be represented as lambdas