


13 JUNE 2018 / DESIGN PATTERNS

# Swift command design pattern

This time I'm going to show you a behavioral pattern. Here is a little example of the command design patten written in Swift.

The command pattern can be handy if you'd like to provide a common interface for different actions that will be executed later in time. Usually it's an object that encapsulates all the information needed to run the underlying action properly.

Commands are often used to handle user interface actions, create undo managers, or manage transactions. Let's see a command pattern implementation in Swift by creating a command line argument handler with emojis. 📄

Command.swift1.75 KB on 

```
1  #!/usr/bin/env swift
2
3  import Foundation
4
5  protocol Command {
6      func execute()
7  }
8
9  class HelpCommand: Command {
10
11      func execute() {
12          Help().info()
13      }
14  }
15
16  class Help {
17
18      func info() {
19          print("""
20
21              🍏 Commander 🍏
22              v1.0
23
24              Available commands:
25
26              🚩 help      This command
27              🚩 ls        List documents
28
29              Bye! 🙋
30
31              """)
32      }
33  }
34
35  class ListCommand: Command {
36
37      func execute() {
38          List().homeDirectoryContents()
39      }
40  }
41
42  class List {
43
44      func homeDirectoryContents() {
45          let fileManager = FileManager.default
46          guard let documentsURL = fileManager.urls(for: .documentDirectory, in: .userDomainMask).first else {
47              print("Could not open documents directory")
48              exit(-1)
49          }
50          do {
51              let fileURLs = try fileManager.contentsOfDirectory(at: documentsURL, includingPropertiesForKeys: nil, options: [])
52              print("\n\ 📄 Listing documents directory:\n")
53              print(fileURLs.map { "\t\t📄 " + $0.lastPathComponent }.joined(separator: "\n\n") + "\n")
54          } catch {
55              print(error.localizedDescription)
56              exit(-1)
57          }
58      }
59  }
60
61
62
63  class App {
64
65      var commands: [String:Command] = [:]
66
67      init() {
68          self.commands["help"] = HelpCommand()
69          self.commands["ls"] = ListCommand()
70      }
71
72      func run() {
73          let arguments = CommandLine.arguments[1...]
74
75          guard let key = arguments.first, self.commands[key] != nil else {
76              print("Usage: ./command.swift [\self.commands.keys.joined(separator: "|")]")
77              exit(-1)
78          }
79
80          self.commands[key]!.execute()
81      }
82  }
83
84  App().run()
```

If you save this file, can run it by simply typing `./file-name.swift` from a terminal window. The Swift compiler will take care of the rest. 🦋

## Real world use cases for the command design pattern:

- various button actions
- collection / table view selection actions
- navigating between controllers
- history management / undo manager
- transactional behavior
- progress management
- wizards

As you can see this pattern can be applied in multiple areas. Apple even made a specific class for this purpose called [NSInvocation](#), but unfortunately it's not available in Swift, due to it's dynamic behavior. That's not a big deal, you can always make your own protocol & implementation, in most cases you just need one extra class that wraps the underlying command logic. 😊

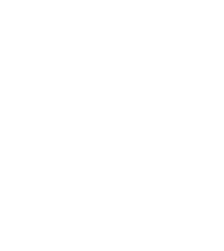
## External sources

- [Command pattern](#)
- [Design Patterns in Swift: Command Pattern](#)
- [Swift World: Design Patterns—Command](#)
- [Command Patterns and UICollectionView](#)

## Subscribe to The.Swift.Dev.

Get the latest posts delivered right to your inbox

[Subscribe](#)



**Tibor Bödecs**

Read [more posts](#) by this author.

[Read More](#)

— The.Swift.Dev. —

## Design patterns



[Lazy initialization in Swift](#)

[Swift object pool design pattern](#)

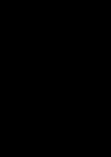
[Swift facade design pattern](#)

[See all 18 posts →](#)

IOS

## Mastering iOS auto layout anchors programmatically from Swift

Looking for best practices of using layout anchors? Let's learn how to use the iOS autolayout system in the proper way using Swift.

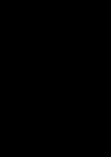


4 MIN READ

DESIGN PATTERNS

## Swift prototype design pattern

The prototype design pattern is used to create clones of a base object, so let's see some practical examples written in Swift.



1 MIN READ