**Examples: Switch .. Case**

//Create a new playground called Switch and set up a switch.

**//Example:01 Your first switch**

import Cocoa

var statusCode: Int = 404

var errorString: String

switch statusCode {

case 400:

errorString = "Bad request"

case 401:

errorString = "Unauthorized"

case 403:

errorString = "Forbidden"

case 404:

errorString = "Not found"

default:

errorString = "None"

}

=================

**//Example 02: Switch cases can have multiple values**

import Cocoa

var statusCode: Int = 404

var errorString: String = "The request failed with the error:"

switch statusCode {

case 400, 401, 403, 404:

errorString = "There was something wrong with the request."

fallthrough

default:

errorString += " Please review the request and try again."

}

**Note:**

Here, fallthrough tells the switch statement to “fall through” the bottom of a case to the next one. If a matching case has a fallthrough control transfer statement at the end of it, it will first execute its code, and then transfer control to the case immediately below. That case will execute its code – whether or not it matches the value being checked against. If it also has a fallthrough statement at the end, it will hand off control to the case below, and so on. fallthrough statements allow you to enter a case and execute its code without having to match against it.

In this example, the fallthrough statement means that even though the first case matches, the switch statement does not stop. It proceeds to the default case. Without the fallthrough keyword, the switch statement would have ended execution after the first match. The use of fallthrough in this example allows you to build up errorString without having to use strange logic that would guarantee that the comparison value matched all of the cases of interest.

================

Refer the screen shots below.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**>>> Ranges:**

**//Example 03: Switch cases can have single values, multiple values, or ranges of values**

import Cocoa

var statusCode: Int = 404

switch statusCode {

case 100, 101:

errorString += " Informational, 1xx."

case 204:

errorString += " Successful but no content, 204."

case 300...307:

errorString += " Redirection, 3xx."

case 400...417:

errorString += " Client error, 4xx."

case 500...505:

errorString += " Server error, 5xx."

default:

errorString = "Unknown. Please review the request and try again."

}

**//Example 04: Using value binding**

switch statusCode {

case 100, 101:

errorString += " Informational, \(statusCode)."

case 204:

errorString += " Successful but no content, 204."

case 300...307:

errorString += " Redirection, \(statusCode)."

case 400...417:

errorString += " Client error, \(statusCode)."

case 500...505:

errorString += " Server error, \(statusCode)."

case let unknownCode:

errorString = "\(unknownCode) is not a known error code."

default:

errorString = "Unknown. Please review the request and try again." }

**>>> where clauses**

**//Example 05: Using where to create a filter**

import Cocoa

//var statusCode: Int = 404

var statusCode: Int = 204

var errorString: String = "The request failed with the error:"

switch statusCode {

case 100, 101:

errorString += " Informational, \(statusCode)."

case 204:

errorString += " Successful but no content, 204."

case 300...307:

errorString += " Redirection, \(statusCode)."

case 400...417:

errorString += " Client error, \(statusCode)."

case 500...505:

errorString += " Server error, \(statusCode)."

case let unknownCode where (unknownCode >= 200 && unknownCode < 300)

|| unknownCode > 505:

errorString = "\(unknownCode) is not a known error code."

default:

//errorString = "\(statusCode) is not a known error code."

errorString = "Unexpected error encountered."

}