CSC 491 / 391
Mobile Application Development for iOS II

Prof. Xiaoping Jia
School of Computing, CDM
DePaul University
xjia@cdm.depaul.edu

1

Pattern Matching in Switch Statement

2

Patterns in Switch Statement – Patterns Based on Types

```
let numbers: [Any] = [ 0, 0.0, 2018, Double.pi, -Double.pi, Float.pi ]
for n in numbers {
    switch n {
    case 0 as Int:
        print("zero as an Int")
    case 0 as Double:
        print("zero as a Double")
    case let i as Int:
        print("an integer: \(i)")
    case let d as Double where d > 0:
        print("a positive double \(d)")
    case is Double:
        print("a negative double \(n)")
    default:
        print("some other number \(n)")
    }
}
```

Output:
zero as an Int
zero as a Double
an integer: 2018
a positive double 3.14159265358979
a negative double -3.14159265358979
some other number 3.14159

10

Patterns in Switch Statement – Patterns Based on Values

```
let points = [ (0, 0), (1, 1), (4, 0), (0, -5),
               (-1, -1), (-3, 3), (-2, 1)  ]
for p in points {
    switch p {
    case (0, 0):
        print("(0, 0) is at the origin")
    case (_, 0):
        print("(\(p.0), 0) is on the x-axis")
    case (0, _):
        print("(0, \(p.1)) is on the y-axis")
    case (-2...2, -2...2):
        print("(\(p.0), \(p.1)) is inside the box …")
    default:
        print("(\(p.0), \(p.1)) is outside of the box …")
    }
}
```

Output:
(0, 0) is at the origin
(1, 1) is inside the box …
(4, 0) is on the x-axis
(0, -5) is on the y-axis
(-1, -1) is inside the box …
(-3, 3) is outside of the box …
(-2, 1) is inside the box …

19

Patterns in Switch Statement – Patterns with Value Binding

```
let points = [ (0, 0), (1, 1), (4, 0), (0, -5), (-1, -1), (-3, 3), (-2, 1)  ]
for p in points {
    switch p {
    case (let x, 0):
        print("on the x-axis with an x value of \(x)")
    case (0, let y):
        print("on the y-axis with a y value of \(y)")
    case let (x, y) where x == y:
        print("(\(x), \(y)) is on the line x == y")
    case let (x, y) where x == -y:
        print("(\(x), \(y)) is on the line x == -y")
    case let (x, y):
        print("somewhere else at (\(x), \(y))")
    }
}
```

Output:
on the x-axis with an x value of 0
(1, 1) is on the line x == y
on the x-axis with an x value of 4
on the y-axis with a y value of -5
(-1, -1) is on the line x == y
(-3, 3) is on the line x == -y
somewhere else at (-2, 1)

28

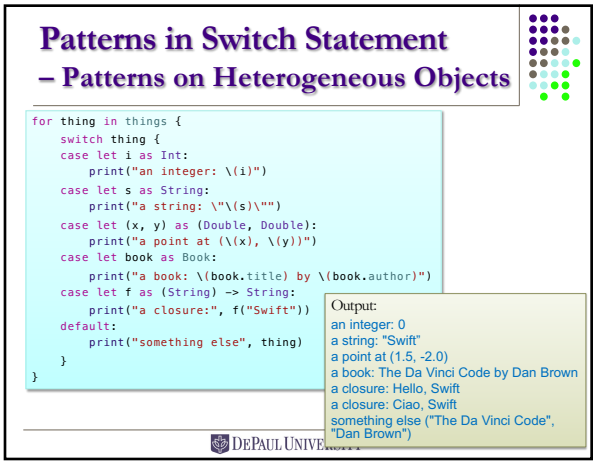
Patterns in Switch Statement – Patterns on Heterogeneous Objects

```
struct Book {
    var title: String
    var author: String
}

func hello(name: String) -> String {
    return "Hello, \(name)"
}

let things: [Any] = [
    0, "Swift", (1.5, -2.0),
    Book(title: "The Da Vinci Code", author: "Dan Brown"),
    hello,
    { (name: String) -> String in "Ciao, \(name)" },
    (title: "The Da Vinci Code", author: "Dan Brown")
]
```


29


Patterns in Switch Statement
– Patterns on Heterogeneous Objects
for thing in things {
switch thing {
case let i as Int:
print("an integer: \(i)")
case let s as String:
print("a string: \"\(s)\"")
case let (x, y) as (Double, Double):
print("a point at (\(x), \(y))")
case let book as Book:
print("a book: \(book.title) by \(book.author)")
case let f as (String) -> String:
print("a closure:", f("Swift"))
default:
print("something else", thing)
}
}
Output:
an integer: 0
a string: "Swift"
a point at (1.5, -2.0)
a book: The Da Vinci Code by Dan Brown
a closure: Hello, Swift
a closure: Ciao, Swift
something else ("The Da Vinci Code", "Dan Brown")
DePaul Univ

38