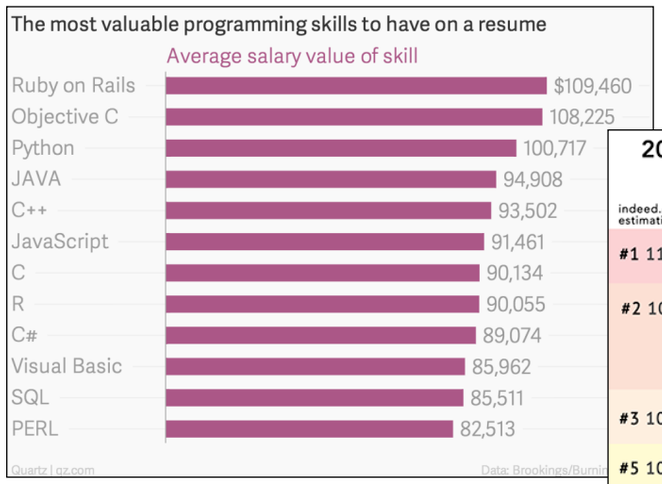
CS 125 - Lecture 41

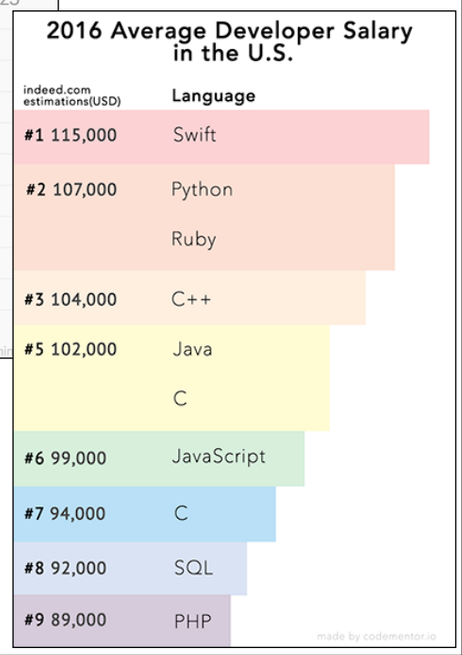
Objectives: model-view-controller (MVC) paradigm; iOS; Swift

Up next: all MPs regraded and TC final grading next Wednesday

Model-View-Controller is valuable…



from 2014



Did you know?

Chris Lattner (http://en.wikipedia.org/wiki/Chris_Lattner), the primary developer of the LLVM compiler (Objective-C) and of the Swift programming language, graduated from UIUC just ten years ago.

Model–view–controller (MVC) implementation

The model is the data, the view is the window on the screen, and the controller is the glue between the two.

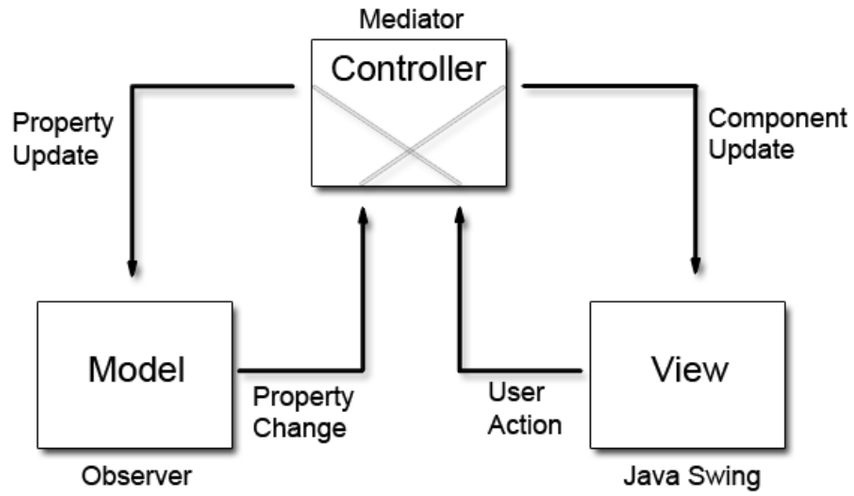
A **model** is an object representing data , e.g. a database table.

A **view** is some form of visualization of the state of the model.

A **controller** facilities changes to the state of the model and/or the views.

Important: In **MVC**, the model never interacts directly with the view, or vice versa.

Model–view–controller (**MVC**) is a software architectural pattern (design pattern) for implementing applications. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. In this pattern, the views never own the data they display.



iOS resources: Today's Demo - Gilmore Girls Fan Club App

Try it yourself: campus Mac labs with Xcode installed: https://www.cites.illinois.edu/ics/labs.html | CocoaNuts UIUC student organization http://www.cocoanutsdev.com

Swift Book URL: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/

Many excellent web resources : google "swift tutorials" "swift iOS tutorials" | Stanford iTunesU course with Paul Hegarty

Swift Notes: To declare a variable in Swift you use the keyword *var*

```
var number = 1
```

The main difference with java is that we did not have to define the type, this is because Swift uses **type inference** and is able to understand that the type of the variable is an integer because its initial value is an integer.

However we are also able to define the type if we wish, like so:

```
var number: Int = 1
```

Let's declare a string variable in Swift:

```
var language = “Swift"
```

How does this compare with java?

```
String language = “Swift";
```

Note: Semicolons are not necessary in Swift.

Constants (immutables) are also supported in Swift and can be declared using the keyword *let*

```
let language = "swift"
```

or

```
let language: String = "Swift"
```

Working with variables in Swift is no different from other languages however it's worth noting that it has an easy method for letting you add values in strings such as

```
var designers = 4
var developers = 4

var teamSize = "The team has \(designers + developers) members”
// creates the string containing “The team has 8 members”
```

Conditionals

```
if city == "MEL" {
    print("Melbourne")
} else if city == "SYD" {
    print("Sydney")
} else {
    print("Perth")
}
```

Functions (methods in java)

```
func sayName(name: String, lastName: String) {
  print("\(name) \(lastname)")
}
```

Arrays

In Swift you create arrays and dictionaries using brackets [ ] and access their elements by writing the index or key in brackets.

```
var arr = ["first" , "second"]
```

You can get an item from the array using the index

```
var order = arr[17]
```

and set the value using

```
arr[0] = “third"
```

You can enumerate the array using a *for item* loop

```
for item in arr {
    print(item) // do something
}
```

To add another item to the array you can use the += operator

```
arr += "fourth"
```

Dictionaries

You can declare dictionaries in Swift by defining key-value pairs. When declaring an empty dictionary you have to define the type of the key and the type of the value.

```
var dict = [String: String]()
```

You can declare and assign values

```
var cityDict = [“MEL”:”Melbourne”,”SYD”:"Sydney","CMI":"Champaign"]
```

To get an item from a Dictionary use

```
var entry = dict[“MEL”]
```

To set or add an item in the Dictionary use

```
dict["PER"] = “Perth"
```

Finally to iterate over a dictionary you can use

```
for (cityCode, cityName) in dict {
    print("\(cityCode) : \(cityName)")
}
```

Iterating

```
for var number = 1; number <= 5; number++ {  //deprecated!
for i in 1 … 5 {        // OR (for i in 1..<6)
   //do something
}
for city in citiesArray {
  print(city)
}
```