CS 220 Lecture 2 9-15-2015

Programming Methodology

03: Classes, Object, and Git

* Scala Maps and Tuples
  + A classic programmer’s saying is,
    - If you can only have one data structure, make a hash table
* Constructing a map
  + You can construct a map as
    - val scores = Map(“Alice” -> 10, “Bob” -> 3, “Cindy” -> 8)
  + This constructs an immutable Map[String, Int]
  + If you want a mutable map, use:
    - val scores = scala.collection.mutable.Map Map(“Alice” -> 10, “Bob” -> 3, “Cindy” -> 8)
  + Of course, you can import scala.collection.mutable.Map:
    - val scores = Map(“Alice” -> 10, “Bob” -> 3, “Cindy” -> 8)
* Constructing pairs
  + The -> operator makes a pair, the value of
    - “Alice” -> 10
    - is
    - (“Alice”, 10)
  + You could have similarly defined the map as:
    - val scores = Map((“Alice”,10), (“Bob”,3) ,(“Cindy”,8))
* Accessing Map Values
  + The Analogy between functions and maps is particularly close because you use the () notation to look up key values
    - val obsScore = scores(“Bob”) // Like scores.get(“Bob”) in java
  + If the map doesn’t contain a value for the requested key, an exception is thrown
  + To check whether there is a key with the given value do this:
    - val bobsScore = if (scores.contains(“Bob”)) scores(“Bob”) else 0
  + Since this is common there is a short cut that you see often:
    - val bobsScore = scores.getOrElse(“Bob”,0)
* Option Value
  + The call map.get(key) returrns an Option object that is either Some(value for key) or None.
    - val v = scores.get(“Bob”).getOrElse(0)
  + We discuss this further when we look at functional programming techniques and how we handle exceptional control flow.
* Updating Mutable Map Values
  + In a mutable map, you can update a map value, or add a new one, with a () to the left of an = sign
  + Check online
* Immutable Map Values
  + You cant update an immutable map, but you can do this to create a newly constructed map
    - val newScores= scores + (“Bib” -> 10, “Fred” -> 7)
    - The newScores map contains the same associations as scores, except that “Bob” has been updated and “Fred” added.
    - Instead of saving the result as a new value, you can use a var:
      * var scores = …
      * scores = scores + (“Bob” -> 10, “Fred” -> 7)
* Iterating over maps
  + The following amazingly simple loop iterates over all key/value pairs of a map
    - check back
* Tuples
  + A tuple value is formed by enclosing individual values in parenthesis
    - (1, 3.14, “Fred”)
    - is a tuple of type Tuple3[Int, Double, java.lang.String]
    - Also written as
      * (Int, Double, java.lang.String)
  + If you have a tuple
    - val t (1,3.14, “Fred”)
    - then you can access its component with the methods \_1, \_2, \_3, …
      * val second = t.\_2 // Sets second to 3.14
* Tuples and Pattern Matchin
  + You can use pattern matching, also known as destructuring assignment, to easily assign tuple contents to variables:
    - val (first, second, third) = t
  + If you do not need the third part, you can leave it out:
    - val (first, second, \_) = t
* IntelliJ Practice
* Simple Classes
  + In their simplest form, Scala classes look like Java
  + class Counter
    - private var value = 0
    - def increment
  + To use this class, you must construct objects and invoke methods.
    - val myCounter = new Counter // Or new Counter()
    - myCounter
* Primary Constructor
  + Every class has a primary constructors that I part of the class def. The arguments are placed right after the class name
    - Class Person(val name: String, val age: Int) {
      * //Parameters of primary constructor in (…)
      * …
    - }
    - The parameter sof the primary constructor become fields that are initialized with the construction arguments.