Scala: Language Explore

Zoe Nacol

Kyle Dymowski

Charles Tandy

Anthony Nguyen

December 7, 2015

1 Introduction to Scala

Scala is both an object-oriented and functional language [1]. A portmanteau on "scalable" and "language", Scala was created based upon criticism of Java. Running on Java Virtual Machine, language interoperability can be seen between Java and Scala. Both Java and Scala classes and libraries can be mixed. Despite the conventional syntax and it being based off of Java, Scala supports first class functions, immutable data structures, and a general preference of immutability over mutation.

2 Scala Features

2.1 Concepts Learned in Class

"At the root, the language's scalability is the result of a careful integration of object-oriented and functional language concepts." Scala can be treated as both a scripting language and "workhorse" language. Because of this, Scala supports many useful features. Many of which were covered in CSCI 400.

Pattern Matching

Type Inference

Higher Order Functions

other: functions are objects, lazy evaluation, currying

2.2 Concepts Not Learned in Class

Value Classes and Universal Traits In Java there is no difference between the integer type, it can be used through the program for counters, an entity identifier or a number in an arithmetic expression. In most cases these Integers have nothing to do with each other. It is a common mistake to compare them do arithmetic operations on them, or pass the wrong one into a function as a parameter. A value class allows a new level of type checking that looks at the code rather than relying on the person doing the coding. A value class works like a wrapper class that gives the ability to enrich objects with a precise type without allocation overhead.

However, limitations stop values classes from being universally used:

- primary constructor for the value class must have exactly one val parameter whose type is not a class [3]
- class may not have specialized type parameters.
- class may not have nested or local classes, traits, or objects
- class may not define a equals or hashCode method.
- class must be a top-level class or a member of a statically accessible object
- class can only have defs as members. In particular, it cannot have lazy vError! Hyperlink reference not valid.als, vars, or vals as members.
- class cannot be extended by another class.

Implicit Classes Scala allows for extension methods to be added onto another type by introducing implicit classes. The main purpose of an implicit class is to make the process more concise. The method definitions must be declared where the scope allows for method definitions. This works by the implicit method mimicking the constructor of the class [4].

String Interpolation Scala makes use of string interpolation to create strings of data. This process is used to directly embed variables into string literals. It makes use of three methods of string interpolation. This first is s where string literal allows the usage of variables directly in the string. Second, is f which allows the creation of simple formatted strings similar to printf in PHP. Third, is raw which is essentially the same as s, but with no escaping of literals in the string [5].

Type System This should be moved up to concepts learned in class.

Raw Strings

Flexibility

Type Enrichment

Concurrency

3 Demo Program

```
val height = 1.9d
val name = "James"
println(f"$name\%s is $height\%2.2f meters tall") // James is 1.90 meters
    tall
```

4 References

1. http://www.scala-lang.org/what-is-scala.html
2. https://en.wikipedia.org/wiki/Scala_%28programming_language%29
3. https://ivanyu.me/blog/2014/12/14/value-classes-in-scala/
4. http://docs.scala-lang.org/overviews/core/implicit-classes.html
5. http://docs.scala-lang.org/overviews/core/string-interpolation.html