Richard Hsu 3-6-16
CS6221 – Project 1

Project Idea: Chatbot

The project idea that I selected was the Chatbot. After much research, I became interested in developing a program that included aspects of artificial intelligence, which is a field that I am very interested in. ELIZA is an example of one of the very first Natural Language Processing Systems designed by Joseph Weigenbaum in 1966 in order to mimic psychotherapy. In the program that he wrote, Weigenbaum used an algorithm to parse certain words in a user's input and then uses it to generate logical responses that can move the conversation forward. The end goal was to create the illusion that ELIZA actually understood what the user was telling it even though the program is merely following a set of instructions to generate meaningful responses.

Language Selected: Scala

Language Overview

Scala or "Scalable Language" is a popular language because it can grow with the size of one's project. Many large companies including Twitter, Linkedin, and even Intel use it. One reason that it is so popular is the fact that it is both object-oriented and functional.

The language is object-oriented in the sense that every value is an object and every function can be thought of as methods. Even functions themselves can be assigned to variables in Scala.

In addition to being object-oriented, Scala is also functional in that you can write everything inside of functions and there is a large library of immutable data structures. More importantly, unlike other functional programming languages like Haskell, Scala provides programmers with a way to easily transition to functional programming, a very different way to program than imperative programming.

Moreover, Scala can also run on the java virtual machine or JVM and this allows Scala developers to also utilize classes in the Java language. It is also easy to find IDEs that will work with Scala. Some of these include Eclipse, IntelliJ, and Netbeans.

Why did you select it?

Originally, I started my project in Haskell. Learning Haskell was fun but it was challenging to think about my problem in terms of different functions. I would try to call various functions from my main function and the ghci compiler for Haskell would always be upset. Then, I found out that Scala is both object-oriented and functional so it offers programmers a better way to transition into functional programming if he or she chooses to do so. However, in the end, my program is not a good reflection of functional programming since including AI components meant that there is a lot of string

manipulation and parsing which would be difficult to accomplish with a functional programming language like Haskell.

What did you learn about the language?

I learned a lot about the nature of the language. Here are just some of the highlights.

Data Types and Variables

Scala has the same data types as Java including byte, short, int, long, float, double, char, String, and Boolean.

There are two ways to declare variables in Scala. First, you can use the var keyword if the variable is mutable or able to change such as:

var myVariable = “hello”

One the other hand, if the value of the variable is a constant, you can use the val keyword like so:

val myConstant = “constant hello”

Scala supports type inference, which means that the Scala compiler can figure out the type of the variable based on the value that you assign it.

var myVar = 10
var myVal = “Hello, world”

In this case, the compiler knows that myVar is an int and myVal is a string.

Control Structures

If/else conditions in Scala are very similar to those in Java.

Here is the syntax:

```
if (Boolean_expression)
{
        …
}
else
{
        …
}
```

The three types of loops (while, do… while, for) from Java also exist for Scala.

Note that in Scala, when the execution leaves a scope, all the objects that were created in that scope are destroyed. For this reason, Scala does **not** support break or continue statements like Java. In Scala 2.8, there is a way to break a loop with the following construct:

```
import scala.util.control._
val loop = new Breaks

loop.breakable
{
        for (…)
        {
                loop.break
        }
}
```

Scala does have both functions and methods. A method is a function defined inside the member of an object. Functions are objects that can be assigned to variables. Here is the function declaration signature and form:

```
def functionName ([list of parameters]) : [return type] =
{
        function body
        return [expr]
}
```

Functions that don't return anything can return a Unit that is equal to void. Here is an example:

```
def printMe( ) : Unit =
{
        println("Hello, Scala!")
}
```

Classes/Encapsulation

Classes or blueprints for objects can also be created in Scala and instantiated with the keyword "new." Here is an example of a class for a Point on a graph.

```
class Point(val xc: Int, val yx: Int)
{
        var x: Int = xc
        var y: Int = yc
        def move(dx: Int, dy: Int)
        {
```

```
                x = x + dx
                y = y + dy
                println("Point x location : " + x)
                println("Point y location : " + y)
        }
}
```

Scala also supports inheritance with the "extends" keyword. Moreover, like Java, Scala allows the inheritance of one class only. Above all, Scala is actually even more object-oriented than Java because there is no notion of methods or variables being "static." Instead, Scala supports singleton objects which is a class that can have just one instance.

Unlike Java, Scala also bundles and methods and variables together to form traits. Traits provide the encapsulation for similar methods and a class can actually use any number of traits. Traits are declared just like classes but they also include the "trait" keyword:

```
trait Equal
{
        def isEqual(x: Any): Boolean
        def isNotEqual(x: Any): Boolean = !isEqual(x)
}
```

Special Features/Interesting Features

One thing that makes Scala unique is that it includes a rich set of collections. These collections can be either mutable, meaning that the contents can change. On the other hand, contents inside an immutable collection cannot change.

Scala Lists are very similar to arrays except that they are immutable. This means that the elements cannot be changed by assignment. Furthermore, Scala lists represent a linked list while arrays are actually simply flat. Here is a list of strings:

```
val fruit = List("apples", "oranges", "pears")
```

Scala Tuples combines various elements together so they can be passed around. Here is an example:

```
val t = (1, "hello")
```

This is incredibly useful if you need to bundle several values such as the coordinates of a point.

Another interesting feature in Scala is pattern matching. It involves defining several cases to match on and then a default case, which catches all the other cases. Here is an example:

```
def matchTest(x: Int) : String = x match
{
        case 1 => “one”
        case 2 => “two”
        case _ => “many”
}
```

Here we match on the x being the int 1, or int 2. If it is another int other than those two, we catch those with “many.”

Development Environments

I am on a mac operating system and I installed Scala using instructions from the following link:

http://www.scala-lang.org/download/

The site lists two different ways of using Scala. First, there is browser-based command line tool called “Lightbend Activator.”

The other option would be to use a Scala IDE, which includes Eclipse, IntelliJ with Scala plugin, or Netbeans with Scala plugin.

For this project, I used a sample text editor, Sublime to do most of my coding.

For windows users, the instructions are similar.

How I Use the Language

First, it was difficult for me to initialize a 2-dimensional array. Therefore, I decided to make a list with many lists inside of it.

Second, since Scala did not support breaks, I had to use a special construct to build that functionality. It involved using the breakable keyword and then wrapping the loop inside of it.

Third, there were many manipulations I had to do with the user’s input string. Fortunately, the Scala String library was very robust in that I was able to use a method like split to get the tokens inside of the string. I was also able to do pattern matching with the findFirstIn function to check if the keyword was inside the string the user was giving to the program. Lastly, I was able to get indexes of strings with the indexOfSlice method and also a substring of the original string with the substring method.

Another major hurdle that I overcame was the fact that users would type in things and Eliza would need access to the second to last input in order to make the exchange even more meaningful. To do this, I manipulated a counter for the number of exchanges to

store that value for processing. Once the counter reaches 2 or higher, we store the second to last input so that we can use it when necessary.

I included screenshots of two runs of my program in action. Please run it to see some interesting outputs!

References

Here are some of the references that I used for this project:

The Official Scala Website: http://www.scala-lang.org/

Scala/TutorialsPoint: http://www.tutorialspoint.com/scala/index.htm