Programming Languages and Paradigms

COMP 302, Winter 2018

Assignment 3

Due date: Monday, March 26, 2018
6pm

This assignment focuses on building an evaluator for the WML language within Scala. Your code must run without error or modification using Scala 2.12.

Note that you must follow the given naming, input, and output requirements precisely. All code should be well-commented, in a professional style, with appropriate variables names, indenting (uses spaces and avoid tabs), etc. **The onus is on you to ensure your code is clear and readable. Marks will be very generously deducted for bad style, lack of clarity, or failing to follow the required instructions.**

Your code should endeavour to follow a pure functional programming style. In particular, and unless specifically stated otherwise, all data types must be *immutable*, and *data may not be modified once assigned or bound*. Note that this means you may not use `var` declarations, `while` or `do`-loops, `ArrayBuffers` or other mutable structures, nor may you reassign `Array` element values after creation.

The goal of this assignment is to be able to successfully evaluate the WML language described in class. To do this you will need a parser that generates an AST from an input WML program. You can base your design on your own code from assignment 2, or use the parser solution given for assignment 2. You can assume all WML input is both syntactically correct and correctly executable.

1. A *pretty printer* receives an AST as input, and prints out a nicely formatted version of the program the AST represents. In most cases it aims to re-present the input program in a nicely formatted, readable form. Here, you will instead reformat the input to present it in its most compact form. **8**

   Define a function `prettyPrint` that accepts a root `ASTNode` and generates a `String` representation of the program. Your code should print out the WML code, eliminating all unnecessary whitespace in the program.

   Recall that in general whitespace found around invocation names and arguments, definition names and parameter names, and around variable (tvar) names and optional parts is discarded, and is only required when necessary to disambiguate tokens.

   Note that it is critical that your pretty printer only modifies whitespace, and does not change the meaning of the code. Thus, given an AST rooted at `n` and a WML parser `p`, it must be the case that:

   ```
   prettyPrint(n) == prettyPrint(p.parseAll(p.program,prettyPrint(n)))
   ```

   Provide a program, `q1.scala` that can accept two arguments, `-p` *`filename`* and if so prints out the pretty-printed form of the WML code in *filename*.

2. Define an `eval` function. This function should accept an `ASTNode` and an `Environment`, and recursively evaluate the node, generating a `String` and function pair.

   The evaluation behaviour for all nodes was described in class. Your design should respect static scoping rules, allowing access to all parent (grand-parent, etc) environment symbols, modulo shadowing.

   In general, your implementation should:

   (a) Correctly evaluate all strings that are not WML invocations, parameters/variables, or definitions. **2**

   (b) Ignore whitespace in template names, arguments and parameters, and variable references. **3**

   (c) Properly evaluate definitions, including names and parameter declarations, and construction of an appropriate binding. **6**

(d) Properly evaluate invocations, including argument evaluation, template-name lookup (following static binding), creating correct parameter name bindings, and evaluating the corresponding body. **5**

(e) Properly evaluate parameter references, including lookup (following static binding) and returning an evaluation of the optional/default value if not defined. **5**

(f) Properly handle passing of (anonymous) functions as closures, allowing them to be returned from invocations, bound to parameters, and invoked in invocations. **5**

Provide a program, `q2.scala` that receives the name of a file containing a WML program, and then parses and evaluates the program, printing out to the console the resulting output string.

3. Modify your evaluator to recognize `#if`, `#ifeq`, and `#expr` as special functions which perform conditional testing (with optional else parts, and incorporating lazy evaluation of the then/else) and numerical expression evaluation. **6**

   Provide a file `q3.scala` with your full implementation. As with the previous question, it accepts the name of a file containing a WML program, and then parses and evaluates the program, printing out to the console the resulting output string.

What to hand in

Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**. Assignments must be submitted on the due date **before 6pm**.

This assignment is worth 10% of your final grade. **40**