Programming Languages and Paradigms

COMP 302, Fall 2016

Assignment 4

Due date: Monday, December 5, 2016
6pm

In the first part of this assignment you will apply WML to create a (limited) object system in WML using only functional features of the language. Use either your own solution to assignment 3, or the supplied solution as an interpreter for testing.

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 or lack of clarity.**

1. To model buttons, as used in clothing, we could start from the following small hierarchy,

   - *material*. All materials have an attribute of *cost*, and may be subdivided into the following:
     - *plastic*. Plastic materials have a property of being composed of different *chemical*s.
     - *metal*. Metals have a property of being *ferrous* or not.
   - *attachment*. There are different ways of attaching buttons to clothing:
     - *holed*. Different *number*s of holes may be used, such as 2 or 4.
     - *shank*. Buttons may also be attached through a projecting base component, and have a boolean property *self-shank* to indicate whether it is integral or a separate component.

   (a) Define a set of WML templates for modelling this hierarchy of objects (note that there are two distinct hierarchies here). You should define WML templates for constructing *material*, *plastic*, *metal*, *attachment*, *holed*, and *shank* objects. Each constructor template accepts arguments to define its specific properties, and uses default values (your choice) for parent parameters. **10**

   Each constructor returns a function which will act as the "object" constructed. This object can be invoked, passing in the name of any functionality it or a parent type provides, and should execute that function. As basic behaviours, provide "getter" functionality for each property that object may have (e.g., *getstrength*, *getferrous*, etc). For example,

   ```
   {:talkaboutplastic|p|
      A {{ {{{p}}}|getchemical}} button costs {{ {{{p}}}|getcost}}:}.
   {{talkaboutplastic|{{plastic|polyurethane}}}}
   ```

   emits the string "A polyurethane button costs X.", where X is whatever you use as a default cost.

   Your design should model inheritance: an object should only have actual getters for its own, unique properties, and should delegate requests for inherited properties to a parent object.

   Each object should also respond to a *tostring* request, returning a string describing that all object properties (including any inherited ones), formatted as you would a JavaScript object definition. Note that this function is overridden by each sub-type in your hierarchy.

   (b) Actual buttons combine features in the two heirarchies, and also have their own property, the *ligne number* giving the diameter of the button. Define a *button* constructor which is based on 5 arguments, a material-type (`plastic` or `metal`), an attachment-type (`holed`, or `shank`), a *ligne number*, and the material and attachment-specific arguments. Your button objects should respond to messages appropriate to its composition; e.g., metal buttons understand *getferrous*, while plastic buttons do not. **5**

   (c) A button collection will be held in a cons-list. Define WML templates for `cons`, `car`, and `cdr`. **3**

2. Now you need to display your button collection. Define a template, **6**

   `{{buttoncollection|buttons|title|filter}}`

   that produces a filtered and formatted HTML table of a given cons-list of buttons.

   The first row should consist of a table header containing the `title`. Each row consistent of a single table cell that displays one button, as per its `tostring` function.

   The `filter` argument is optional. If present it is one of `metal`, `plastic`, and ensures that only buttons of that type are displayed.

3. Consider the following JavaScript code: **11**

```
function a(k, x1, x2, x3, x4) {
  function b() {
    k -= 1;
    return a(k, b, x1, x2, x3);
  }
  return (k > 0) ? b() : x3() + x4();
}

function x(n) {
  return function () {
    return n;
  };
}
```

   Assume we have base types `int` and `bool`, and given the invocation,

   `a(10, x(1), x(-1), x(1), x(0));`

   What type is returned? What are the principle types of `a`, `b`, and `x`? Give a formal proof that your types are correct.

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**.

For each WML question n, include an ASCII file `qn.txt` with the source code of your answer. This file should allow basic cut-and-paste into a WML interpreter. Do not include the provided files (`wml.html` and `wml.js`), or `assig3.js`. For question 3, provide either a `q3.pdf` or a well-formatted `q3.txt` file.

This assignment is worth 10% of your final grade. **35**