University of Arizona, Department of Computer Science

CSc 520 — Assignment 3 — Due noon, Wed Mar 26 — 10%

Christian Collberg
March 6, 2008

1 Introduction

In this assignment you'll experience polymorphism first hand! In the first part you'll write a generic array search routine in Ada (using *parametric polymorphism*) and in the second part you'll write a list search routine in Smalltalk (using *inclusion polymorphism*).

1. In this assignment you'll be doing some "language archeology" — you'll be programming in two languages you know very little about, and have to learn on the fly! This is a very useful skill!
2. In this assignment you are encouraged to "cheat" — make use of whatever resources you can find on the web! There are lots of language tutorials, example programs, etc. that you may find helpful! You can't, however, copy from other students in the class!
3. In your README file you should list any resources you have been using.
4. You should work in a team of two students. Avoid the temptation to say "Lucy will do the Ada program and Bob will do the Smalltalk program." A better strategy is to sit down together and work on each program in turn. This will both be more fun and more educational!

2 Parametric Polymorphism in Ada

Write a generic package for searching through an array for a particular value. Use linear search. Your program should consist of three files: `search.ads` (the package specification, `search.adb` (the package implementation) and `searchmain.adb`, the main program.

The file `search.ads` should look something like this:

```
generic
   <stuff here>
package SEARCH is
   function MEMBER(arr : in VECTOR; Elmt : ITEM) return Boolean;
end SEARCH;
```

The file `search.adb` should look something like this:

```
package body SEARCH is
   function MEMBER(ARR : in VECTOR; Elmt : ITEM) return Boolean is
       <stuff here>
   end MEMBER;
end SEARCH;
```

Finally, the main program in `searchmain.adb` should look like this:

```
with Ada.Text_Io; use Ada.Text_Io;
with Search;

procedure SEARCHMAIN is
   Max : constant Positive := 25;
   type IntArr is array (Positive range <>) of Positive;
   Ints : IntArr(1..Max);

   <stuff here>
begin
   Ints(1) := 5;  Ints(2) := 3;
   Found := INTSEARCH.MEMBER(Ints(1..2),3);
   if Found then
      Put("found!");
   else
      Put("not found!");
   end if;
   New_Line;
end SEARCHMAIN;
```

1. GNU's Ada implementation has been installed on lectura. You can install Ada on your own machines, of course, although it probably isn't worth it for such a small program Get the compiler from `https://libre.adacore.com`.
2. To run the compiler on lectura start by adding `/usr/local/gnat/bin` to your `PATH`.
3. Complete the files above. Compile and run with

   ```
   > gnatmake search.ads search.adb searchmain.adb
   > searchmain
   found!
   ```

4. Extend `searchmain` such that you can now search for the real number 3.0 in an array `[5.0,3.0]`. I.e. you should get the output:

   ```
   > gnatmake search.ads search.adb searchmain.adb
   > searchmain
   found!
   found!
   ```

5. Create a record `Pair` that has two integer fields. Extend your program so that `MEMBER` can search for a `Pair` in an array of `Pairs`!

3 Inclusion Polymorphism in Smalltalk

1. Get *squeak* from `http://squeak.org` and install it on your machine (I haven't installed it on lectura).
2. Download *Squeak by Example* (SBE) from `http://squeakbyexample.org`.
3. If you don't have a three-button mouse, you may want to consider investing in one...

4. Start playing with squeak by following the examples in SBE.
5. Implement a linked list "package" in Smalltalk. It should have a method `add` that lets you add a new object to the beginning of a list and a method `toString` that returns a string representation of the list. As much as is convenient, mirror this Java program:

```
class ListNode {
   ListNode next;
   public String toString() {
      return "";
   }
}

class List {
   ListNode first;

   public List () {
      first = new ListNode();
   }

   public void add (ListNode obj) {
      obj.next = first.next;
      first.next = obj;
   }

   public String toString() {
      ListNode f = first.next;
      String s = "[";
      while (f != null) {
         s += f.toString();
         if (f.next != null)
            s += ",";
         f = f.next;
      }
      s += "]";
      return s;
   }
}

class IntNode extends ListNode {
   int val;
   public IntNode(int val) {
      this.val = val;
   }
   public String toString() {
      return val + "";
   }
}

class FloatNode extends ListNode {
   float val;
   public FloatNode(float val) {
      this.val = val;
   }
   public String toString() {
      return val + "";
   }
}

class Main {
    public static void main (String args[]) {
        List list = new List();
        list.add(new IntNode(42));
        list.add(new FloatNode(4.2F));
        System.out.println(list.toString());
    }
}
```

4 CV update

Update your CV with the line:

- *Fluent in C, Java, ...,* **Ada**, ... **Smalltalk**, ...

5 Submission and Assessment

The deadline for this assignment is noon, Wed Mar 26. It is worth 10% of your final grade.

You should submit the assignment electronically using the `Unix` command

```
turnin cs520.3 search.ads search.adb searchmain.adb list.st Makefile README
```

Your electronic submission *must* contain a working `Makefile`, and *all* the files necessary to build the interpreter. If your program does not compile "out of the box" you *will* receive *zero* (0) points. The grader will *not* try to debug your program or your makefile for you!

Don't show your code to anyone, don't read anyone else's code, don't discuss the details of your code with anyone. If you need help with the assignment see the instructor or the TA.