# CLU Examples

- From Liskov et al, CACM Aug77 article *Abstraction Mechanisms in CLU*
  - EG1 - how to define a class in CLU; shows rep type and operation signatures but not the specification
  - EG2 - how to define an iterator in CLU



Fig. 3. The *wordbag* cluster.

```
wordbag = cluster is
   create,      % create an empty bag
   insert,      % insert an element
   print;       % print contents of bag
   rep = record [contents: wordtree, total: int];
create = proc ( ) returns (cvt);
         return (rep${contents: wordtree$create ( ), total: 0});
         end create;
insert = proc (x: cvt, v: string);
         x.contents := wordtree$insert (x.contents, v);
         x.total := x.total + 1;
         end insert;
print = proc (x: cvt, o: outstream);
        wordtree$print (x.contents, x.total, o);
        end print;
end wordbag;
```

class interface

Rep type

Data abstraction is wordbag;
Rep type is record - binary tree
Outside cluster only reveal wordbag
Within cluster see implementation of operations in terms of rep type – binary tree record

```
wordtree = cluster is
   create,     % create empty contents
   insert,     % add item to contents
   print;      % print contents
   node = record [value: string, count: int,
                  lesser: wordtree, greater: wordtree];
   rep = oneof [empty: null, non_empty: node];
create = proc ( ) returns (cvt);
   return (rep$make_empty (nil));
   end create;
insert = proc (x: cvt, v: string) returns (cvt);
   tagcase x
      tag empty:
            n: node := node${value: v, count: 1,
                            lesser: wordtree$create ( ),
                            greater: wordtree$create ( )};
            return (rep$make_non_empty (n));
      tag non_empty (n: node):
            if v = n.value
                  then n.count := n.count + 1;
               elseif v < n.value
                  then n.lesser := wordtree$insert (n.lesser, v);
               else n.greater := wordtree$insert (n.greater, v);
               end;
            return (x);
      end;
   end insert;
print = proc (x: cvt, total: int, o: outstream);
   tagcase x
      tag empty: ;
      tag non_empty (n: node):
         wordtree$print (n.lesser, total, o);
         print_word (n.value, n, count, total, o);
         wordtree$print (n.greater, total, o);
      end;
   end print;
end wordtree;
```

oneof: A union type where value depends on whether or not wordtree is empty.

Tagcase stmt: depends on rep type of current wordtree



Iterator: to separate selection of next object from a collection and performing an operation on that object. Provides objects from a collection one-by-one without repeats.

Fig. 5. Use and definition of a simple iterator.

```
count_numeric = proc (s: string) returns (int);
   count: int := 0;
   for c: char in string_chars (s) do
      if char_is_numeric (c)
         then count := count + 1;
         end;
      end;
   return (count);
   end count_numeric;
string_chars = iter (s: string) yields (char);
   index: int := 1;
   limit: int := string$size (s);
   while index <= limit do
      yield (string$fetch (s, index));
      index := index + 1;
      end;
   end string_chars;
```