Mars DiPillo

1. Does your language support objects or something similar (e.g., structs or records)?
   1. For Fortran, object support only exists after Fortran 2003, any flavor of that language before that, the support does not exist. However, from Fortran 77 and 66, records and structures exist in the sense that structures are a record’s template. Additionally, there are no intrinsic constructors. For Fortran 95 and 90, structures and records were replaced with types, that act similarly to structures and records but on a simpler level.
   2. Are there naming conventions for objects, instance variables, or functions that people writing in your language should be aware of?
      1. For Fortran 66/77:
         1. For structures:
            1. Start writing your record/structure with STRUCTURE /structName/ and end the record/structure with END STRUCTURE.

You will additionally need to add a record /structName/ current, prior, next, line(10), in order for your structure to work properly.

* + - 1. For records:
         1. RECORD /structName/ record-list [/structName/ record-list]
    1. For Fortran 95/90
       1. TYPE
          1. Include type nameof\_type

Character(establish length here):: var1

Integer:: var2

Real:: var3

* + - * 1. End the type
        2. Type(nameof\_type) :: thing
        3. Thing = nameof\_type(“text”, 2. 3.0)
    1. Fortran 2003 and later flavors:
       1. Start with ‘module nameof\_modle’ and end with ‘end module nameof\_module’. There is a requirement to make the class public or private, and to what type of variables you need. Then a procedure may be used underneath the contents. The contains section may also include a subroutine.

1. Does your language have standard methods for functions that serve a similar purpose across all objects? For example, toString() in Java and \_\_str\_\_ in Python allow information about the objects to be printed. Are there similar functions in your language?
   1. Fortran 2003 and above, there seems to be no standard methods or ability to include functions inside these classes. Instead, there are quick procedures, subroutines, and the ability to create and customize a string function.
2. How does inheritance work (if it does)? Does your language support multiple inheritance?
   1. For emulation prior to Fortran 2003, multiple inheritance was not technically feasible. After the 2003 flavor, multiple inheritance was possible to emulate. According to Karla Morris’s research, Fortran’s inheritance is structured similarly to how C++ handles inheritance. Multiple inheritance in newer Fortran works as follows:
      1. Create a superclass that can be inherited from other classes
      2. Create distribution and collection classes so data can be collected and distributed to all other classes.
      3. Constructor functions return objects
      4. For Fortran 95/90: inheritance works by creating a type, then when creating the next type, put in: type(nameof\_type):: typething
         1. When assigning values to the object, you need to do this:
            1. Type1(type2(“namehere”, id here), “class name here”)
3. If there is inheritance, how does your language deal with overloading method names and resolving those calls?
   1. Despite lengthy research, there are no obvious answers at all for how Fortran handles overloading method names and how to resolve those calls. Judging from how Fortran operates and basic experience, it seems as if Fortran would not allow the usage of overloading. That is, and this is very much an educated guess, that Fortran only allows unique function names. As in, you cannot re-use the same name despite using different parameters.
4. Is there anything else that’s important to know about objects and inheritance in your language?
   1. I know this question is vague, but objects are were languages can vary a lot, so it’s hard to ask questions that will apply to everyone.
   2. Know your version of Fortran! If you have Fortran 2003 and later, you can implement objects fine. Though, if you using older versions of Fortran, for whatever reason, then implementing objects will be far more complex. With a long search of answers on how to emulate objects in older Fortran flavors, I have come to no basic conclusions. The best bet would be to utilize structures and records, use functions that inherit variables from a main function (not the MAIN function, but a superfunction). Again, that is all a guess from a programmer who is still learning Fortran.
   3. For Fortran 90/95:
      1. When printing from an inherited class and all, make sure you format it like this name%type1%first\_name or maxwell%person%f\_name

 Sources

<https://docs.oracle.com/cd/E19957-01/805-4939/6j4m0vnb1/index.html>

<https://onlinelibrary.wiley.com/doi/10.1155/2015/126069>

<https://cyber.dabamos.de/programming/modernfortran/object-oriented-programming.html>

<https://fortranwiki.org/fortran/show/Object-oriented+programming>

<https://icl.utk.edu/~mgates3/docs/fortran.html>