# Polymorphism

Jules Kouatchou

NASA GSFC Code 606 (ASTG)
Greenbelt, Maryland 20771

Jules.Kouatchou@nasa.gov

October 25, 2018

# Agenda

## Polymorphism

- Introduction
- Procedure Polymorphism
- Data Polymorphism
  - Polymorphic Pointer Variable
  - Allocatable Polymorphic Variable


# Introduction

- Use to describe a variety of techniques employed by programmers to create flexible and reusable software components.
- A polymorphic object is an entity, such as a **variable** or a **procedure**, that can hold or operate on values of differing types during the program's execution.


# Introduction - Cont

There are two basic types of polymorphism:

Procedure polymorphism: Deals with procedures that can operate on a variety of data types and values.

Data polymorphism: Deals with program variables that can store and operate on a variety of data types and values.


# Procedure Polymorphism

- Occurs when a procedure, such as a function or a subroutine, can take a variety of data types as arguments
- Accomplished when a procedure has one or more dummy arguments declared with the **CLASS** keyword.


# Example

```
1  type, extends(shape) :: polygon
2     integer :: color
3  contains
4     procedure :: get_area
5     procedure :: set_color
6  end type polygon
7
8  subroutine set_color(plg, color)
9      class(polygon), intent(inOut) :: plg
10     integer, intent(in) :: color
11     plg%color = color
12 end subroutine set_color
```


# Comments

- The `set_color` subroutine takes two arguments, `plg` and `color`.
- The `plg` dummy argument is polymorphic, based on the usage of `class(plygon)`.
- The subroutine can operate on objects that satisfy the "is a" polygon relationship. So, `set_color` can be called with a `polygon`, `circle`, `rectangle`, `square`, or any future type extension of shape.


# Another Example

```
1   subroutine initialize_polygon(plg, color, radius, length, width)
2       class(polygon) :: plg
3       integer :: color
4       real, optional :: radius
5       real, optional :: length, width
6
7       plg%color = color
8
9       SELECT TYPE (plg)
10      type is (polygon)
11      class is (circle)
12          if (present(radius)) then
13             plg%radius = radius
14          else
15             plg%radius = 0
16          endif
17      class is (rectangle)
18          if (present(length)) then
19             plg%length = length
20          else
21             plg%length = 0
22          endif
23          if (present(width)) then
24              plg%width = width
25          else
26              plg%width = 0
27          endif
28      class default
29           stop 'initialize: unexpected type for plg object!'
30      end select
31  end subroutine initialize_polygon
```

SSAI

# Data Polymorphism

- A polymorphic variable is a variable whose data type is dynamic at runtime.
- It must be a pointer variable, allocatable variable, or a dummy argument.


# Example

```
1  subroutine init(plg)
2    class(plygon) :: plg ! polymorphic dummy argument
3    class(plygon), pointer :: p ! polymorphic pointer variable
4    class(plygon), allocatable :: alp ! polymorphic allocatable variable
5  end subroutine init
```

- The `plg`, `p` and `alp` polymorphic variables can each hold values of type shape or any type extension of polygon.
- The `plg` dummy argument receives its type and value from the actual argument to `plg` of subroutine `init()`.


# Polymorphic Pointer Variable

The polymorphic pointer variable `p` can point to an object of type `polygon` or any of its extensions.

```
1  subroutine init(plg)
2    class(polygon), target :: plg
3    class(polygon), pointer :: p
4
5    select type (plg)
6    type is (polygon)
7      p => plg
8      : ! shape specific code here
9    type is (circle)
10     p => plg
11     : ! rectangle specific code here
12   type is (rectangle)
13     p => plg
14     : ! rectangle specific code here
15   type is (square)
16     p => plg
17     : ! square specific code here
18   class default
19     p => null()
20   end select
21 end subroutine init
```


# Allocatable Polymorphic Variable

- An allocatable polymorphic variable receives its type and optionally its value at the point of its allocation.
- By default, the dynamic type of a polymorphic allocatable variable is the same as its declared type after executing an allocate statement.

```
1  class(polygon), allocatable :: alp1, alp2
2  allocate(alp1)
3  allocate(rectangle::alp2)
```


# Example

```
1   subroutine init(plg)
2     class(polygon) :: plg
3     class(polygon), allocatable :: alp
4
5     select type (plg)
6     type is (polygon)
7       allocate(polygon::alp)
8       select type(alp)
9       type is (polygon)
10        alp = plg ! copy sh
11      end select
12    type is (circle)
13      allocate(circle::alp)
14      select type(alp)
15      type is (circle)
16        alp = plg ! copy sh
17      end select
18    type is (rectangle)
19      allocate(rectangle::alp)
20       select type (alp)
21       type is (rectangle)
22         alp = plg ! copy plg
23       end select
24    type is (square)
25      allocate(square::alp)
26      select type (alp)
27      type is (square)
28        alp = plg ! copy plg
29      end select
30    end select
31  end subroutine init
```

SSAI

# Exercise

- For each child class, implement a method that computes the perimeter of a polygon
- Write a subroutine that takes as argument an arbitrary `polygon` and prints the perimeter of the `polygon`.
- Write a simple program that initialzes various polygons and calls the above subroutine.