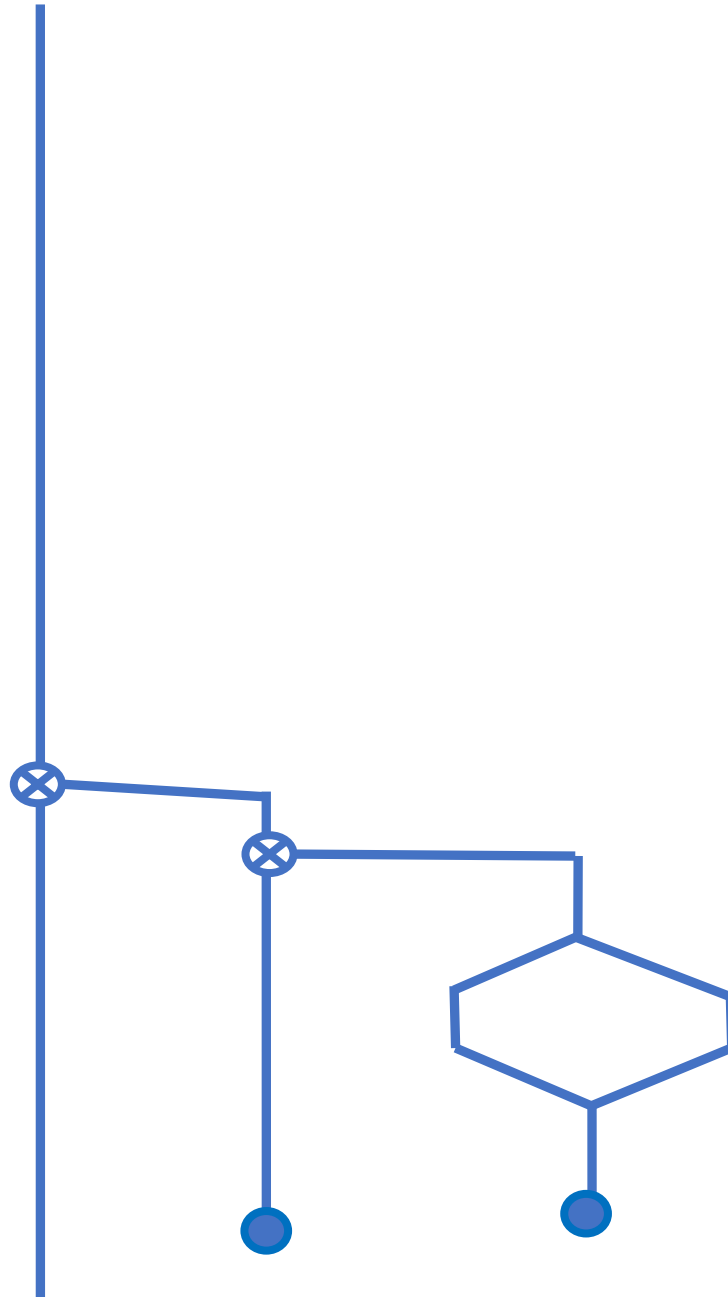


In Introduction To Win32Forth Object Oriented Programming



SVFIG
Oct. 25, 2025
Bill Ragsdale

Object Oriented Programming

Initially, by Ole-Johan Dahl and Kristen Nygaard in the 1960s leading to Simula.

The term "object-oriented programming" coined by Alan Kay around 1966 working on Smalltalk.

But we know the concepts were developed by Charles Moore in the 1960s via CREATE DOES >.

The Win32Forth implementation is patterned after NEON and YERK.

OOP Key Elements

A class is comprised of similar objects and provides computational resources. Think “colon-definitions & variables”.

Those resources may be passed downward to ‘child’ classes and objects. Inheritance.

An object is an isolated, functional structure for a specific purpose.

Objects provide encapsulation Their components:

- Data elements are only accessible from within.

- Methods form the only access into an object.

OOP Key Elements

A **class** inherits resources from a super-class.

It creates an **object** defining word, variables and methods.

The data items are called **instance variables**.

The executable code is called a **method**. These simply are colon definitions usable only when associated with an object.

Parameters are passed to methods via **messages**. The term 'message' is just a high-priced term for stack values.

Using An Internal Method

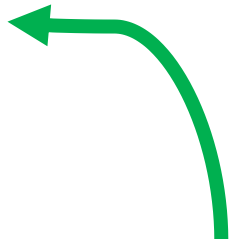
As methods are shared, their usage must link to an object.

We have three ways to link a method to an object.

Within the structure of an object 'self' refers to methods within the object:

```
:M Product:  dup * ;M
```

```
:M Scale:  length Product: self ;M
```



The Early Binding Method

When an object has been defined before use, a prefix syntax is used; the method followed by its object:

```
Class: MyClass <super object
```

```
    int length
```

```
:M Product: length * ;M ;CLASS
```

```
MyClass MyObject \ creates an object
```

```
: Area 1000 Product: MyObject ;
```



The Late Binding Method

When a Class has been defined, but no objects, the late binding syntax is used. The object is later passed by its address.

```
Class: MyClass <super object
```

```
    int length
```

```
:M Product: length * ;M ;CLASS
```

```
: Area ( n1 addr --- ) Product: MyClass ;
```

```
MyClass MyObject
```

```
1000 MyObject Area
```



Instance Variables

byte <name> to hold a byte value

short <name> to hold a word value

int <name> to hold a cell value

dint <name> to hold a double or float

b/float bytes <name> to hold a float

int Holder cell to hold bits

3 bits 3By 5 bits 5By 24 bits filler

Record: Local int left int right ;Record

Create Does > Birthday Example

```
: SetBirthDay ( month day name --- ) 2! ;
```

```
: GetBirthDay ( name --- month day ) 2@ ;
```

```
: Family CREATE 2 cells allot DOES> ;
```

```
Family Betty
```

```
Family U.S.A
```

```
10 5 Betty SetBirthday
```

```
7 4 U.S.A. SetBirthday
```

The code in Family is shared.
Each Family word has its
own storage area.

Create Does > Birthday Report

Betty GetBirthDay

.(for Betty) swap .(Month) . .(Day) .

See: for Betty Month 10 Day 5

U.S.A GetBirthDay

.(for U.S.A) swap .(Month) . .(Day) .

See: for U.S.A Month 7 Day 4

OOP Birthday Example

```
:CLASS Family      <Super Object
    dint Birthday  \ month & day
:M SetBirthday:    TO Birthday ;M
:M GetBirthday:    Birthday ;M ;CLASS
```

```
Family Betty      10 5  SetBirthday: Betty
```

```
Family U.S.A.     7 4  SetBirthday: U.S.A.
```

OOP Birthday Report

GetBirthDay: Betty

.(for Betty) swap .(Month) . .(Day) .

See: for Betty Month 10 Day 5

GetBirthDay: U.S.A.

.(for U.S.A.) swap .(Month) . .(Day) .

See: for U.S.A Month 7 Day 4

OOP President Example

```
:CLASS U.S.Constitution
```

```
<Super Object
```

```
;CLASS
```

The class inherits from the super-class Object.

OOP President Example

```
:CLASS U.S.Constitution <Super Object ;CLASS
```

```
:CLASS President <Super U.S.Constitution  
;CLASS
```

We now have a class consisting of U.S. Presidents. They share common inheritance from the U.S. Constitution, and the (to be defined) common methods and each to have their (to be defined) OWN instance variables.

OOP President Example

```
:CLASS U.S.Constitution <Super Object ;CLASS
```

```
:CLASS President <Super U.S.Constitution
```

```
    int ArmySize    int UStreasury    int USdebt
```

```
;CLASS
```

Adding integer instance variables.

OOP President Example

```
:CLASS U.S.Constitution <Super Object ;CLASS
```

```
:CLASS President <Super U.S.Constitution
```

```
    int ArmySize    int UStreasury    int USdebt
```

```
:M Entry:  to USdebt    to UStreasury    to ArmySize ;M
```

```
:M Record: +TO USdebt +TO UStreasury +TO ArmySize ;M
```

```
:M Report: ArmySize UStreasury USdebt ;M
```

```
;CLASS \ Adding methods.
```


President Support

```
: .header    ( president ---)
    cr cr ."   President " 2 cells- .name
    cr cr ."   Year      Army    Treasury  Debt" ;

\ shows report data for one year.
: .Show      ( army treasury debt year --- )
    cr 6 .r 2 spaces rot 7 .r swap 8 .r 8 .r ;
```

President Data

```
: YearChanges      ( year --- Army Treas Debt )
CASE      1791 of      1000      0      -11 endof
          1792 of        500      0       -5 endof
          1793 of        500      0       -6 endof
 1794 of     10000      0       -1 endof
          1795 of     -8500      0       -1 endof
          1796 of          0      0        -2 endof
          1797 of        500      0        -2 endof
          1861 of        400      0       -26 endof
 1862 of     73600      0     -433 endof
          1863 of    510000      0     -476 endof
          1864 of    445000      0     -800 endof
          1865 of          0      0     -900 endof
      abort" Bad Year" endcase ;
```

President Report

```
: Action ( president start end --- )  
  2 pick .header           \ president show header  
  2 pick Report: President \ current year's data  
  4 pick 1- .Show          \ pres start end  
  1+ swap                  \ over years of presidency  
  do  
    i YearChanges          \ get year's changes  
    3 pick Record: President \ place changes  
    dup Report: President \ year's data  
    i .Show                \ year's display  
  loop drop ;
```

President Washington's Definition

President Washington

500 0 -54 Entry: Washington \ prior year

Washington 1791 1797 Action

Washington's Report To Congress

President WASHINGTON

Year	Army	Treasury	Debt (millions)
1790	500	0	-54
1791	1500	0	-65
1792	2000	0	-70
1793	2500	0	-76
→ 1794	12500	0	-77
1795	4000	0	-78
1796	4000	0	-80
1797	4500	0	-82 ok

President Lincoln's Definition

President Lincoln

16000 0 -65 Entry: Lincoln \ prior year

Lincoln 1861 1865 Action

Lincolns's Report To Congress

President LINCOLN

Year	Army	Treasury	Debt (millions)
1860	16000	0	-65
1861	16400	0	-91
1862	90000	0	-524
→ 1863	600000	0	-1000
1864	1045000	0	-1800
1865	1045000	0	-2700 ok

Conclusion

OOP has features in common with Forth.

However, it has greatly expanded internal data and methods (execution code).

OOP has multi-level inheritance.

I find about a three to one increase in function per unit of code. Matrices in one page.

Win32Forth OOP documentation is the bare minimum.

As an OOP novice I am on a detective search for knowledge.