Typology of programming languages

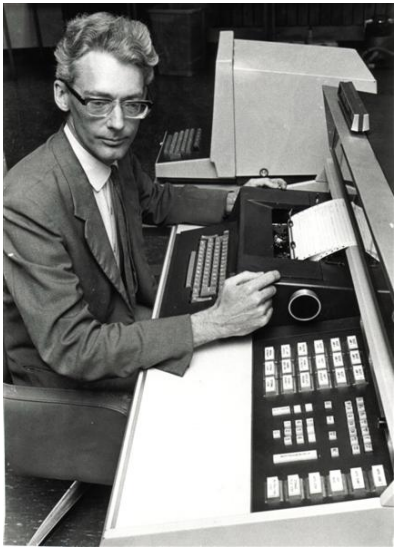
Simula

Table of Contents

Simula

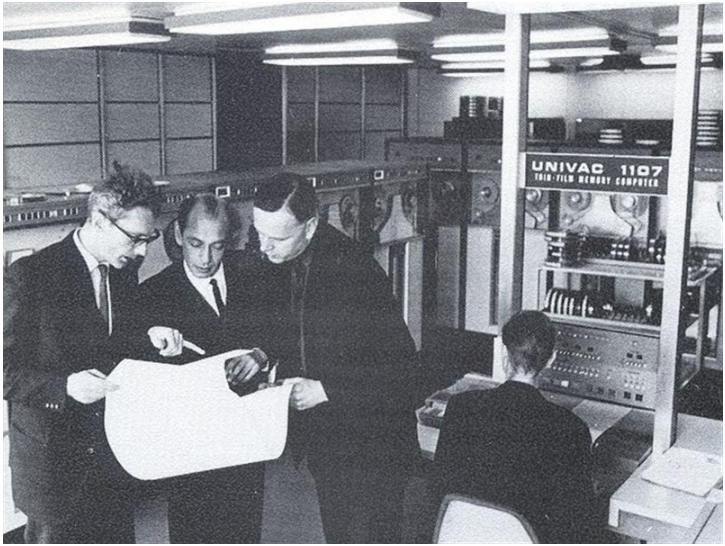
Ole-Johan Dahl

Simula



Ole-Johan Dahl

Simula



Dahl & Nygaard

Simula

Ole-Johan Dahl & Kristen Nygaard
(ca. 1963)

Simula

Nygaard & Dahl: Turing Award 2001

2002... Sad Year

Ole-Johan Dahl

Oct 12, 1931, Mandal, NO

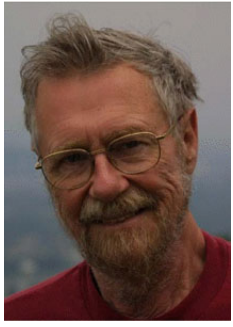
June 29, 2002, Asker, NO
"...are there too many basic mechanisms floating around doing nearly the same thing?"

Kristen Nygaard

Aug 27, 1926, Oslo, NO

Aug 10, 2002, Oslo, NO
"To program is to understand!"

Edsger Wybe Dijkstra



May 11, 1930, Rotterdam, NL
Aug 06, 2002, Nuenen, NL
"Do only what only you can"

Table of Contents

Simula

> *In the spring of 1967 a new employee at the NCC in a very shocked voice told the switchboard operator: "two men are fighting violently in front of the blackboard in the upstairs corridor. What shall we do?" The operator came out of her office, listened for a few seconds and then said: "Relax, it's only Dahl and Nygaard discussing SIMULA." — Kristen Nygaard, Ole-Johan Dahl.*

Physical system models. Norwegian nuclear power plant program.

Simula

> *In the spring of 1967 a new employee at the NCC in a very shocked voice told the switchboard operator: "two men are fighting violently in front of the blackboard in the upstairs corridor. What shall we do?" The operator came out of her office, listened for a few seconds and then said: "Relax, it's only Dahl and Nygaard discussing SIMULA." — Kristen Nygaard, Ole-Johan Dahl.*

Physical system models. Norwegian nuclear power plant program.

Basic concepts (1/3)

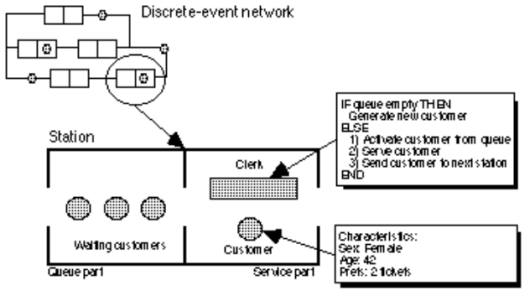
- A **system**, consisting of a finite and fixed number of active components named **stations**, and a finite, but possibly variable number of passive components named **customers**.
- A **station** consisting of two parts: a queue part and a service part. Actions associated with the service part, named the **station's operating rule**, were described by a sequence of ALGOL (or ALGOL-like) statements.

Basic concepts (2/3)

- A customer with no operating rule, but possibly a finite number of variables, named **characteristics** .
- A real, continuous variable called **time** and a **function position**, defined for all customers and all values of time.

Basic concepts (3/3)

This structure may be regarded as a **network**, and the events (actions) of the stations' service parts are regarded as **instantaneous and occurring at discrete points of time**, this class of systems was named **discrete event networks**.



Simula I

- An ALGOL 60 preprocessor
- A subprogram library
- An original per “process” stack allocation scheme

Not yet the concept of objects.

Quasi-parallel processing is analogous to the notion of coroutines described by Conway in 1963.

Table of Contents

Simula 67

- Introduces:
  - the concept of object,
  - the concept of class,
  - literal objects (constructors),
  - the concept of inheritance (introduced by C. A. R. Hoare for records),
  - the concept of virtual method,
  - attribute hiding!
- Immense funding problems
  steady support from C. A. R. Hoare, N. Wirth and D. Knuth.
- Standardized ISO 1987.

Shape in Simula (1/5)

```
class Shape(x, y); integer x; integer y;
virtual: procedure draw is procedure draw;;
begin
   comment -- get the x & y components for the object --;
   integer procedure getX;
      getX := x;
   integer procedure getY;
      getY := y;
   comment -- set the x & y coordinates for the object --;
   integer procedure setX(newx); integer newx;
      x := newx;
   integer procedure setY(newy); integer newy;
      y := newy;
   comment -- move the x & y position of the object --;
   procedure moveTo(newx, newy); integer newx; integer newy;
      begin
         setX(newx);
         setY(newy);
      end moveTo;
   procedure rMoveTo(deltax, deltay); integer deltax; integer deltay;
      begin
         setX(deltax + getX);
         setY(deltay + getY);
      end moveTo;
end Shape;
```

Shape in Simula (2/5)

```
Shape class Rectangle(width, height);
  integer width; integer height;
begin
   comment -- get the width & height of the object --;
   integer procedure getWidth;
      getWidth := width;
   integer procedure getHeight;
      getHeight := height;
   comment -- set the width & height of the object --;
   integer procedure setWidth(newwidth); integer newwidth;
      width := newwidth;
   integer procedure setHeight(newheight); integer newheight;
      height := newheight;
   comment -- draw the rectangle --;
   procedure draw;
      begin
         Outtext("Drawing a Rectangle at:(");
         Outint(getX, 0);  Outtext(",");  Outint(getY, 0);
         Outtext("), width ");  Outint(getWidth, 0);
         Outtext(", height ");  Outint(getHeight, 0);
         Outimage;
      end draw;
end Rectangle;
```

Shape in Simula (3/5)

```
Shape class Circle(radius); integer radius;
begin
   comment -- get the radius of the object --;
   integer procedure getRadius;
      getRadius := radius;

   comment -- set the radius of the object --;
   integer procedure setRadius(newradius); integer newradius;
      radius := newradius;

   comment -- draw the circle --;
   procedure draw;
      begin
         Outtext("Drawing a Circle at:(");
         Outint(getX, 0);
         Outtext(",");
         Outint(getY, 0);
         Outtext("), radius ");
         Outint(getRadius, 0);
         Outimage;
      end draw;
end Circle;
```

Shape in Simula (4/5)

```
comment -- declare the variables used --;
ref(Shape) array scribble(1:2);
ref(Rectangle) arectangle;
integer i;

comment -- populate the array with various shape instances --;
scribble(1) :- new Rectangle(10, 20, 5, 6);
scribble(2) :- new Circle(15, 25, 8);

comment -- iterate on the list, handle shapes polymorphically --;
for i := 1 step 1 until 2 do
   begin
      scribble(i).draw;
      scribble(i).rMoveTo(100, 100);
      scribble(i).draw;
   end;

comment -- call a rectangle specific instance --;
arectangle :- new Rectangle(0, 0, 15, 15);
arectangle.draw;
arectangle.setWidth(30);
arectangle.draw;
```

Shape in Simula – Execution (5/5)

```
> cim shape.sim
Compiling shape.sim:
gcc -g -O2 -c shape.c
gcc -g -O2 -o shape shape.o -L/usr/local/lib -lcim
> ./shape
Drawing a Rectangle at:(10,20), width 5, height 6
Drawing a Rectangle at:(110,120), width 5, height 6
Drawing a Circle at:(15,25), radius 8
Drawing a Circle at:(115,125), radius 8
Drawing a Rectangle at:(0,0), width 15, height 15
Drawing a Rectangle at:(0,0), width 30, height 15
```

Impact of Simula 67

All the object-oriented languages inherit from Simula.

Smalltalk further with object orientation,
further with dynamic binding.

Impact of Simula 67

All the object-oriented languages inherit from Simula.

Smalltalk further with object orientation,
further with dynamic binding.

Objective-C, Pascal, C++, etc.
further with messages.

Impact of Simula 67

All the object-oriented languages inherit from Simula.

Smalltalk further with object orientation,
further with dynamic binding.

Objective-C, Pascal, C++, etc.
further with messages.

CLOS further with method selections.

Impact of Simula 67

All the object-oriented languages inherit from Simula.

Smalltalk	further with object orientation, further with dynamic binding.
Objective-C, Pascal, C++, etc.	further with messages.
CLOS	further with method selections.
Eiffel	further with software engineering, further with inheritance.

Impact of Simula 67

All the object-oriented languages inherit from Simula.

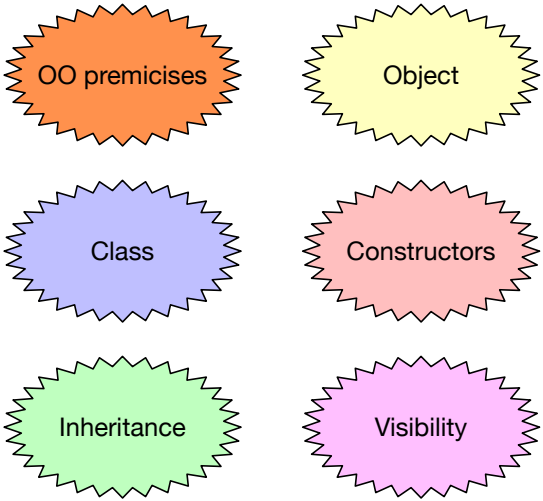
Smalltalk	further with object orientation, further with dynamic binding.
Objective-C, Pascal, C++, etc.	further with messages.
CLOS	further with method selections.
Eiffel	further with software engineering, further with inheritance.
C++	further with static typing and static binding, deeper in the *.

Impact of Simula 67

All the object-oriented languages inherit from Simula.

Smalltalk	further with object orientation, further with dynamic binding.
Objective-C, Pascal, C++, etc.	further with messages.
CLOS	further with method selections.
Eiffel	further with software engineering, further with inheritance.
C++	further with static typing and static binding, deeper in the *.
Hybrid languages	logic, functional, assembly, stack based etc.

Summary


OO premicises
Object
Class
Constructors
Inheritance
Visibility