Object Oriented Languages

Object Oriented Languages support:

- Data abstraction (the encapsulation of state with operations).
- Information hiding (encapsulation).
- Message passing and polymorphism.
- Inheritance, including dynamic binding.

Significant languages in the history of Object Orientation.

- Simula-67
- Smalltalk
- C++
- Java
- Oberon, Eiffel and many others

Simula-67

- Originated in Norway by Nygaard and Dahl in 1961, and was fully developed by 1967.
- Based on Algol 60.
- Designed for discrete event simulation.
- It extended Algol to contain objects that had state and a thread of control (similar to coroutine) that could simulate parallel activities.
- It had an inheritance mechanism.

Simula-67 Example

A simulation of a post office with a door, 4 counters, each with a clerk and a queue, and a random supplied of customers entering the door.

Simula-67 Example

```
BEGIN

CLASS customer(tasks, oldlady)
  INTEGER tasks; BOOLEAN oldlady;      BEGIN ... END;

CLASS queue;                           BEGIN ... END;

CLASS clerk(q); ref(queue)q;           BEGIN ... END;

CLASS door;                            BEGIN ... END;

CLASS counter;                         BEGIN ... END;

REAL opentime, closetime;
REF(counter) c1, c2, c3, c4;

opentime := 8.00;
closetime := 17.00;

hold(opentime);
c1 :- NEW counter;
c2 :- NEW counter;
c3 :- NEW counter;
c4 :- NEW counter;
NEW door;

hold(closetime);
END
```

Simula-67 Example

```
CLASS customer(tasks, oldlady)
  INTEGER tasks; BOOLEAN oldlady;
BEGIN
  REF(customer) next;
  REF(counter) service;

  WHILE tasks>0 DO
  BEGIN
    <assign value to service>;
    IF oldlady THEN <enter front of service.q>
               ELSE <enter tail of service.q>;
    IF service.postofficer is free
    THEN <activate service.postofficer>
    ELSE passivate;
    <participate in transaction>;
    <leave counter>
    tasks := tasks-1;
  END
END;
```

Simula-67 Example

```
CLASS queue;
BEGIN
  REF(customer) first, last;
END;


CLASS clerk(q); REF(queue)q;
BEGIN
  REF(customer) person;
seviceing:
  WHILE q.first =/= NONE DO
  BEGIN person :- q.first;
        <take person out of the queue>;
        <engage with him in transactions>;
  END;
  <do other work until interrupted by the
   arrival of a new customer in the queue>;
  GOTO servicing;
END;
```

Simula-67 Example

```
CLASS door;
BEGIN
  REAL arrtime;

  WHILE time<=closetime DO
  BEGIN arrtime := <time of next arrival>;
        hold(arrtime);
        NEW customer(<initial value of tasks>,
                     <initial value of oldlady>);
  END;
END;



CLASS counter;
BEGIN
  REF(queue) q;
  REF(clerk) postofficer;

  q :- NEW queue;
  postofficer :- NEW clerk(q);
END;
```

Simula-67 Class Hierachy

A hierachy of classes can be declared.

```
  CLASS a ...;
a CLASS b ...;
a CLASS c ...;
b CLASS d ...;
b CLASS e ...;
```

```
     Hierarchy              Objects


                     a     b     c     d     e
                    ---   ---   ---   ---   ---
          a        | a | | a | | a | | a | | a |
         / \        ---  |---| |---| |---| |---|
        /   \            | b | | c | | b | | b |
       b     c            ---   ---  |---| |---|
      / \                            | d | | e |
     /   \                            ---   ---
    d     e
```

Simula-67 Various Primitive

Componentes in a simulation which may be scheduled should be prefixed by the class `process`. The class outline is:

```
link CLASS process;
BEGIN BOOLEAN PROCEDURE idle; ...;
      REAL PROCEDURE evtime; ...;
      REF(process) PROCEDURE nextev; ...;
      detach;
      ...
      passivate;
END;
```

Simula-67 Various Primitive

```
REF(process) PROCEDURE current; ...
```

returns the current process.

```
REAL PROCEDURE time; ...
```

returns the current time.

```
PROCEDURE hold(t); REAL t; ...
```

suspend the current process until time `t`..

```
PROCEDURE cancel(p); REF(process) p; ...
```

remove p from the event queue.

```
ACTIVATE p;
ACTIVATE p AT time;
ACTIVATE p DELAY time;
```

activate `p` at specified time.