Concurrent Object Oriented Languages

Monitors

https://wiki.cse.yorku.ca/course/6490A

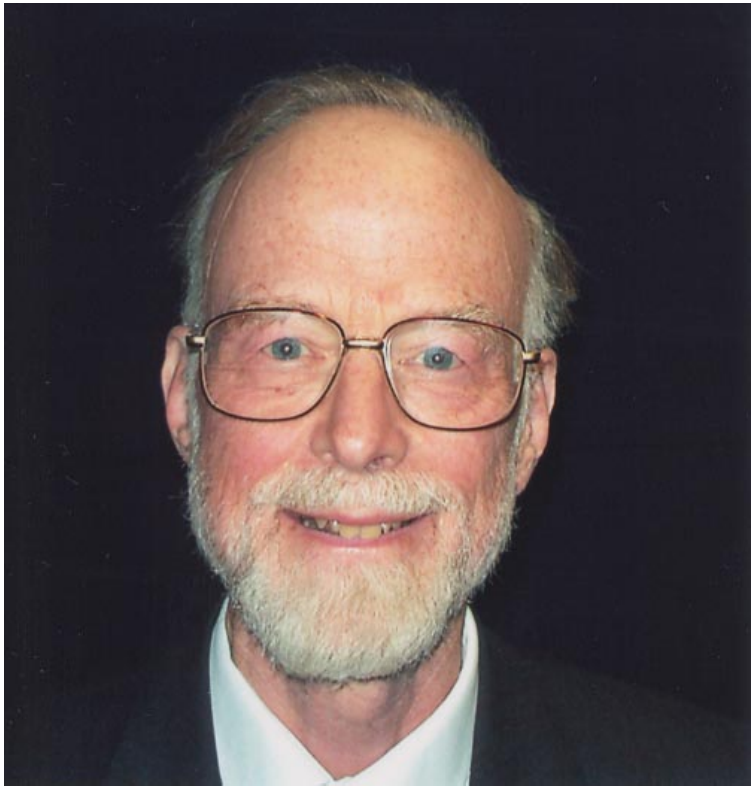
Monitors

Monitors were invented by Tony Hoare and Per Brinch Hansen.

C.A.R. Hoare. Monitors: an operating system structuring concept. *Communications of the ACM*, 17(10):549.557, October 1974.

sir Charles Antony Richard (Tony) Hoare

- Fellow of the Royal Society (1982)
- Fellow of the Royal Academy of Engineering (2005)
- Recipient of the Turing Award (1980)

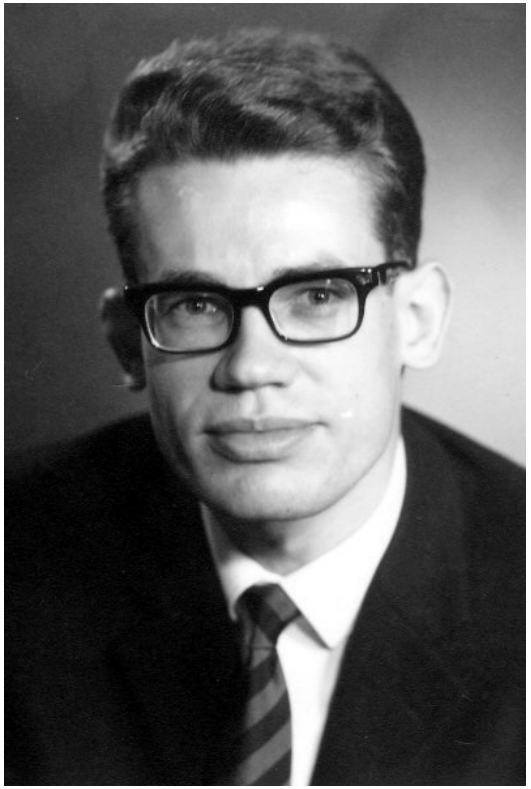


Tony Hoare

source: cs.ox.ac.uk

Per Brinch Hansen

- IEEE Computer Pioneer Award (2002)



Per Brinch Hansen (1938–2007)

source: Per Brinch Hansen

Monitors

A *monitor* consists of

- data: variables and their initialization
- procedures

The variables can only be accessed within the monitor and, at any moment, at most one thread can be executing a procedure of a monitor.

Data Races

A data race occurs when two or more threads access the same memory location concurrently, and. at least one of the accesses is for writing.

There cannot be any data races on the variables of a monitor.

A Simple Example

```
Counter : monitor
begin
  value : int;
  procedure increment(result number : int)
  begin
    value := value + 1;
    number := value;
  end
  procedure decrement(result number : int)
  begin
    value := value - 1;
    number := value;
  end
  value := 0;
end
```

A Simple Example

Question

One thread executes

```
x : int; Counter.increment(x);
```

and another thread executes

```
int y; Counter.decrement(y);
```

What are the final values of x and y?

A Simple Example

Question

One thread executes

```
x : int; Counter.increment(x);
```

and another thread executes

```
int y; Counter.decrement(y);
```

What are the final values of x and y?

Answer

1 and 0 or 0 and -1.

Synchronization

A *condition variable* can be thought of as an event that has no value.

On a condition variable we perform the following operations.

- The wait operation
  - is issued inside a procedure of the monitor, and
  - causes the calling thread to be delayed.
- The signal operation
  - is issued inside a procedure of the monitor, and
  - causes exactly one of the waiting threads to be resumed (if there are no waiting program, the operation has no effect).

A Simple Example

```
Resource : monitor
begin

 procedure acquire()
 begin

 end
 procedure release()
 begin

 end

end
```

A Semaphore

```
Semaphore : monitor
begin

 procedure P()
 begin

 end
 procedure V()
 begin

 end

end
```

The Consumer-Producer Problem

```
BoundedBuffer : monitor
begin
    N : int;
    buffer : int[];
    next : int;
    size : int;
    procedure put(value : int)
    begin

    end
    procedure get(result value : int)
    begin

    end
    N := 10;
    next := 0;
    size := 0;
```

The Consumer-Producer Problem

```
procedure put(value : int)
begin
  buffer[next] := value;
  size := size + 1;
  next := (next + 1) mod N;
end
```

The Consumer-Producer Problem

```
procedure get(result value : int)
begin
  value := buffer[(next - size) mod N];
  size := size - 1;
end
```

The Readers-Writers Problem

```
ReadersAndWriters : monitor
begin

  procedure startRead()
  begin

  end
  procedure stopRead()
  begin

  end
  procedure startWrite()
  begin

  end
  procedure stopWrite()
  begin
```

The Dining Philosophers Problem

```
Table : monitor
begin

 procedure getForks(int i)
 begin

 end
 procedure putForks(int i)
 begin

 end

end
```