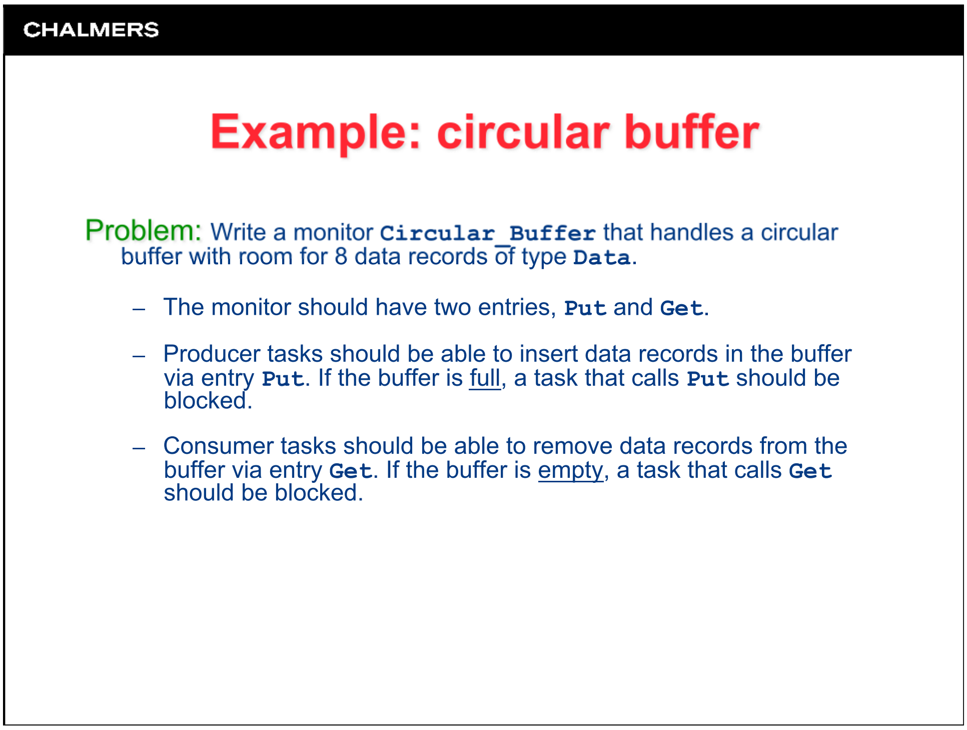
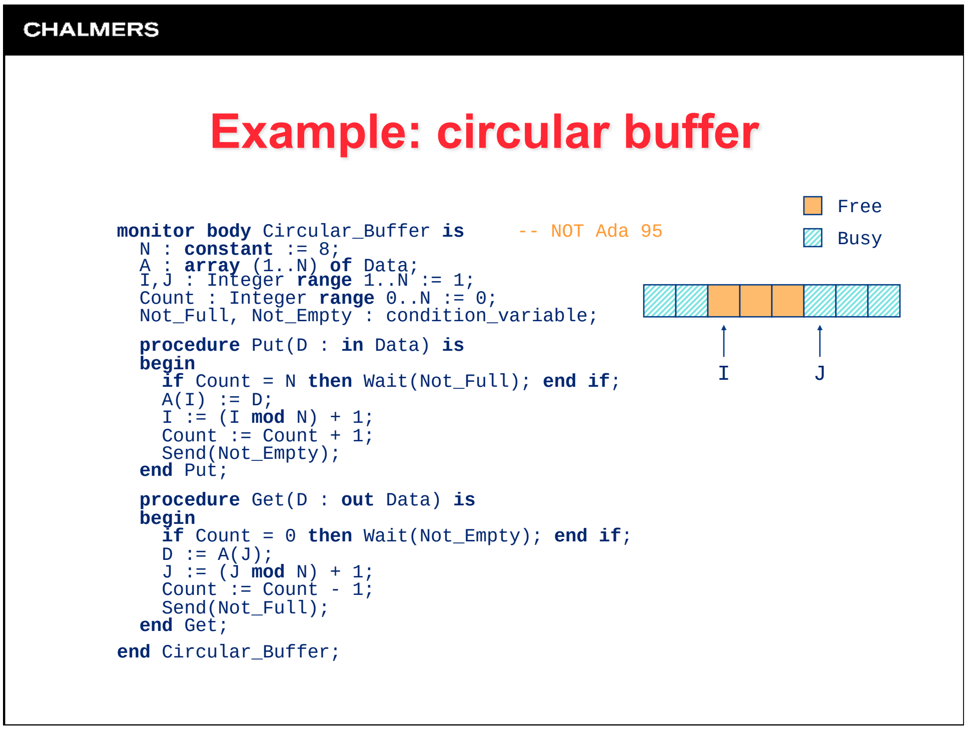


CHALMERS

Example: circular buffer

Problem: Write a monitor **Circular_Buffer** that handles a circular buffer with room for 8 data records of type **Data**.

- The monitor should have two entries, **Put** and **Get**.
- Producer tasks should be able to insert data records in the buffer via entry **Put**. If the buffer is full, a task that calls **Put** should be blocked.
- Consumer tasks should be able to remove data records from the buffer via entry **Get**. If the buffer is empty, a task that calls **Get** should be blocked.



CHALMERS

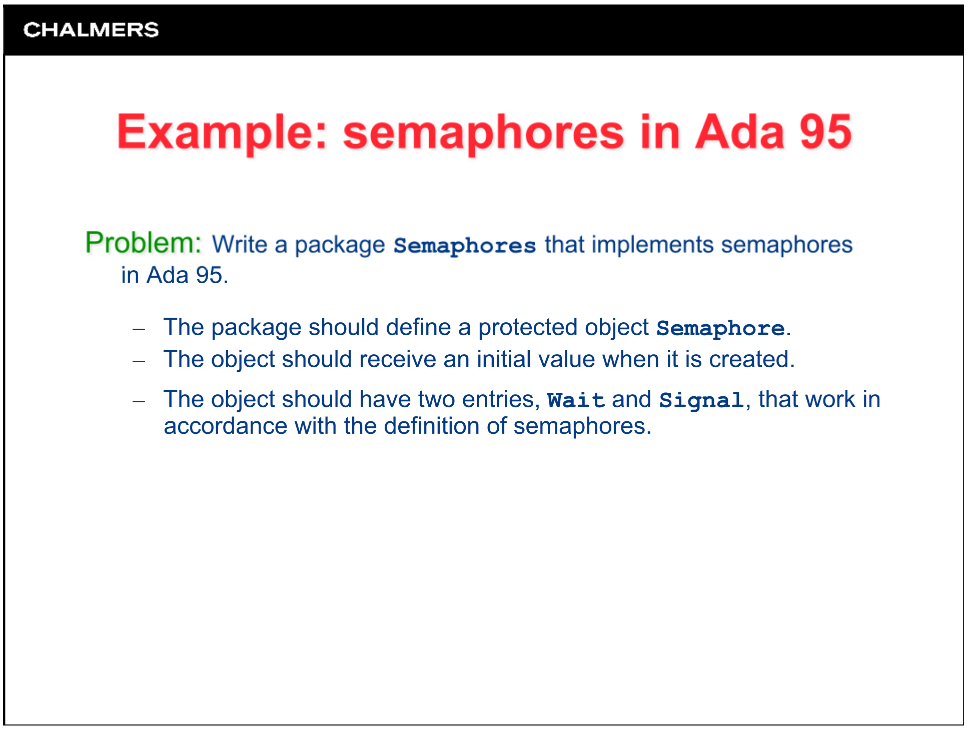
Example: circular buffer

```
monitor body Circular_Buffer is      -- NOT Ada 95
  N : constant := 8;
  A : array (1..N) of Data;
  I,J : Integer range 1..N := 1;
  Count : Integer range 0..N := 0;
  Not_Full, Not_Empty : condition_variable;

  procedure Put(D : in Data) is
  begin
    if Count = N then Wait(Not_Full); end if;
    A(I) := D;
    I := (I mod N) + 1;
    Count := Count + 1;
    Send(Not_Empty);
  end Put;

  procedure Get(D : out Data) is
  begin
    if Count = 0 then Wait(Not_Empty); end if;
    D := A(J);
    J := (J mod N) + 1;
    Count := Count - 1;
    Send(Not_Full);
  end Get;
end Circular_Buffer;
```

🟧 Free
▨ Busy

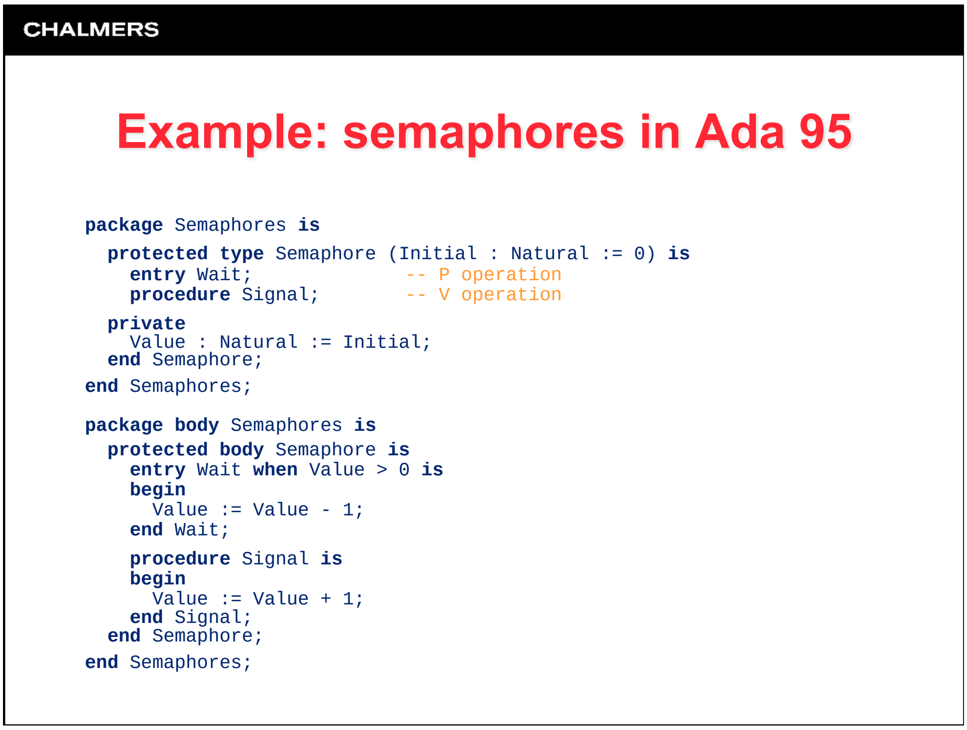


CHALMERS

Example: semaphores in Ada 95

Problem: Write a package **Semaphores** that implements semaphores in Ada 95.

- The package should define a protected object **Semaphore**.
- The object should receive an initial value when it is created.
- The object should have two entries, **Wait** and **Signal**, that work in accordance with the definition of semaphores.



CHALMERS

Example: semaphores in Ada 95

```
package Semaphores is
  protected type Semaphore (Initial : Natural := 0) is
    entry Wait;              -- P operation
    procedure Signal;        -- V operation
  private
    Value : Natural := Initial;
  end Semaphore;
end Semaphores;

package body Semaphores is
  protected body Semaphore is
    entry Wait when Value > 0 is
    begin
      Value := Value - 1;
    end Wait;
    procedure Signal is
    begin
      Value := Value + 1;
    end Signal;
  end Semaphore;
end Semaphores;
```