Objects

- A single object represents both a value and a set of operations
- Example interface of a stack object:

```
S={NewStack}
{S push(X)}
{S pop(X)}
{S isEmpty(B)}
```

- The stack value is stored inside the object S
- Example use of a stack object:

```
S={NewStack}
{S push(a)}
{S push(b)}
local X in {S pop(X)} {Browse X} end
```

Implementing the stack object

- Implementation of the stack object:

```
fun {NewStack}
          C={NewCell nil}
          proc {Push X} C:=X|@C end
          proc {Pop X} S=@C in C:=S.2 X=S.1 end
          proc {IsEmpty B} B=(@C==nil) end
in
          proc {$ M}
               case M of push(X) then {Push X}
               [] pop(X) then {Pop X}
               [] isEmpty(B) then {IsEmpty B} end
          end
end
```

- Each call to NewStack creates a new stack object
- The object is represented by a one-argument procedure that does procedure dispatching: a case statement chooses the operation to execute
- Encapsulation is enforced by hiding the cell with static scoping

Stack as ADT and stack as object

- Here is the stack as ADT:

```
local Wrap Unwrap in
        {NewWrapper Wrap Unwrap}
        fun {NewStack} {Wrap nil} end
        fun {Push W X} {Wrap X|{Unwrap W}} end
        fun {Pop W X} S={Unwrap W} in X=S.1 {Wrap S.2} end
        fun {IsEmpty W} {Unwrap W}==nil end
end
```

- Here is the stack as object: (represented by a record)

```
fun {NewStack}
        C={NewCell nil}
        proc {Push X} C:=X|@C end
        proc {Pop X} S=@C in X=S.1 C:=S.2 end
        fun {IsEmpty} @C==nil end
in
        stack(push:Push pop:Pop isEmpty:IsEmpty)
end
```

- Any data abstraction can be implemented as an ADT or as an object

Final remarks on objects

- Objects are omnipresent in computing today
- The first major object-oriented language was Simula-67, introduced in 1967
  - It directly influenced Smalltalk (starting in 1971) and C++ (starting in 1979), and through them, most modern object-oriented languages (Java, C#, Python, Ruby, and so forth)
- Most modern OO languages are in fact data abstraction languages: they incorporate both objects and ADTs
  - And other data abstraction concepts as well, such as components and modules
- The next lesson will be completely focused on object-oriented programming