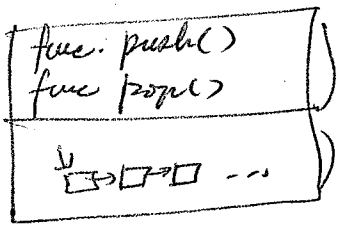
Ch. 11, 12 | Abstract data types / data encapsulation
O.O.P.

- Grouping of data and operations

  - procedure / func.
    - all details are accessible
    - represents an action (operation)
  - module / package
    - interface: public
    - implementation: private
    - ◦ state partition of a program
    - ◦ hide representation (private part)
    - ◦ check access — import/export — Modula-2, Ada
    - ◦ represents a collection of data and related actions (operations)
  - class / struct (defined types)
    - ◦ hides representation (private part)
    - ◦ check access
    - → objects are initialized upon creation
    - ◦ represents a collection of data and related actions (operations)
    - → dynamic objects at run time — create / delete.

- class — class (type) of objects
- object — run time entity with data (on which operation is performed)

- what an object does is independent of how it works.



ex) stack

ex) array, linked list, ...

— information hiding.

— C++ Struct/class

class — generalization of struct

struct — class with only public members (data, func.)

by default,
- struct — all members are public
- class — all members are private

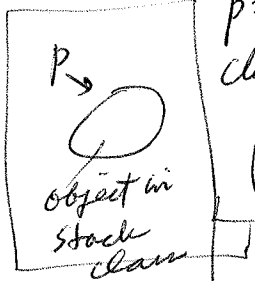
— class/struct name can be used as a type.

```
// struct stack
{ ...
};
stack s1;
```

```
class stack
{ ...
  char pop();
...
};
stack s1;
char c = s1.pop();
```

or, using pointer

```
stack *p;
p = new stack;
char c = p->pop();
   (or (*p).pop();)
delete p;
```



- constructor — initialization code
- destructor — cleans up dynamic mem. allocated to the object.

— initialization with parameters

— overloaded function names

- imperative language
- + data abstraction (encapsulation of data + operations)

⇒ modules/packages (Ada)

- definition part (public)
- implementation (private)

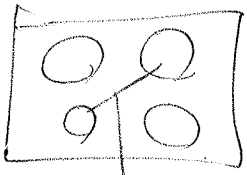
Accessibility — Export/import

Separate interface (logical) and implementation (physical)

- + automatic initialization/finalization (of values of given types (class))
- + extension (inheritance) ex). derived class — C++
  extended class — Java

⇒ class/objects

we can create any # of instances (objects) of an abstract data type (class)



at most 1 object at a time can be executed ~~at~~ (↔ concurrent prog.)

communication — via func call
member

— membership of a class

- public members — accessible to outside code
- private — accessible to members of the class, but not visible to derived class
- protected — private, but visible to derived class

C++ class stack
{ public:
    stack() {...}
    char pop();
    void push(char);
  private:
    int top;
    char elements[100]; — hidden
};

— friend class

class A
{ ...
  friend class B;
  ...
};

⇒ class B can access private members of A.

— in-line expansion

member functions definition in the class — are expanded in-line at compile time.

— function call statement is substituted with the body of the function definition

macro-expansion

— Inheritance

C++, derived class.

⇒ class list

{ ... } — by default, private; invisible to the members of the derived classes. (but visible to inherited members in the derived classes).

public:
...

protected:
...) — private, but visible only to derived classes.

};

class Queue = public list — base class

{
...
};

members of the base (parent) class retain their accessibility in the derived class

class stack: private list — base class

{
...
};

makes all inherited members private in the derived class.

(ex) class Base

{ A ——— private (by default)
  public: B
  protected: C
}

member — either data/func.

class Derived : public Base

{ D
  E — can accen B, C
      cannot accen A (∵ private)
}

⇒ members of Derived :

public : B } inherited from Base
protected : C }
private : A, D, E
(A: inherited; D, E: new)

- outsiders/any one can accen Derived.B — public ;
- only derived class members (D, E) can accen C (protected);
- only B, C can accen A (∵ B, C are inherited members) ;
- friend class (of Derived) can access any members, i.e., B, C, A, D, E.

class Derived2 : private Base

{ D
  E — cannot accen A, B, C
}

⇒ members of Derived2 :

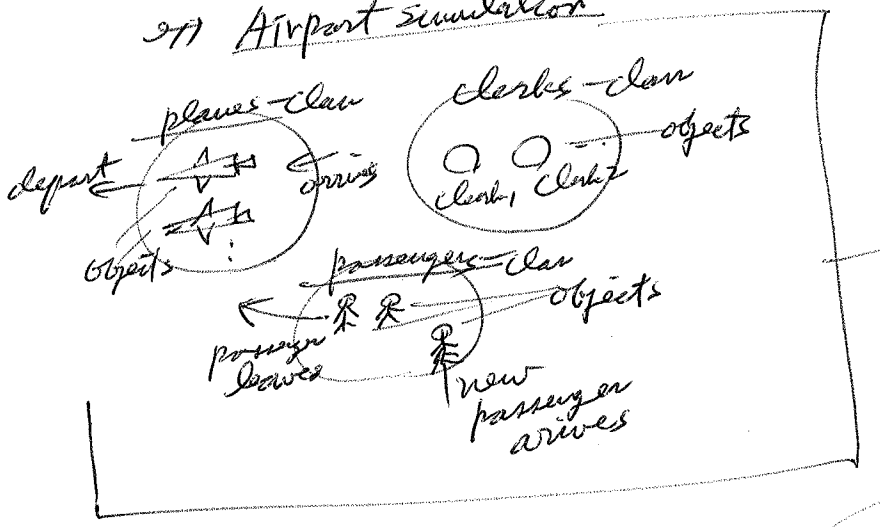
public : ∄
protected : ∄
private : A, B, C, D, E — only friend class can accen all of these.
(A, B, C: inherited; D, E: new)

- derived class can use (accen / call) constructor/destructor of the base class, but the constructor cannot be a member of the derived class.

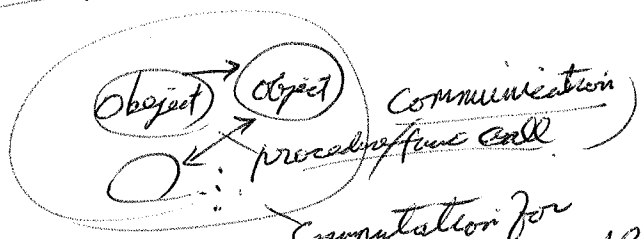
O.O.P.

- Simulation of an application
- Object is an entity in the solution to a problem.

eg) Airport simulation



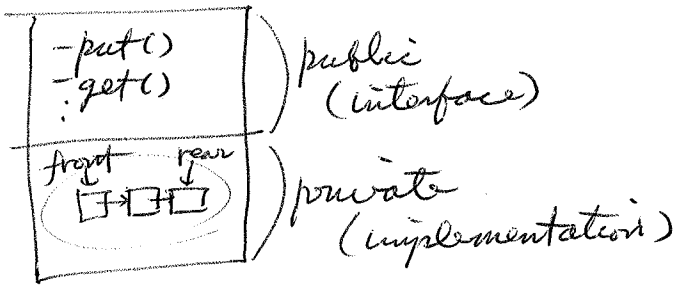
class : set (type) of objects



- information hiding

what an object does is independent from how it does.

(interface (operations)) (implementation)



procedure/func
↓ + data encapsulation
module/package — (static)
↓ + (auto initialization/finalization of values of given types
extension of data abstraction with new op's.
class/struct — (dynamic)

info. hiding
- public — interface
- private — implementation

abstract types (class)

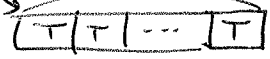
*) derived class — C++
extended class — Java

— support for O.O.P.
- inheritance — *) base/derived class — C++
  super class/sub-class — Java
- polymorphism
  - template class
  - overloaded func. names

*) C++

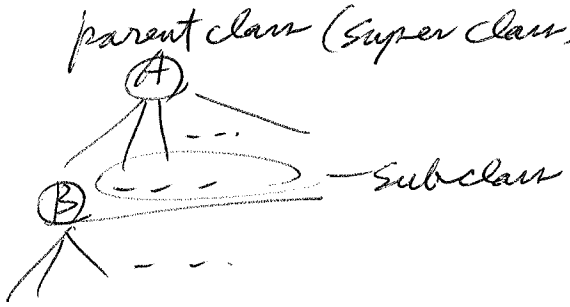
```
template <class T> class stack        // T: optype
{
  T* elements;
  public:
    stack(int n) // constructor
    { elements = new T[n]; }          // in-line expansion
    void push(T a) { top++; elements[top] = a; }
    T pop() {----}
      :
};
:
stack<int> S1(10);
stack<char> S2(20);
```

elements: pointer to type T; n



(para. to open type T. para. to constructor.

9) Java class hierarchy



— class B extends A

[{ ≡≡≡ } ← public / protected / private) members Same as C++.

— inner class

class in the class definition.

(Clients of outer class (main class) cannot access inner class).

ex) public class Stack // client of class Node.

{ private class node — // inner class — (invisible to the clients of class Stack.

{ int val;
node next;
node (int v; node n) {val = v; next = n;}
};

— private node stack = null; — stack →

— public int pop()

{ int result = stack.val;
stack = stack.next;
return result;
}

— public void push (int v)

{ stack = new node (v, stack);
}
}

int (val) node (next)

? Should use different name from class name

accessible to the clients of stack

1 push(v)

stack → [v | —] ...

1 push (v2)

stack → [v2 | —] → [v | —] ...

ref

C++ — copy semantics for arrays/objects

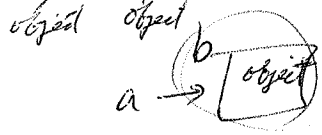
$a \leftarrow b$

(Copies value of b into a.)

Java
- copy semantic for primitive types
- reference semantics for arrays/objects

$a \leftarrow b$ means — object assignment

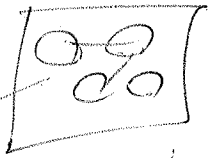
(a: object assigned, b: object)



a and b reference the same object.

⇒ any change to a means change to b. too.

O.O.P. VS. Concurrent prog.

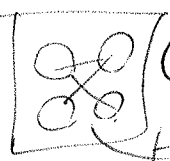


objects are controlled by the centralized way

comm. by member func call

- at most 1 object at a time can be executed.

- no central control for ordering the sequence of operation
- objects (parallel entity) work concurrently.



objects (Threads)

- Communication
- Synchronization