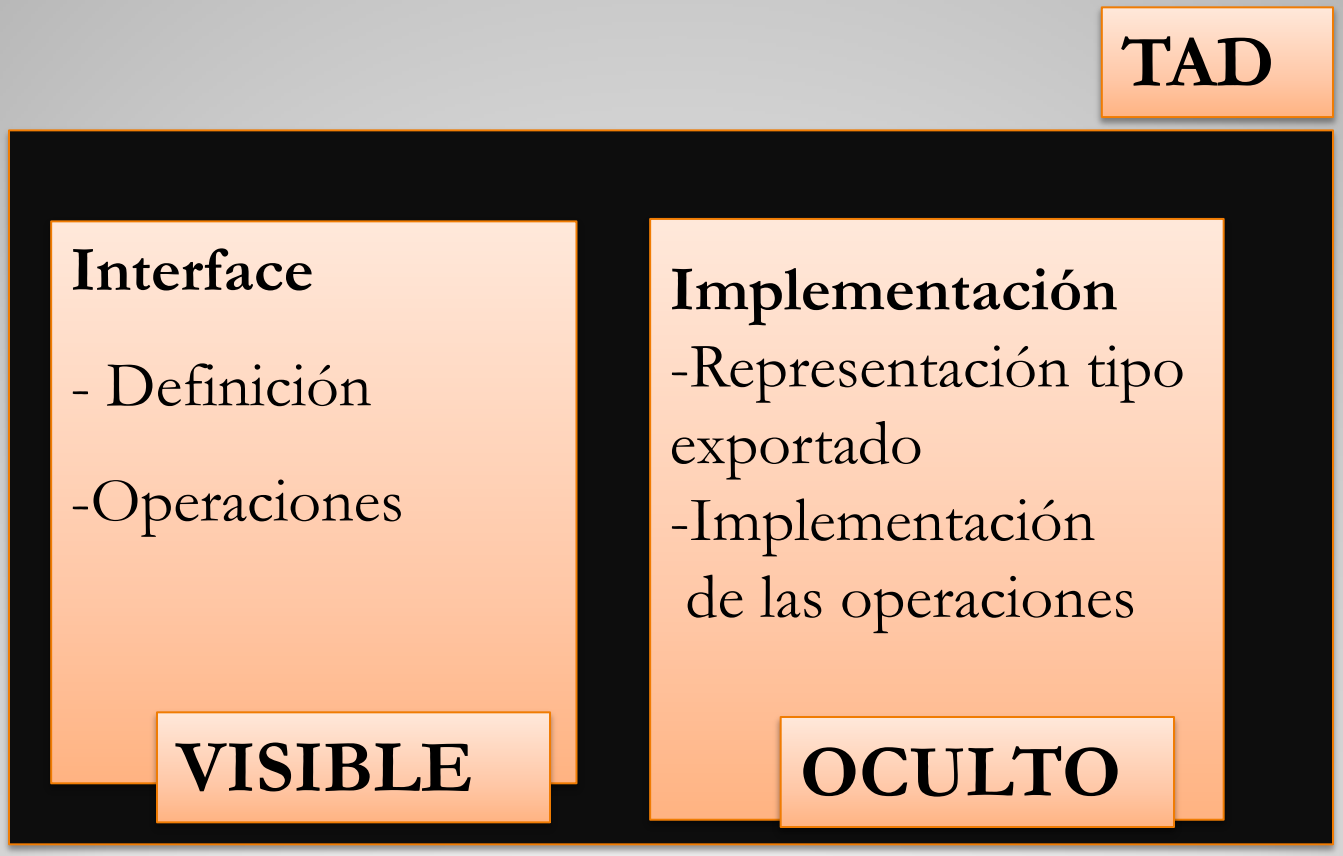
Programación II - 2014

TAD

Recordamos


TAD
Interface
- Definición
-Operaciones
VISIBLE
Implementación
-Representación tipo exportado
-Implementación de las operaciones
OCULTO

TAD Micro

Se desea diseñar el TAD Micro que representa un transporte de pasajeros que tiene un nombre determinado y se dirige a un lugar particular. Este transporte dispone de 40 asientos que pueden ocuparse parcial o totalmente.

Representar el TAD Micro, e implementar las operaciones de crear, mostrar el destino del micro, mostrar el nombre del micro, asignar un micro a otro, ocupar un asiento y consultar si el micro está completo.

Tad Micros

interface

type exportado micro;

Procedure crear (var m:micro; nom,dest: string);

Procedure mostrarDestino (m:micro; var dest:string);

Procedure mostrarNombre (m:micro; var nom: string);

Procedure asignar (var m1:micro; m2: micro);

{asigna el valor de m2 a m1}

Procedure ocuparAsiento (var m:micro);

Function microCompleto (m:micro): boolean;

Definición del tipo Exportado

```
implementation
  asientos = array [1..40] of boolean;
  micro = record
              asiento: asientos;
              nombre: string,
              destino : string;
              completo: 0..40;
            end;
```

Implementación de Operaciones

```
Procedure crear (var m:micro; nom,dest: string);
  begin
     m.nombre := nom;
     m.destino := dest;
     m.completo := 0;
     for i:= 1 to 40 do
         m.asiento[i] := false
  end;
```

Implementación de Operaciones

```
Procedure mostrarDestino (m:micro; var dest:string);
  begin
     dest := m.destino;
  end;

Procedure mostrarNombre (m:micro; var nom:string);
  begin
     nom:= m.nombre;
  end;
```

Implementación de Operaciones

```
Procedure asignar (var m1:micro; m2: micro);
{asigna el valor de m2 a m1}
 begin
    m1:= m2;
 end;
```

Implementación de Operaciones

```
Procedure ocuparAsiento (var m:micro);
  begin
        m.asiento [m.completo+1] := true;
        m.completo := m.completo + 1;
end;

Function microCompleto (m:micro): boolean;
begin
     if m.completo = 40 then MicroCompleto := true
                        else  MicroCompleto := false;
end;
```

Usando el TAD creado

Se pide realizar un programa que:
Genere una estructura donde guardar la información de todos los micros que salen de la Terminal de La Plata un día determinado.
Recorra la estructura e informe el nombre y destino de los micros que partieron completos.

```
Program Terminal;
  uses micros;
  type
   lista = ^nodo;
   nodo = record
                m: micro; {micro es el tipo exportado del TAD micros}
                sig : lista;
              end;
var
   L : lista;
   mi : micro;
   nombre, destino : string;
```

Indicamos que vamos a utilizar el TAD micros

```
Begin
L := nil;
writeln(“ Ingrese el nombre”)
read (nombre);
while (nombre <> `ZZZ`) do begin
        read (destino);
        crear (mi, nombre, destino);
        CargarAsientos(mi);
        GuardarLista (L, mi);
        writeln(“Ingrese el nombre”)
        read (nombre);
   end;
   RecorrerLista (L);
End.
```

```
Procedure CargarAsientos (var mi: micro);
 var
     asiento: char;
 begin
  writeln(" ¿Necesita un asiento libre? (Ingrese S o N)")
  read (asiento);
  while (asiento<>"N") and (not MicroCompleto (mi))
     do begin
        ocuparAsiento (mi);
        writeln(" ¿Necesita un asiento libre? (Ingrese S o N)")
        read (asiento)
     end;
   end;
```

```
Procedure GuardarLista (var L: lista; mi: micro);
 var
    nue: lista;
 begin
     new (nue);
     asignar (nue^.m, mi);
     nue^.sig := L;
     L := nue;
 end;
```

```
Procedure RecorrerLista ( L: lista);
  var
     mi : micro;
     nom, dest : string;
 begin
  while ( L <> nil) do begin
    asignar (mi, L^.m);
    if MicroCompleto (mi) then begin
         mostrarNombre (mi, nom);
         mostrarDestino (mi, dest);
         writeln (´Nombre: ´, nom)
         writeln (´Destino: ´, dest);
    end;
    L := L^.sig;
  end;
end;
```

Se puede evitar usar la operación Asignar y directamente escribir:

MicroCompleto(L^.m)