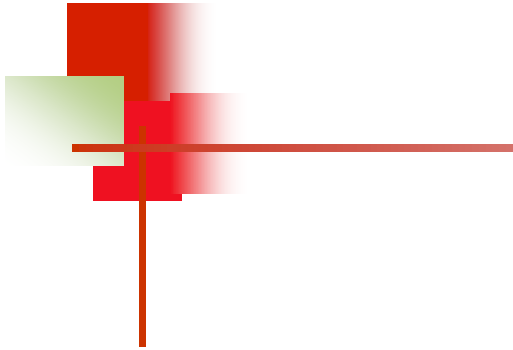


Teoría

Introducción a la Programación Orientada a Objetos (POO)

Programación Orientada a Objetos - Colecciones

Una característica importante de los lenguajes que utilizan la POO es la de permitir las colecciones genéricas:

- Arreglos genéricos
- Listas genéricas
- Arboles genéricos

¿Cómo sería una lista genérica?

Programación Orientada a Objetos - Colecciones

```
Clase Lista;
   Lis = ^nodo;
   nodo = record
       elemento : object;
       sig : lis;
   end;

   primero, actual : lis;
   cantidad : integer;

    Constructor crear();
        begin
            primero := nil;
            cantidad := 0;
        end;
```

¿De qué tipo de dato es elemento?

Programación Orientada a Objetos - Colecciones

```
procedure inicializar();
    begin
       actual := primero;
    end;

procedure avanzar();
    begin
        actual := actual^.sig;
     end;

procedure devolverElemento (var unObjeto : object)
     begin
         unObjeto := actual^.elemento;
     end;
```

Programación Orientada a Objetos - Colecciones

```
procedure agregarElemento (unObjeto : object)

   var aux:lis

    begin
        new(aux);
        aux^.elemento := unObjeto;
        aux^.sig := primero
        primero := aux;
        cantidad := cantidad + 1;
    end;
```

Programación Orientada a Objetos - Colecciones

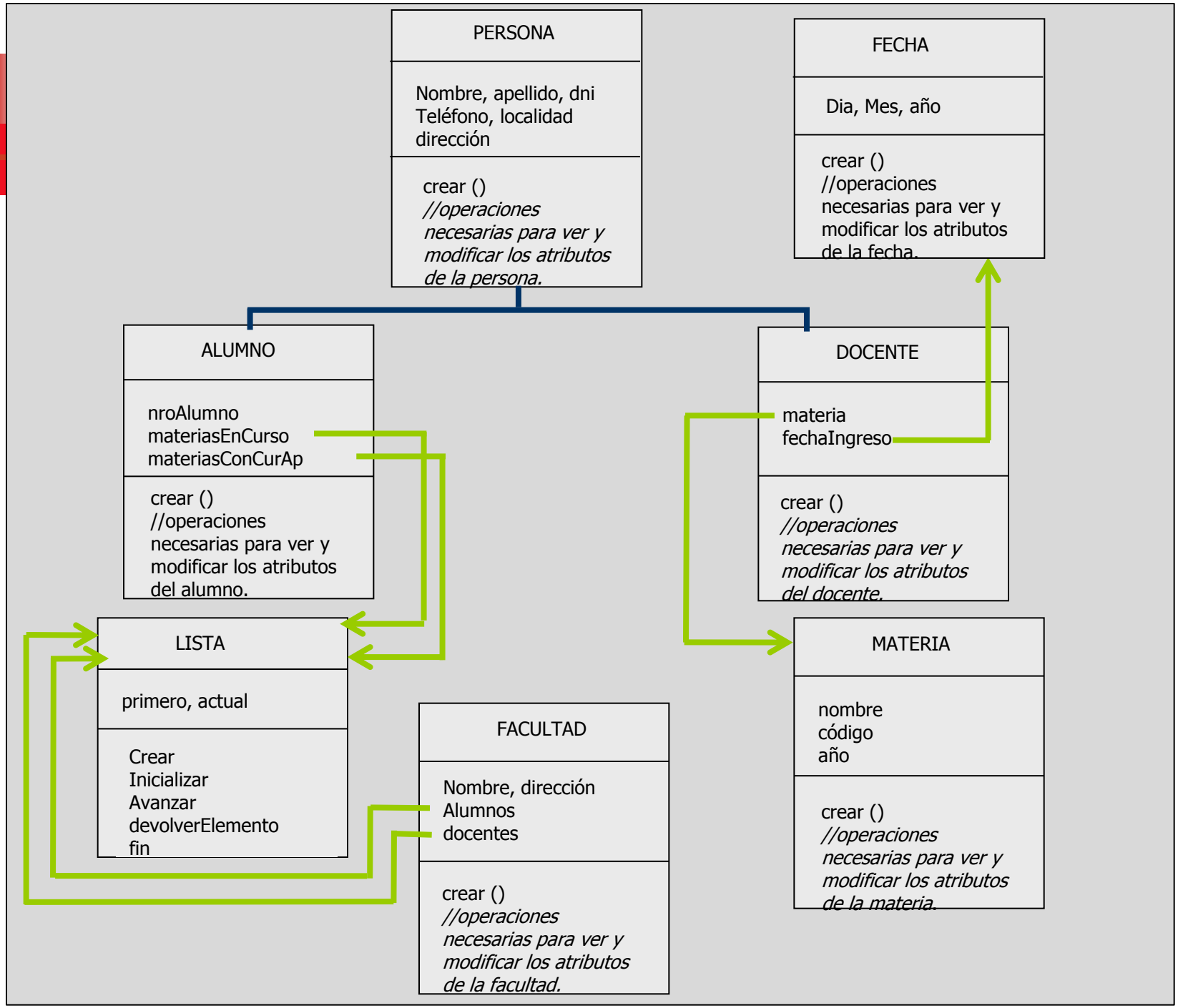
```
    function fin : boolean;
        begin
            fin := (actual = nil);
        end;

    function verCantidad : integer;
        begin
            verCantidad := cantidad;
        end;

End;  {fin de la clase Lista}
```

Programación Orientada a Objetos - Ejemplo

Se desea modelar una Facultad. En la misma existen docentes que dictan materias (cada docente dicta una materia). Además a la Facultad concurren alumnos los cuales mantienen el conjunto de materias de las cuales ya han aprobado su cursada y aquellas que están cursando.

Se pide modelar la Facultad y agregar lo necesario para:

- Inscribir un alumno a la Facultad.
- Informar el docente con mayor antigüedad


PERSONA
Nombre, apellido, dni
Teléfono, localidad
dirección
crear ()
//operaciones
necesarias para ver y
modificar los atributos
de la persona.
FECHA
Dia, Mes, año
crear ()
//operaciones
necesarias para ver y
modificar los atributos
de la fecha.
ALUMNO
nroAlumno
materiasEnCurso
materiasConCurAp
crear ()
//operaciones
necesarias para ver y
modificar los atributos
del alumno.
DOCENTE
materia
fechaIngreso
crear ()
//operaciones
necesarias para ver y
modificar los atributos
del docente.
LISTA
primero, actual
Crear
Inicializar
Avanzar
devolverElemento
fin
FACULTAD
Nombre, dirección
Alumnos
docentes
crear ()
//operaciones
necesarias para ver y
modificar los atributos
de la facultad.
MATERIA
nombre
código
año
crear ()
//operaciones
necesarias para ver y
modificar los atributos
de la materia.

Programación Orientada a Objetos - Ejemplo

```
Clase Fecha:
   dia, mes, año : integer;

   Constructor crear (unDia, unMes, unAño : integer)
      begin
         dia := unDia;
         mes := unMes;
         año := unAño;
      end;

{todas las operaciones para ver y modificar los atributos de la clase fecha}

     function diferenciaDias (f : Fecha) : integer
        var numActual, numf : integer
     begin
           numActual := (año - 1900) * 365 + mes*30 + dia;
           numf := (f.verAño() - 1900) * 365 + f.verMes() *30 + f.verDia();
           diferenciaDias:= abs (numActual - numf);
        end;
          Terminación de la Clase Fecha
End;
```

Programación Orientada a Objetos - Ejemplo

```
Clase Materia;
   nombre : string;
   codigo : integer;
   año : integer;

   Constructor crear (unNombre:string, unCodigo:integer, unAño:integer)
      begin
        nombre := unNombre;
        codigo := unCodigo;
        año := unAño;
      end;

{todas las operaciones para ver y modificar los atributos de la clase materia}

End;
```

Terminación de la Clase Materia

Programación Orientada a Objetos - Ejemplo

```
Clase Persona;
      nombre : string;
      dni : string;
      telefono : string;
      localidad : string;
      dirección : string;

Constructor crear (unNombre, unDni, unaLocalidad, unaDir, unTelefono : string);
     begin
          nombre := unNombre;
          dni := unDni;
          localidad := unaLocalidad;
          direccion := unaDir;
          telefono := unTelefono;
     end;

{todas las operaciones para ver y modificar los atributos de la persona}
End;    Terminación de la Clase Persona
```

Programación Orientada a Objetos - Ejemplo

Clase Alumno (Persona);

```
numAlumno : integer;
materiasEnCurso : Lista [Materia];
materiasConCurAp : Lista [Materia];

Constructor crear (unNumero : integer; unNombre, unDni, unaLocalidad, unaDir,
                   unTelefono : string);
 begin
    numAlumno := unNumero;
    super.crear(unNombre, unDni, unaLocalidad, unaDir, unTelefono);
    materiasEnCurso := Lista.Crear;
    materiasConCurAp := Lista.Crear;
 end;
```

Programación Orientada a Objetos - Ejemplo

```
procedure agregarMatACursar (unaMateria : materia)
    begin
        materiasEnCurso.agregarelemento (unaMateria);
    end;

procedure aprobarMateria (unaMateria : materia)
    begin
        materiasEnCurso.borrarelemento (unaMateria);
        materiasConCurAp.agregarelemento (unaMateria);
    end;

{todas las operaciones para ver y modificar los atributos del alumno}
End;
```

Terminación de la Clase Alumno

Programación Orientada a Objetos - Ejemplo

```
Clase Docente (Persona);
    materia:Materia;
    fechaIngreso: Fecha;

Constructor crear (unNombre, unDni, unaLocalidad, unaDir, unTelefono :
                          string; unaFechaIng : Fecha; unaMateria : Materia);
  begin
      materia := unaMateria;
      fechaIngreso := unaFechaIng;
      super.crear (unNombre, unDni, unaLocalidad, unaDir, unTelefono);
  end;

{todas las operaciones para ver y modificar los atributos del docente}
End;
```

Terminación de la Clase Docente

Programación Orientada a Objetos - Ejemplo

```
Clase Facultad;
   nombre : string;
   direccion : string
   alumnos : Lista [Alumno];
   docentes : Lista [Docente];

Constructor crear (unNombre: string, unaDir: string)
     begin
         nombre := unNombre;
         direccion := unaDir;
         alumnos := Lista.Crear;
         docentes := Lista.Crear;
     end;
```

Programación Orientada a Objetos - Ejemplo

```
procedure inscribirAlumno(a : Alumno);
      begin
         alumnos.agregarelemento (a);
      end;

function cantidadAlumnos:integer;
     begin
        cantidadAlumnos:= alumnos.VerCantidad;
     end;
```

Agregamos el NUEVO alumno a la colección.

Programación Orientada a Objetos - Ejemplo

```
procedure docenteMasAntiguo(var nom : string; fe : Fecha);
     Var max, dif :integer;  unDocente : Docente; fI : Fecha;
Begin
       max:= 0;
       docentes.inicializar;
       While (not docentes.fin) do Begin
              docentes.devolverElemento (unDocente);
              unDocente.verFechaIngreso (fI);
              dif:= fe.diferenciaDias (fI);
              if (dif > max) then begin
                                    unDocente.verNombre(nom);
                                    max := dif ;
                                  end;
               docentes.avanzar;
       End;
End;
{todas las operaciones para ver y modificar los atributos de la Facultad}
End;
```

Terminación de la Clase Facultad

Programación Orientada a Objetos - Ejemplo

```
Program Facultad_De_Informatica;
    Var  fac: Facultad;  fe: Fecha;
         nom, dni, dir, loc, tel : string;
         dia, mes, año: integer;
Begin
     fac := Facultad.crear("Informática", "50 y 120");

     read (nom); read (dni); read (loc); read (dir); read(tel);

     alu := Alumno.crear (fac.cantidadAlumnos + 1, nom, dni, loc, dir, tel);

     fac.inscribirAlumno (alu);

     read (dia, mes, año);
     fe := Fecha.crear(dia, mes, año);

     fac.docenteMasAntiguo(nom, fe);
     Write (nom) ;
 End.
```

Datos de un Alumno

Inscripción del Alumno en la Facultad

Nombre del Docente más antiguo de la Facultad