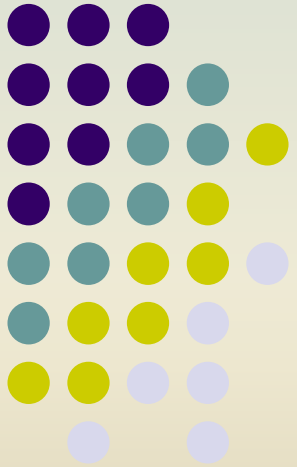
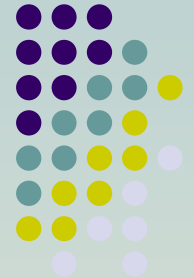
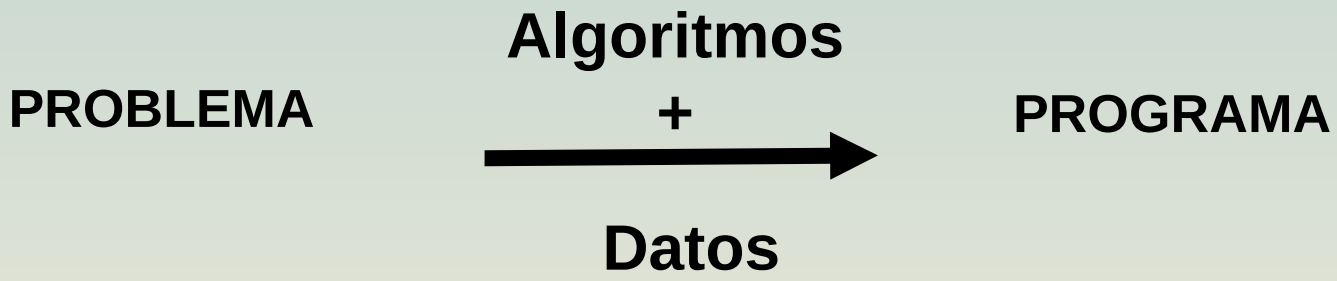
Sem. Programación

INTRODUCCION A LA PROGRAMACION ORIENTADA A OBJETOS



Problemas, programas, algoritmos y estructuras de datos





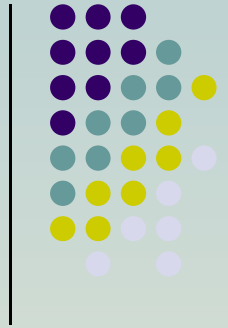
Problema: Conjunto de hechos o circunstancias que no permiten cumplir con un objetivo.

Algoritmo: Especificación rigurosa de una secuencia de pasos (instrucciones) con significado preciso y que se puede ejecutar con una cantidad finita de recursos en un tiempo finito.

Datos: Representación de un objeto del mundo real mediante el cual se pueden abstraer aspectos de un problema que se desea resolver.

Programa: Conjunto de instrucciones, ejecutables sobre una computadora, que permite cumplir una función especifica. (Di giusti)

Concepto Básico de un Objeto



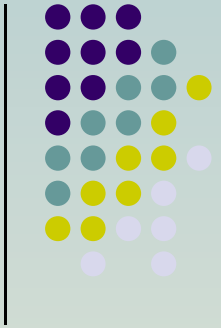
OBJETO → **Datos +** **Comportamiento**

Objeto: Describe un objeto que puede representar algo del mundo real o puede ser un caso abstracto.

Datos: Son los datos que se desean preservar del objeto y que se utilizan para el manejo del objeto.

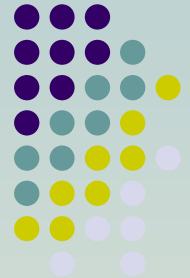
Comportamiento: Conjunto de operaciones que se usan para manipular y administrar los datos del objeto. El comportamiento se puede traducir en funciones y procedimiento dentro del objeto.

Características de la POO



- **Abstracción**
  - Separa el comportamiento del objeto de su implementación
- **Encapsulación**
  - Ocultamiento de información del objeto
- **Modularidad**
  - Módulos cohesivos y débilmente acoplados
- **Jerarquía**
  - Herencia y agregación
- **Polimorfismo**
  - Posibilidad de referenciar distintos objetos de la misma manera
- **Otras propiedades**
  - Concurrencia, persistencia, Genericidad y manejo de excepciones

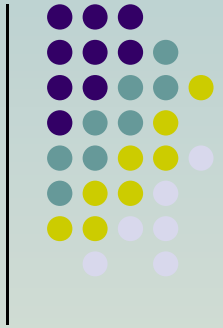
Ejemplos de Definición de Objetos en Delphi



"Los objetos en Delphi se pueden definir como un tipo definido por el usuario dentro del TYPE"

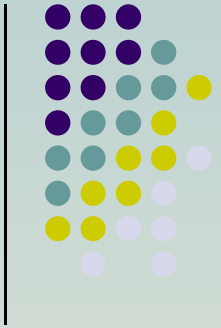
```
TYPE
        Objeto = Object
        Private
                Definición de Datos….(similar a un
                registro)
        Public
                Definición del comportamiento
        End;
```

Ejemplos de Definición de Objetos en Delphi



```
TYPE
        Alumno = Object
        Private
                Legajo:  LongInt;
                Apellido: String;
                Nombres: String;
        Public
                Procedure TomarDatos;
                Function  Alta: Booelan;
                Function  Baja: Boolean;
                Function RetornarDatos: Alumno;
        End;
```

Implementación del Comportamiento del Objeto

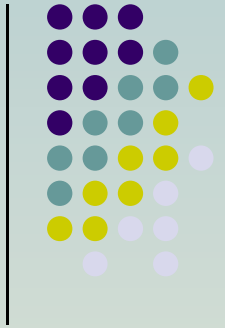


- **En la Sección de Implementación**
  - Procedure Alumno.TomarDatos;
  - Begin
    - Instrucciones

    End;
  - Function Alumno.Alta: boolean;
  - Begin
    - Instrucciones

    ENd;

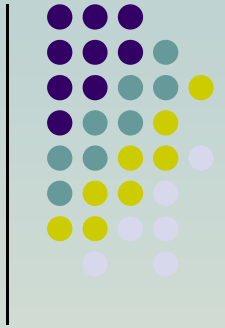
Ejemplo Objeto Abstracto



```
unit LibVectorInt;
interface
  Uses SysUtils, StdCtrls;
Const
  Min = 1;   Max = 10;
Type
  Objeto = Object
      Items: Array[Min..Max] Of Integer;  // Dato o Atributto
      Procedure CargarAlAzar();   // Comportamiento
      Function Maximo(): Integer;
      Procedure Mostrar(Var MM: Tmemo);
  End;

Implementation   // Inicio de la implementación del comportamiento
Procedure Objeto.CargarAlAzar();   // Comportamiento
Var I: Integer;
Begin
      Randomize;
      For I:= Min To Max Do Begin
                Items[I] := Random(100);
      End;
End;
```

Ejemplo Objeto Abstracto

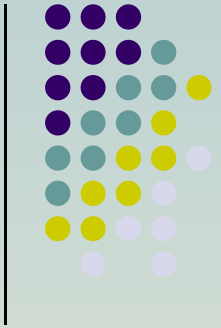


```
Function Objeto.Maximo(): Integer;
Var I, M: Integer;
Begin
      M := Items[Min];
      For I:= Min+1 To Max Do Begin
            If Items[I] > M Then M := Items[I];
      End;
      Maximo := M;
End;

Procedure Objeto.Mostrar(Var MM: Tmemo);
Var I:Integer;
Begin
      MM.Clear;
      For I:= Min To Max Do Begin
            MM.Lines.Add('V[' + inttostr(I) + '] = ' +  inttostr(items[I]));
      End;
End;

end.  // Fin de la implementación
```

Características POO: Modularidad



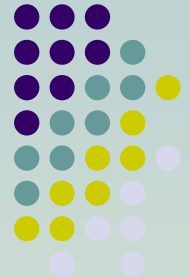
- **Dividir una aplicación en subprogramas (**lo mas independientes posibles)

- **Estructura de un módulo**
  - Interfaz
  - Implementación

- **Reglas de modularización**
  - Manejo de Unidades Modulares
  - Interfaces Adecuadas
  - Interfaces Explicitas
  - Protección de la Información

- **Diseño de Módulos**
  - Acoplamiento de módulos
  - Cohesión de módulos

Abstracción en Lenguajes de Programación



- **Abstracción de Control**
  - Son abstracciones a nivel de sentencias. (For - While - If)
- **Abstracción Procedimental**
  - Se basa en la utilización de funciones o procedimientos que realizan cosas sin importar como lo hacen.
- **Abstracción de Datos**
  - Son los tipos definidos por el usuario
- **Tipos Abstractos de Datos**
  - Separan la “interfaz” de la “implementación”
  - La representación del tipo de dato (est. de datos)
  - Las operaciones (Algorítmos de implementación)

Tipos Abstractos de Datos

- **Ventajas:**
  - **Mejoran la representación**
  - **Mejoran la robustez de la aplicación**
  - **Mejoran el rendimiento**
  - **Separa la implementación de la especificación**
  - **Permiten la extensibilidad del sistema**
  - **Agrupan datos y operaciones**
  - **Permiten ocultamiento de información**