UNIDAD 1

Conceptos y características de la Programación Orientada a Objetos

Ing. Iveth Robles Catari

Contenido

- Fundamentos de programación
- Metodologías
- Metodología Estructurada
- Metodología Modular
- Metodología Orientada a Objetos

FUNDAMENTOS DE PROGRAMACION

Programación

- Se llama Programación a la implementación de un algoritmo en un determinado lenguaje de programación, para realizar un programa.

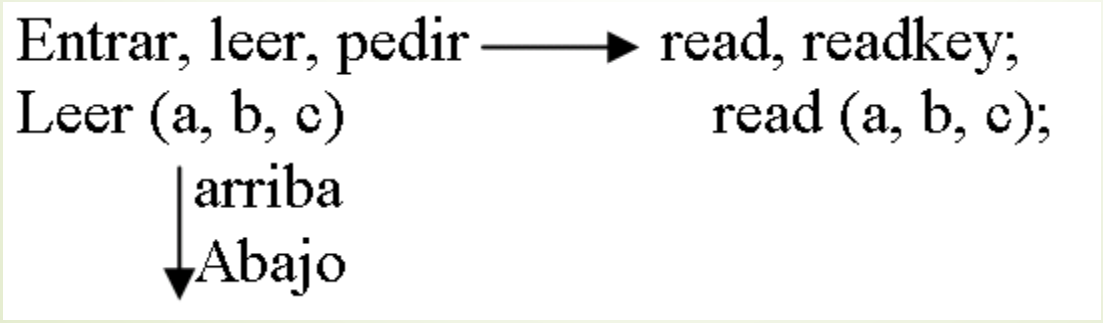
FUNDAMENTOS DE PROGRAMACION

Algoritmo

- "Es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema."

FUNDAMENTOS DE PROGRAMACION

- Todo algoritmo esta sujeto a reglas sintáxicas a un lenguaje de programación formal.
- La tendencia que tienen los algoritmos por lo general es"TOP-DOWN" (de arriba hacia abajo)
  - Ejemplo:



FUNDAMENTOS DE PROGRAMACION

Representación de un algoritmo :

- Un algoritmo se puede representar en 3 formas básicas:

FUNDAMENTOS DE PROGRAMACION

- D**iagramas de Flujo;**

  Es la representación gráfica de un algoritmo determinado a través de la utilización de símbolos gráficos que son interpretados como procesos, donde cada proceso tiene un diseño diferente y están ligados por líneas.

FUNDAMENTOS DE PROGRAMACION

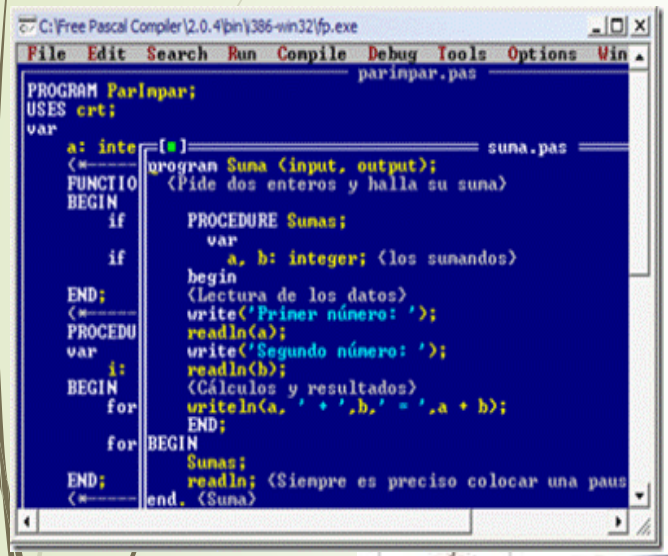
- **Pseudocódigo**

  Es la representación de un algoritmo en el lenguaje coloquial, es decir lenguaje nativo propio de cada lugar.

- **Código**

  Es la representación de un algoritmo escrito en un lenguaje de programación.

Ejemplos de lenguajes de Programación:



Programa

- "Software en inglés, es una secuencia de instrucciones que una computadora puede interpretar y ejecutar."

- Un **programa** informático o **programa** de computadora es una secuencia de instrucciones, escritas para realizar una tarea específica en una computadora.

Programa

- Según Niklaus Wirth un programa está formado por algoritmos y estructura de datos.

- Se han propuesto diversas técnicas de programación, cuyo objetivo es mejorar tanto el proceso de creación de software como su mantenimiento. Entre ellas se pueden mencionar las programaciones lineal, estructurada, modular y orientada a objetos.

Metodologías

- En la realización de un programa es necesario seguir una **metodología** encaminada al cumplimiento de ciertas características.

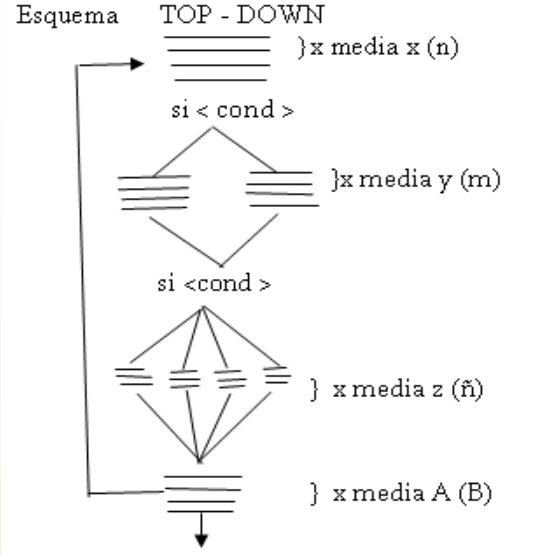
- "*La metodología de la programación es la técnica que permite que la programación sea lo mas eficaz posible en cuanto al desarrollo y al mantenimiento*".

Metodologías de la Programación

- Las técnicas de programación que permiten seguir una metodología de la programación más empleadas son:
  - Metodología Estructurada.
  - Metodología Modular
  - Metodología Orientada a Objetos

Metodología Estructurada

- La programación estructurada hace los programas más fáciles de escribir, verificar, leer y mantener, utiliza en su diseño los siguientes conceptos o principios fundamentales:
  - *Diseño descendente. (top-down).*
  - *Recursos abstractos.*
  - *Estructuras básicas.*



Metodología Estructurada

- En un programa estructurado el flujo lógico se gobierna por las estructuras de control básicas :
  - Estructuras Secuenciales
  - Estructuras Repetitivas.
  - Estructuras Selección o alternativas.

Metodología Estructurada

Estructuras Secuenciales:

- Operación de asignación

  var := exp ;

- Operación de Entrada/Lectura:

  READ(lista parámetros);
  READLN(lista parámetros);

- Operación de Salida/Escritura:

  WRITE(lista parámetros);
  WRITELN(lista parámetros);
  WRITELN;

Metodología Estructurada

- **Estructuras Selección o Alternativas**

```
IF condición THEN
BEGIN
   Sentencias en caso verdadero
END
[ELSE
BEGIN
   Sentencias en caso falso
END];
```

- Nótese que el ';' únicamente se pone al final de la sentencia IF-THEN-ELSE

- Si el grupo 'Sentencias en caso verdadero' (y/o 'Sentencias en caso falso') está formado por una única sentencia, el par BEGIN-END no es necesario

```
IF condición THEN
   S1
[ELSE
   S2];
```

- Sentencia CASE

```
CASE var OF
      Valor_1:S1;
      Valor_2:S2;
      ::::::::::::::::
      Valor_n:Sn;
END;
```

Metodología Estructurada

Estructuras Repetitivas:

- Bucle FOR

```
FOR var:=inicio TO/DOWNTO fin DO
BEGIN
    Cuerpo del bucle
END;
```

- Bucle WHILE

```
WHILE condición DO
BEGIN
    Cuerpo del bucle
END;
```

- Bucle REPEAT-UNTIL

```
REPEAT
    Cuerpo del bucle
UNTIL condición;
```

- Si 'cuerpo del bucle' (en la sentencia FOR y WHILE) está formado por una única sentencia, el par BEGIN-END no es necesario
- Nótese que la sentencia REPEAT-UNTIL no necesita un par BEGIN-END puesto que el cuerpo del bucle queda perfectamente delimitado con REPEAT al principio y UNTIL al final

Metodología Modular

- Nace del concepto de subprograma.
- Esta metodología técnicamente divide el problema en forma lógica en partes diferenciadas que puedan ser analizadas desarrolladas y puestas a punto en forma independiente.
- Los objetivos de esta metodología son los siguientes:

1.- Disminuir la complejidad y aumentar la claridad.
2.- Facilitar la ampliación de nuevos módulos.
3.- Facilitar la reestructuración de un proceso de forma transparente al usuario y al programador.
4.- Facilitar el mantenimiento

Metodología Modular

- La división de un problema en módulos y o programas independientes exige otro módulo que controle y relacione a todos los demás, a este se denomina **modulo base o principal** del problema.
- Un módulo en la aplicación del lenguaje Pascal puede ser una función o un procedimiento .

  - Una función es un proceso que retorna un único valor de tipo simple.
  - Un procedimiento es si este no retorna ningún valor pero tiene la característica de modificar o alterar los parámetros de entrada.

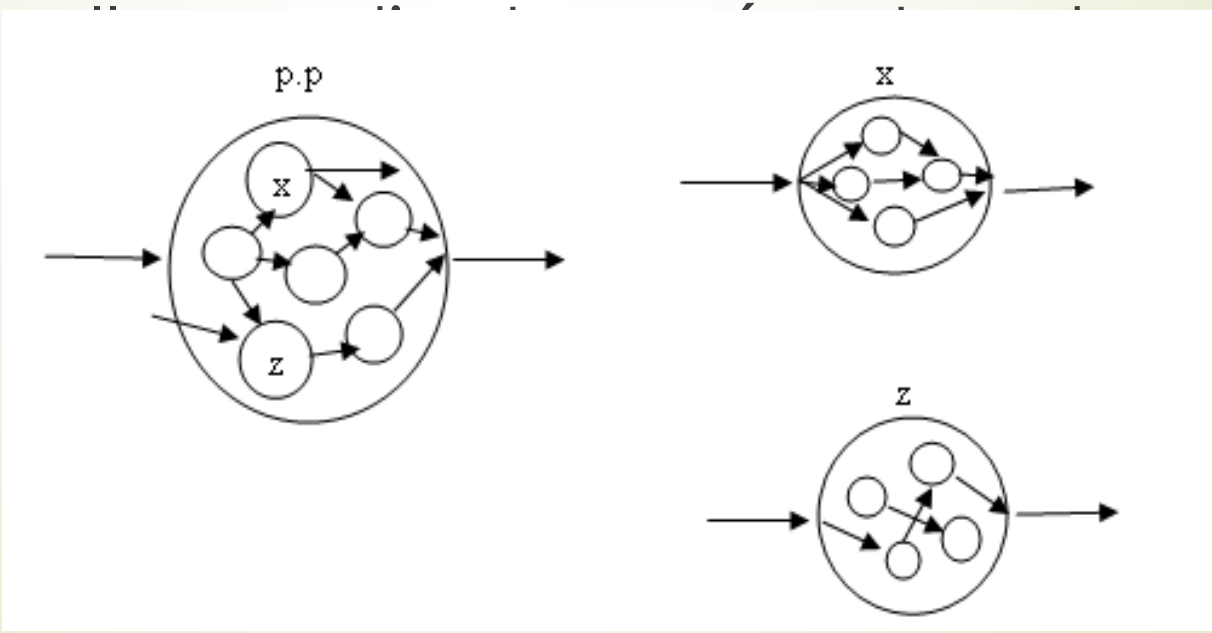
Metodología modular

```
FUNCTION ident_func (parámetros formales):tipo;
    {Declaración variables locales}
BEGIN
    Sentencias;
    ident_func:=expresión;
END;
```

```
PROCEDURE ident_proc (parámetros formales);
    {Declaración variables locales}
BEGIN
    Sentencias;
END;
```

Metodología Modular

- La modulación incluye descripción de los módulos y la relación entre



Metodología Modular

- Se pueden utilizar criterios de programación modular y posteriormente utilizar métodos de programación estructurada dentro de cada módulo independiente.
- Una vez terminado el diseño de los programas de cada módulo, se puede realizar el montaje del programa completo mediante un diseño descendente.
- Ejemplo:

```
program C19_func_operac;

{$APPTYPE CONSOLE}

uses
  SysUtils;


 function suma(a,b:real):real;
 begin
     result:= round(a+b);

 end;

 function resta(a,b:real):real;
 begin
     resta:= a-b;
 end;

 function prod(a,b:real):real;
 begin
     prod:= a*b;
 end;

 function divi(a,b:real):real;
 begin
   if (b <> 0)
    then
     divi:= a / b;
 end;
```

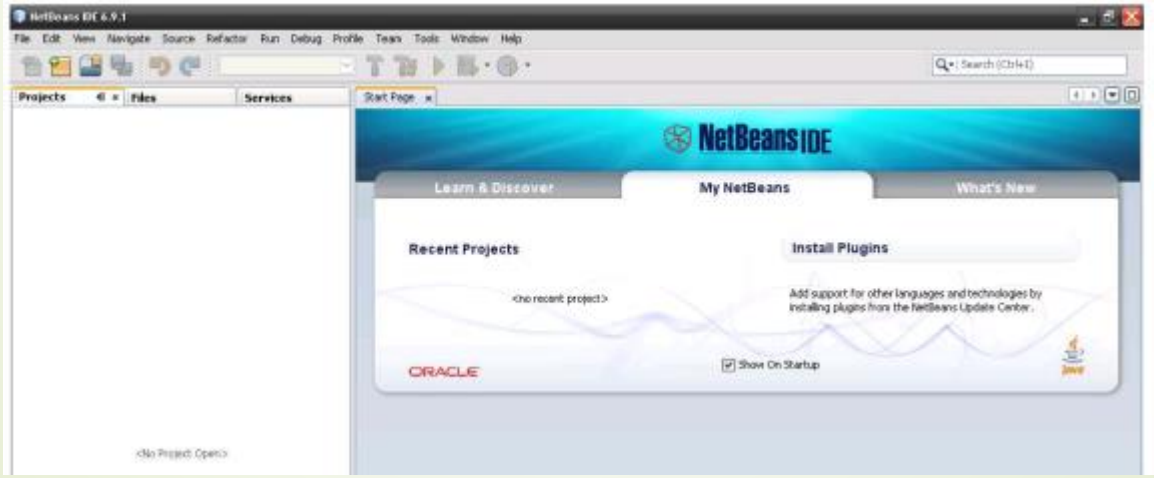
```
 var
  x,y, res:  real;
  rres: real;
  op:integer;

begin
   { TODO -oUser -cConsole Main : Insert code here }
    { funciones}
 repeat
  writeln('========================');
  writeln('      Funciones  ');
  writeln('========================');

  write('x  = '); readln(x);
  write('y  = '); readln(y);
  writeln('desea 1) sumar, 2) restar, 3 prod, 4) division,  5)salir');
  readln(op);
  case op of
        1: res:= suma(x,y);
        2: res:= resta(x,y);
        3: res:= prod(x,y);
        4: rres:= divi(x,y);
        5: exit;
  end;
  writeln;

  writeln('resultado = ', res);
 until(op = 5);
 readln;

end.
```

Metodología Orientada a Objetos

- E l análisis y desarrollo de programas se enfoca hacia las **clases y objetos** significan que en una aplicación se deben identificar los objetos involucrados, los rasgos que tiene cada objeto (necesarios para la aplicación) y las operaciones para cada clase u objeto particular.

NETBEANS

- Netbeans es un **entorno de desarrollo integrado** libre(IDE) que permite editar programas en java, compilarlos, ejecutarlos, depurarlos, construir rápidamente el interfaz gráfico de una aplicación eligiendo los componentes de una paleta, etc.



CLASE

- Las clases ***son abstracciones*** que representan a un conjunto de objetos con un:
  - **Comportamiento e**
  - **Interfaz común**
- Es la ***implementación*** de ***un tipo de dato*** (considerando los objetos como instancias de las clases).
- " **CLASE:** Conjunto que representa a los objetos de características similares"

CLASE

- Llacua – Vasquez

"Es una entidad que tiene Atributos(datos) y Procesos(métodos) inherentes al propósito y funcionamiento de la misma. Los atributos identifican las características de la clase y los procesos establecen su comportamiento"

CLASE

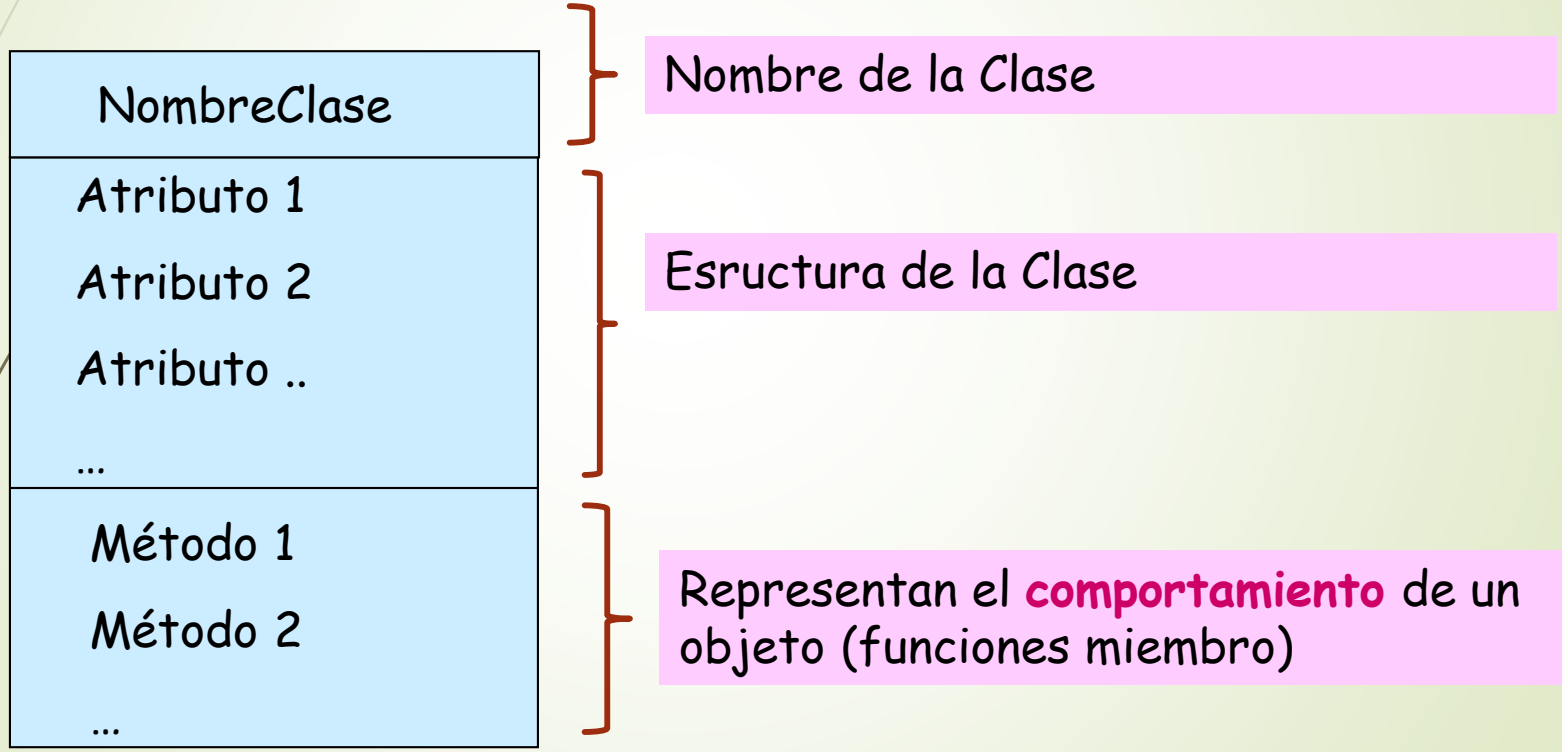
- Luis Joyanes Aguilar:

"Una Clase es un tipo definido por el usuario(programador) que determina las estructuras de datos y las operaciones asociadas con este tipo"

"La Clase es simplemente un modelo que se utiliza para describir uno o más objetos del mismo tipo"

- **Como identificar a los clases por el nombre?**
  - Comienza con mayúsculas.
  - Si el nombre se forma de varias palabras no deje espacios entre ellas, a partir de la segunda palabra, comienza con mayúscula.

- Los elementos que componen la clase son:



Objeto

- Un objeto es una abstracción de una entidad del mundo real.
- Toda interacción con el mundo externo es por medio de una interfase.
- Un objeto es **una instancia de una clase** o una variable del tipo de dato definido por la clase.

- **Como identificar a los objetos por el nombre?**
  - Comienza con minúsculas.
  - Si el nombre se forma de varias palabras no deje espacios entre ellas, a partir de la segunda palabra, comienza con mayúscula.

Ejemplo 1 – Clase y Objetos

Ejemplo 2 – Clase y Objetos

ATRIBUTOS

- Concepto: Características que definen a la clase, es lo que el usuario (de la clase) puede ver. Es la abstracción.
- Otro concepto: variables que están dentro de los objetos y que contienen los valores del objeto.
- También son llamados: propiedades, datos miembro.

EJEMPLO DE OBJETOS

- Cada Objeto tiene sus propias características:

MARIA

- Reg: 23453
- Apellidos: Rosas Perez
- Nombres: Maria
- Sexo: F
- Direccion:Calle Sucre
- Cel: 678445

LUIS

- Reg: 103453
- Apellidos: Rios Sosa
- Nombres: Luis
- Sexo: M
- Direccion:Av bolivar
- Cel: 7778445

ANA

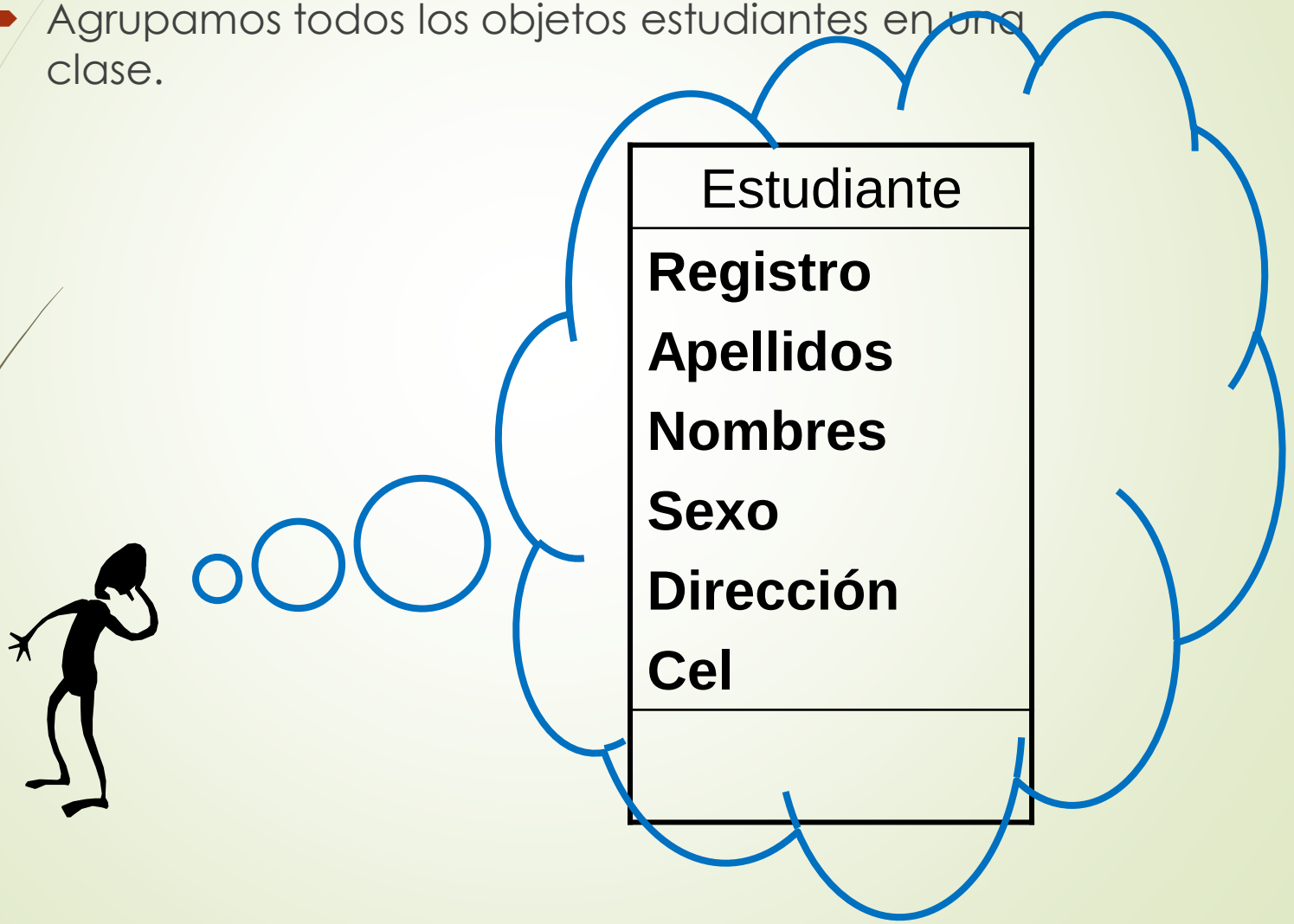
- Reg: 1003453
- Apellidos: Perez Marquez
- Nombres: Ana Luisa
- Sexo: F
- Direccion:Calle Oruro
- Cel: 789445

CARLOS

- Reg: 563453
- Apellidos: Cardozo
- Nombres: Carlos Juan
- Sexo: M
- Direccion:Calle Sucre
- Cel: 5678445

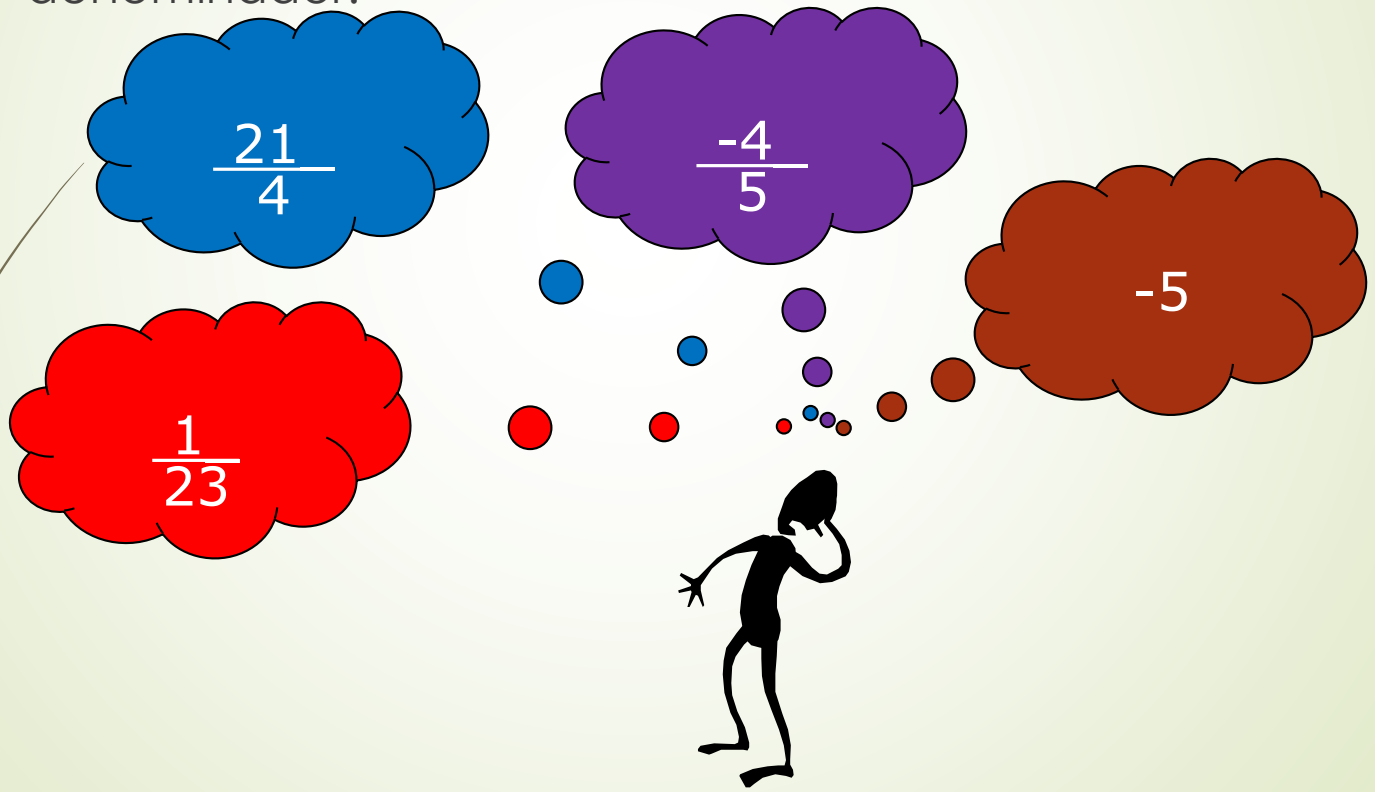
EJEMPLO DE CLASE

- Agrupamos todos los objetos estudiantes en una clase.



ATRIBUTOS – Ejemplo1

- Una Fracción está definido por un par de números naturales, los cuales se denominan el numerador y denominador.



ATRIBUTOS – Ejemplo1

- Una Fracción está definido por un par de números naturales, los cuales se denominan el numerador y denominador.

Fraccion
Numerador
Denominador

ATRIBUTOS – Ejemplo2

- Un Fecha está definido por tres datos: día, mes y año.

ATRIBUTOS – Ejemplo2

- Un Fecha está definido por tres datos: día, mes y año.

Fecha
Día **Mes** **Año**

MÉTODOS

- Concepto: Son las operaciones que se pueden realizar con los objetos de la clase.
- Otro concepto: Son los procesos (funciones o procedimientos) que permiten trabajar con los datos de los objetos
- También son llamados: funciones miembro
- De acuerdo al trabajo que realizan los métodos, estos se clasifican en:Constructores,destructores, ponedores, selectores.

COMPORTAMIENTO

- El comportamiento de un objeto es la forma en que actúa y reacciona, en términos de sus cambios de estado e intercambio de mensajes con otros objetos.
- El comportamiento de un objeto está determinado por las operaciones que pueden ser invocadas sobre el mismo.
- Los métodos representan el comportamiento de la Clase

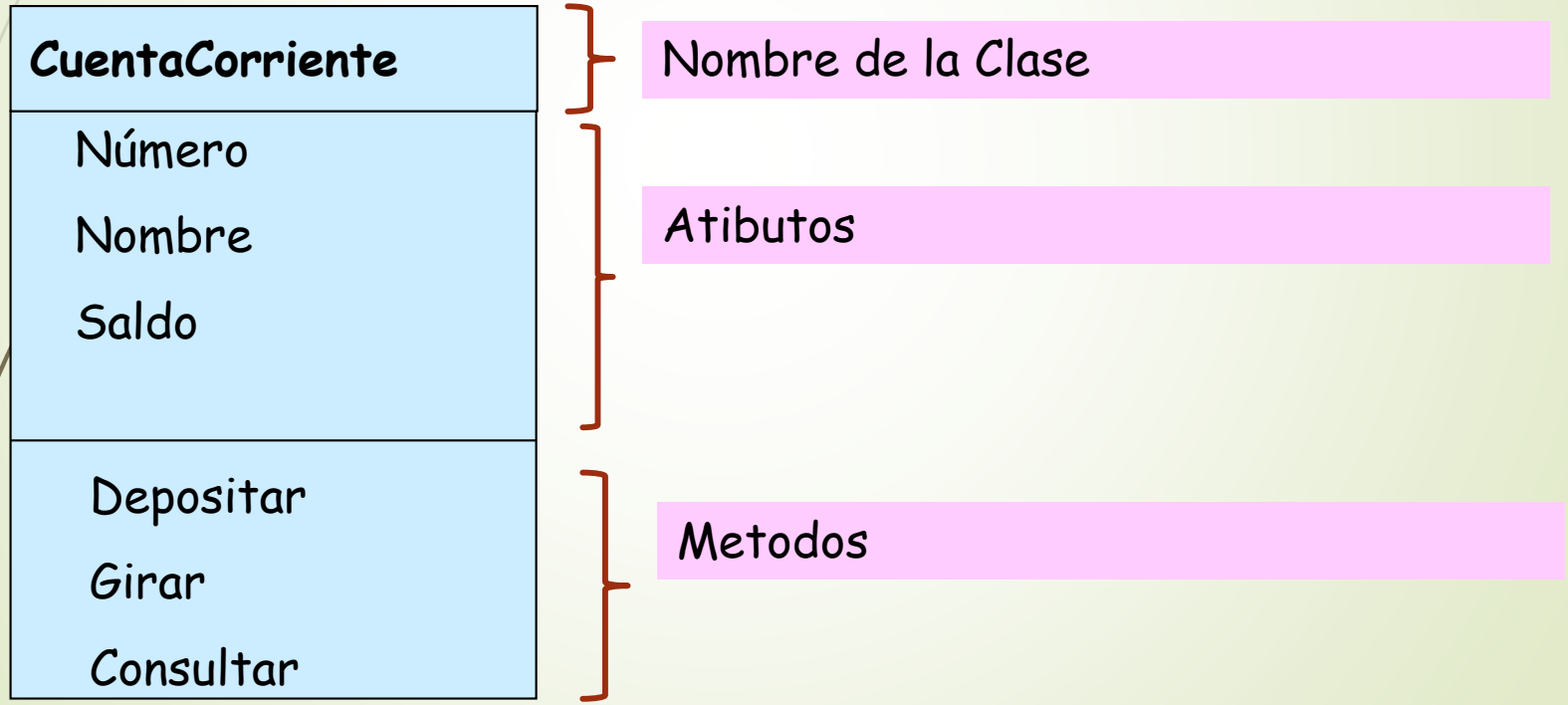
Métodos – Ejemplo 3 - 4

Representación gráfica de los métodos de las clases CQuebrado y CVector

CQuebrado
PonNumerador PonDenominador ObtDenominador ObtNumerador Simplificar

CVector
PonDimension PonElemento ObtDimension ObtElemento Ordenar

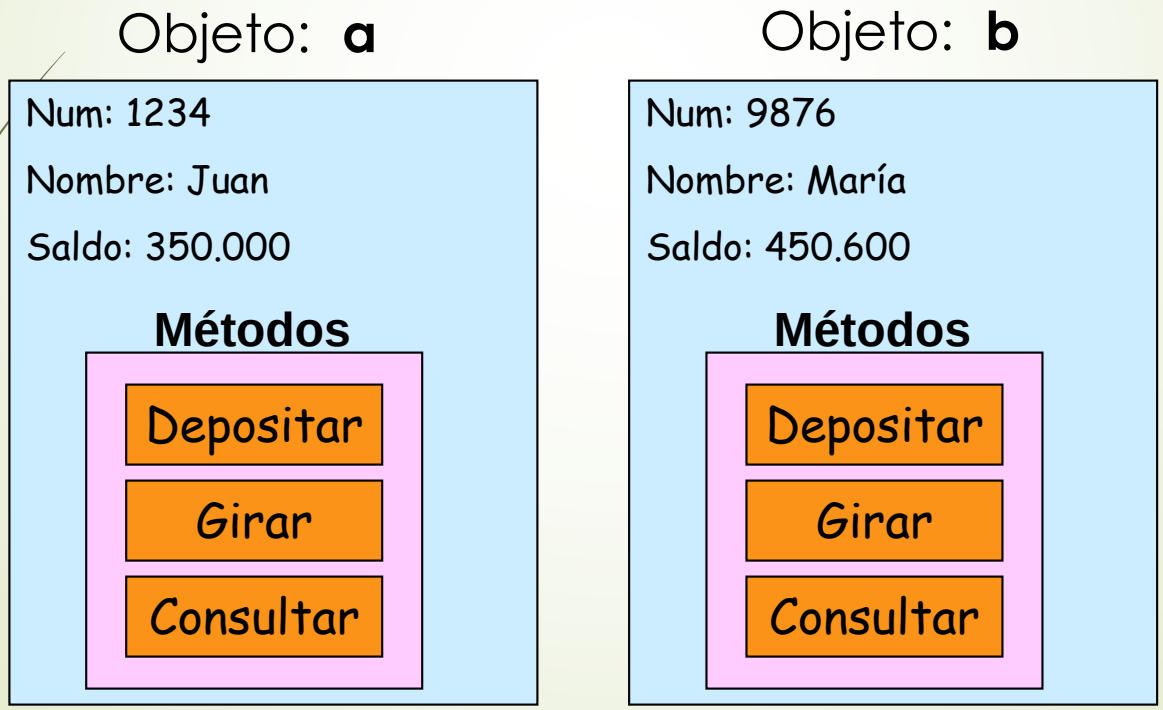
Ejemplo de Clase



Ejemplo de instancia de objetos

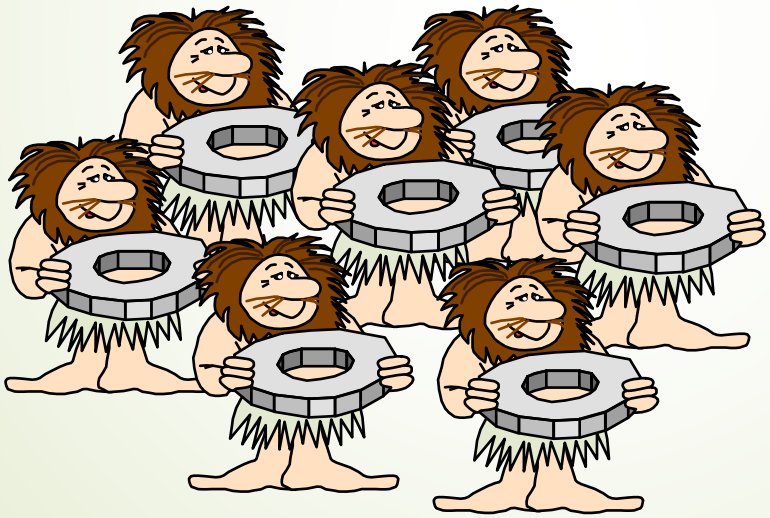
- **Clase:** Cuenta corriente

- **Instanciación: Cuenta Corriente** a, b



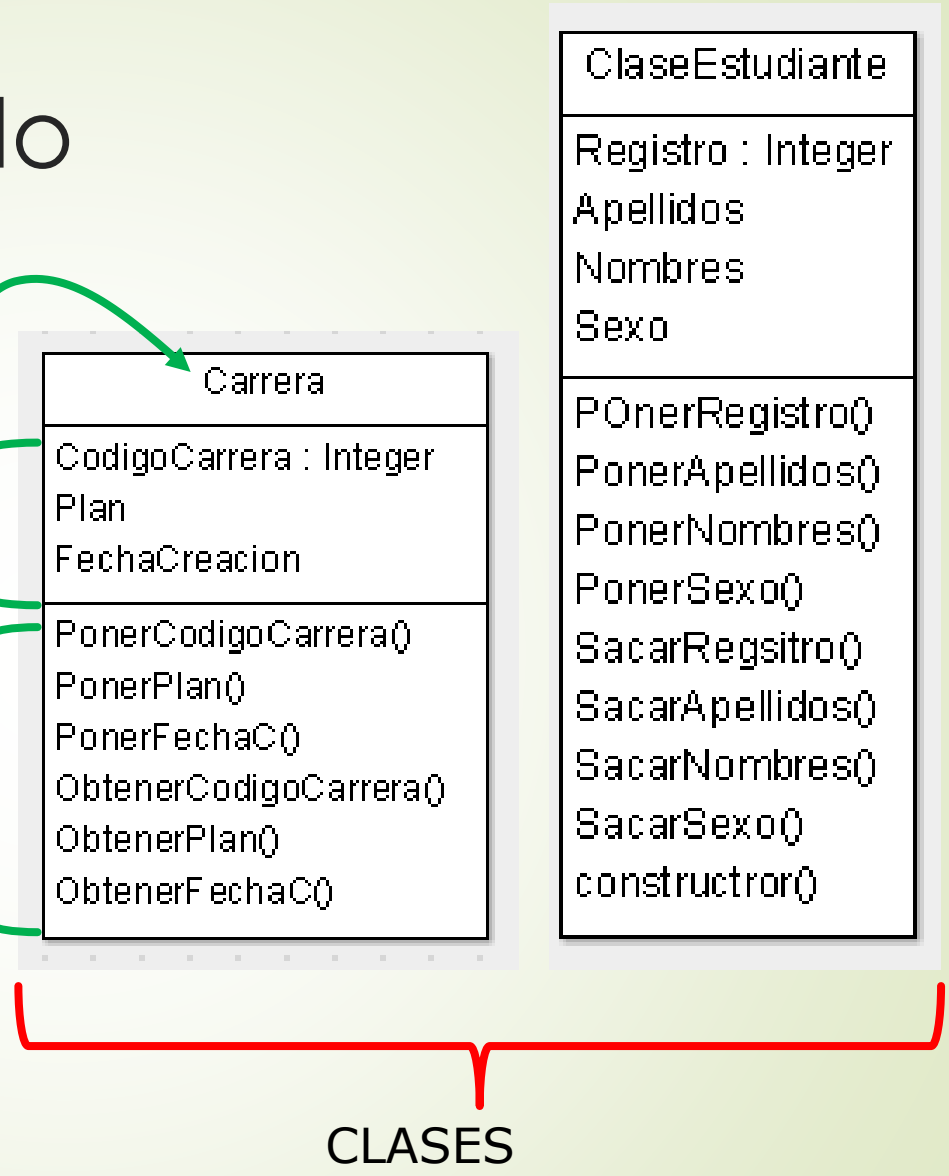
Parte 2

Repasando:



Repasando

- Una clase es un tipo de dato definido por el usuario, que posee un
  - ✓ nombre
  - ✓ estructura y un
  - ✓ comportamiento.
- Una clase describe las características genéricas y el comportamiento de algunos objetos similares.



Repasando

- Un objeto es una instancia de una clase o una variable del tipo de dato definido por la clase.
- Los objetos son entidades reales.
- Cuando el programa se ejecuta, los objetos ocupan parte de la memoria para su representación interna. La relación entre el objeto y la clase es la misma que entre una variable y un tipo.

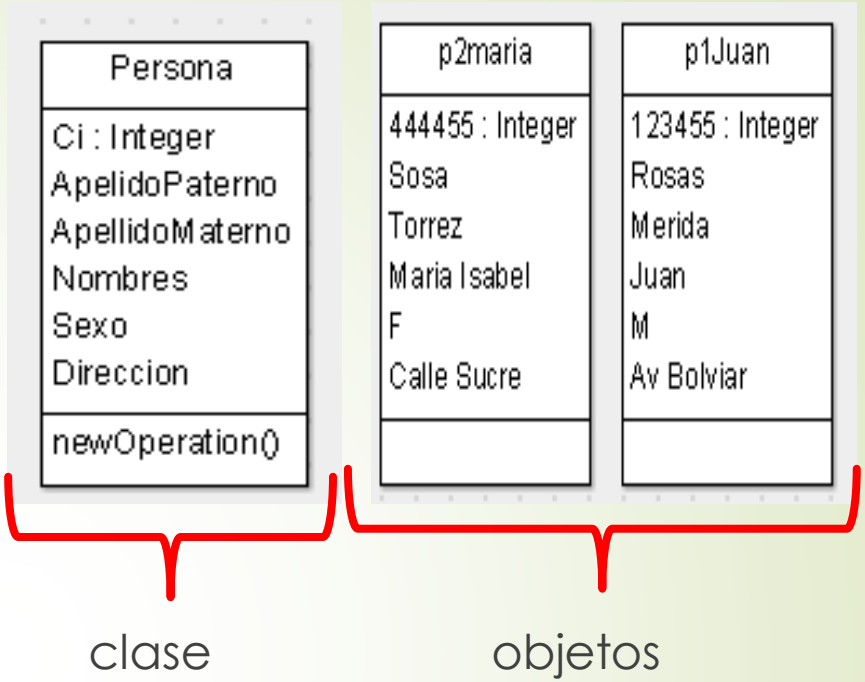
Objeto y clase

Persona p1Juan, p2Maria;

Clase Objetos

son instancias

de la clase Persona



Variable y tipo

int x;

Tipo Variable

- **Variable y tipo**

  x = 0; INICIALIZACION DE VARIABLE

- **Objeto y clase**

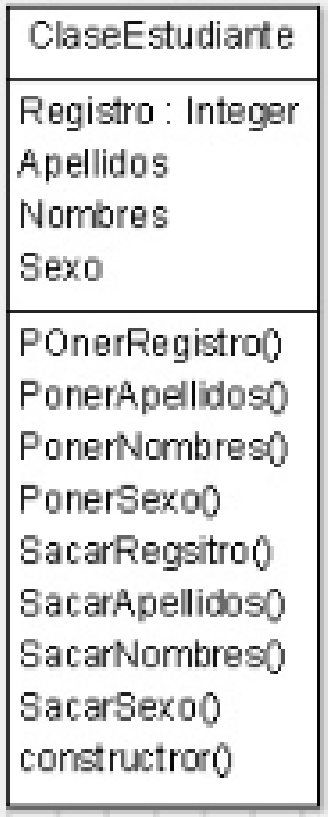
  p1Juan= new Persona ( );

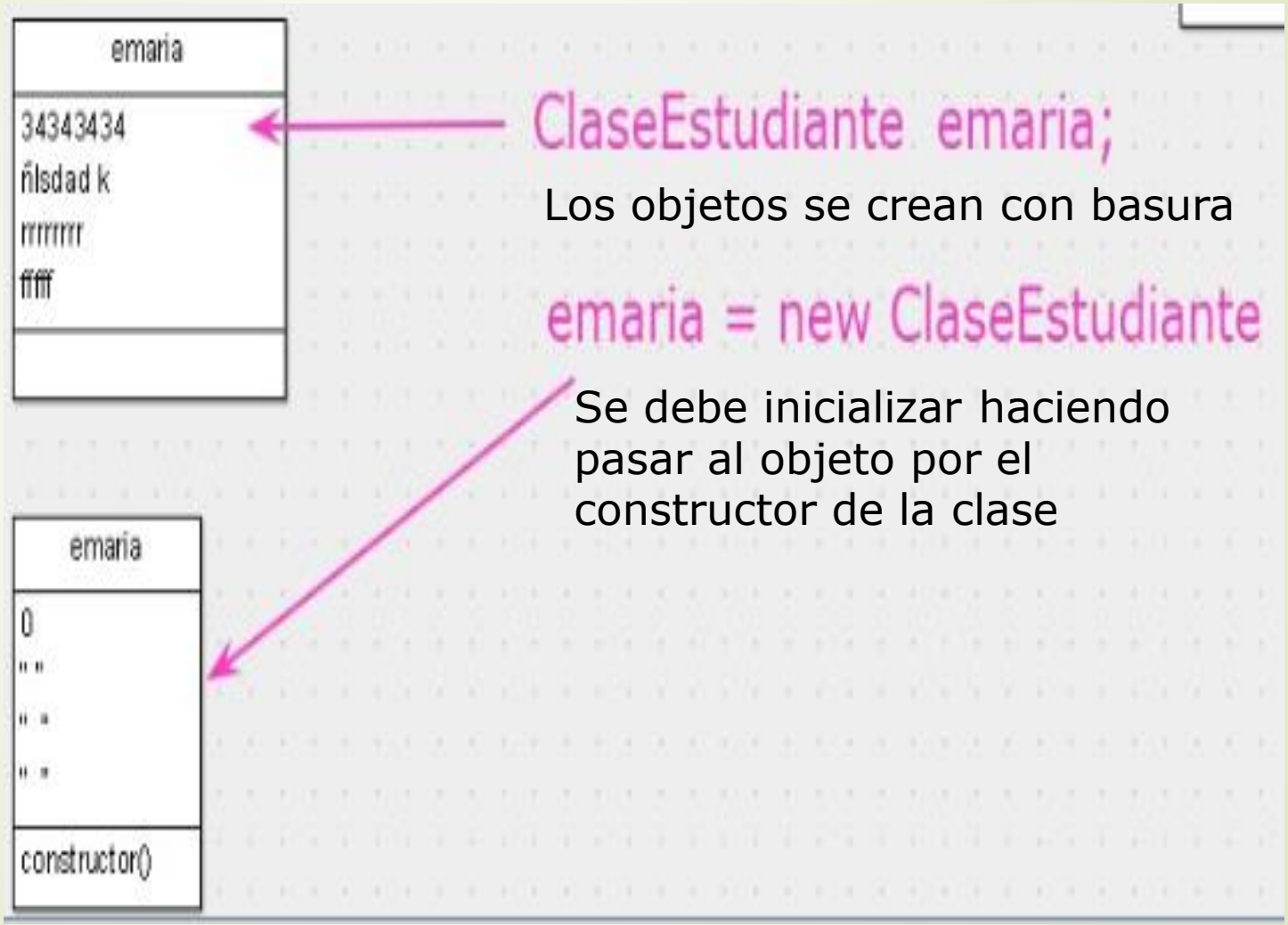
  p2Maria= new Persona ( );

  HACIENDO PASAR POR EL CONSTRUCTOR

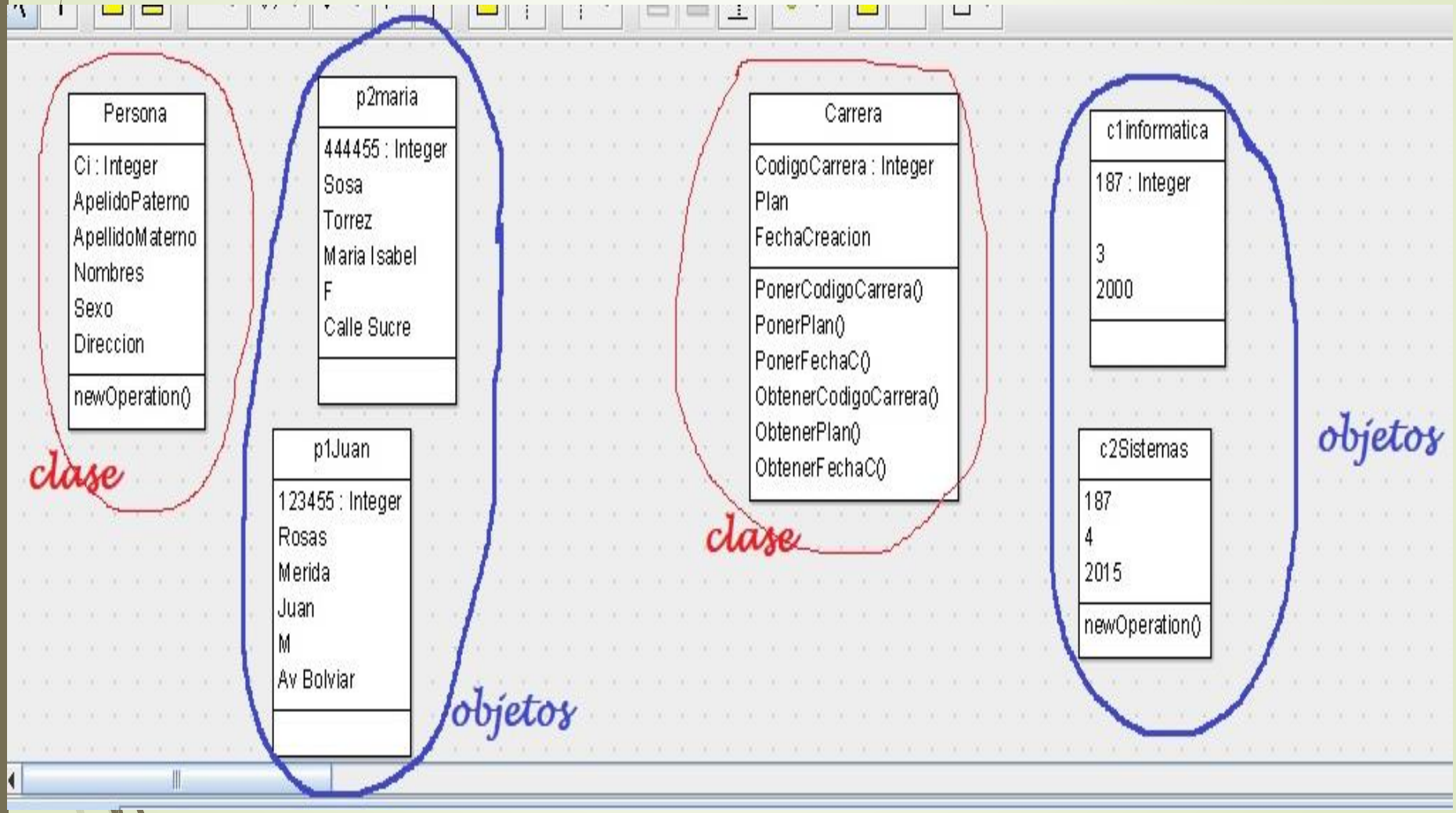
OBJETOS

CLASE

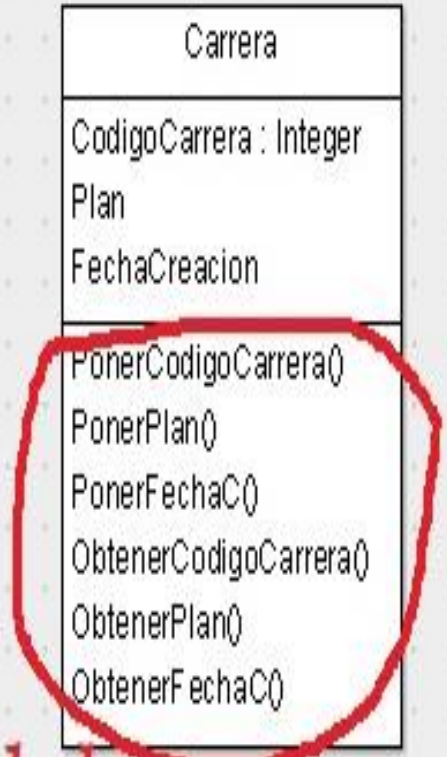




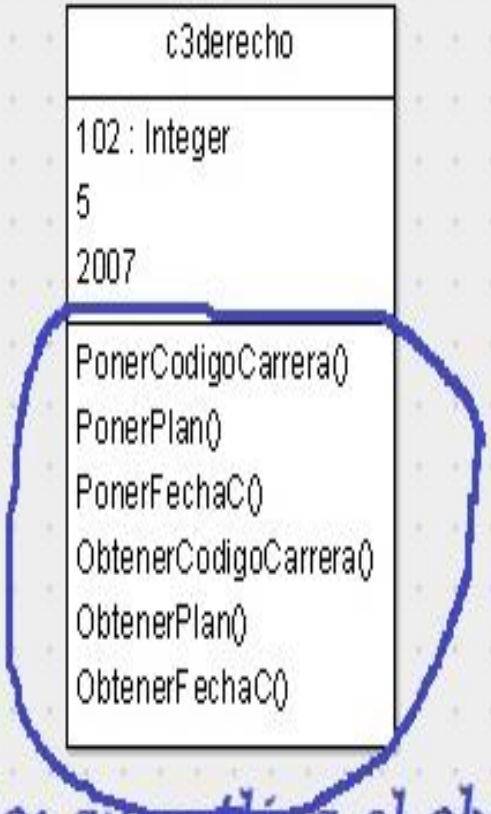
Mas ejemplos de clases y objetos



Métodos de la clase



métodos de la clase carrera



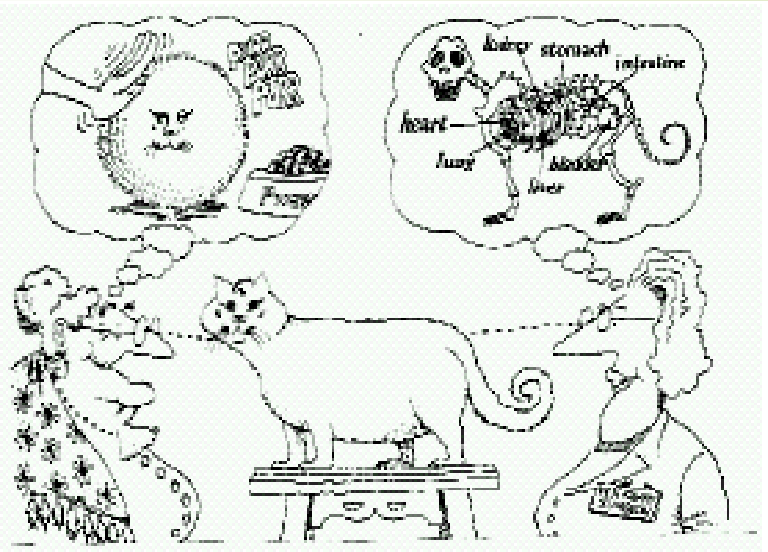
métodos que utiliza el objeto c3derecho y son los mismos que se definieron en la clase, la unica forma de manipular los datos del objeto es a traves de sus métodos

Características de la programación Orientada a Objetos

- Abstracción.
- Encapsulamiento
- Herencia
- Modularidad
- Polimorfismo.

Abstracción

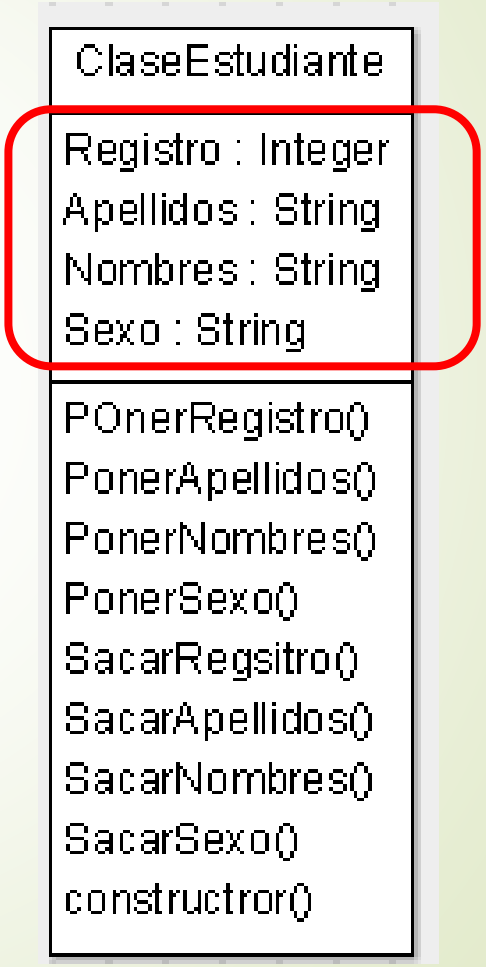
- **Concepto.** Se define como la capacidad para examinar algo sin preocuparse de sus detalles internos.



• Existen dos tipos de abstracciones: abstracción de **datos** y abstracción **funcional**

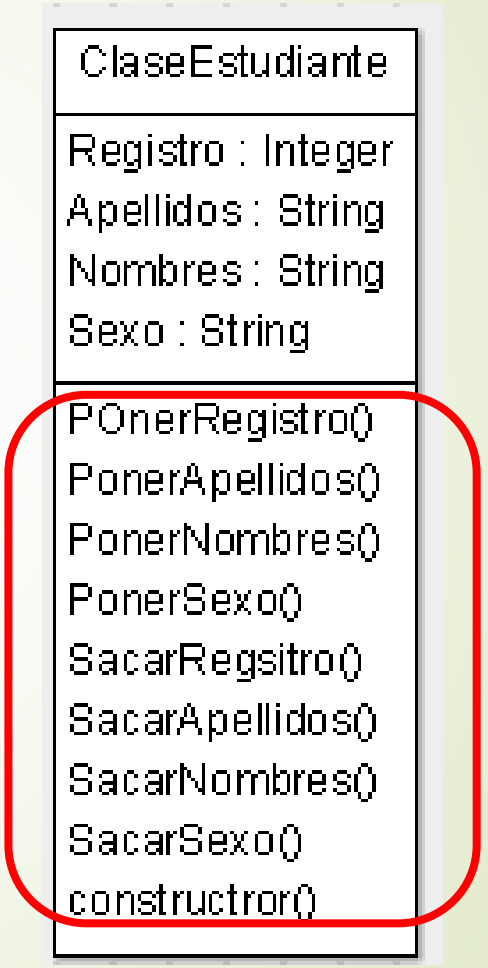
Abstracción de **DATOS**

- Consiste en utilizar los datos sin preocuparse por los detalles de su implementación, es decir, lo importante es **saber que tipo de información** se puede utilizar y **no como está almacenado**.



Abstracción **FUNCIONAL**

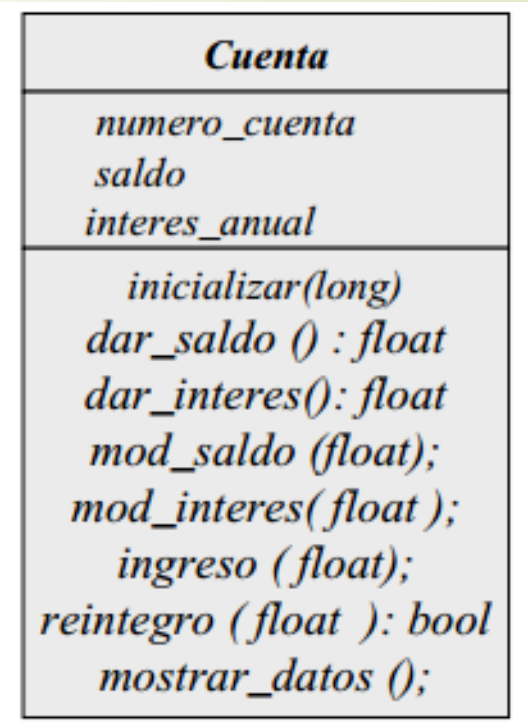
- Consiste en saber **que es lo que hace** un determinado proceso, pero **no como lo hace**.



Encapsulamiento

- Consiste en reunir varias cosas para ser manipuladas como una sola unidad.
- Es un Proceso por el que se ocultan:
  - Las estructuras de datos
  - Los detalles de la implementación

- Al definir una clase estamos encapsulando los atributos y los métodos.
- Una vez creada la clase, las funciones usuarias no requieren conocer los detalles de su implementación.
- Los métodos y atributos pueden ser utilizados por ***otras*** clases ***sólo si*** la clase que los encapsula les brinda los ***permisos*** necesarios para ello



Abstracción y Encapsulamiento

- Estos dos conceptos se complementan uno con otro, **abstracción** pone mayor énfasis en el que hace cierta clase y no como internamente lo hace, y **encapsulamiento** en ocultarnos los miembros internos de una clase.

Ocultamiento

- Consiste en **NO permitir** el acceso a los componentes de una clases (atributos, métodos).
- Esto se logra con la visibilidad que ofrecen los lenguajes de programación.
  - **Privado:** Solo los miembros de la clase tienen acceso
  - **Protegido:** Solo los miembros de la clase y sus derivados tienen acceso
  - **Público:** Todos tienen acceso

Modularidad

- Este concepto ya viene desde la programación modular, y se refiere al hecho de realizar un programa por partes, a las cuales se las denomina módulos.

- Un módulo es un archivo que contiene un conjunto de declaraciones y/o procesos. En OOP normalmente un módulo contiene la interfaz y la implementación de un o más clases relacionadas.

Polimorfismo

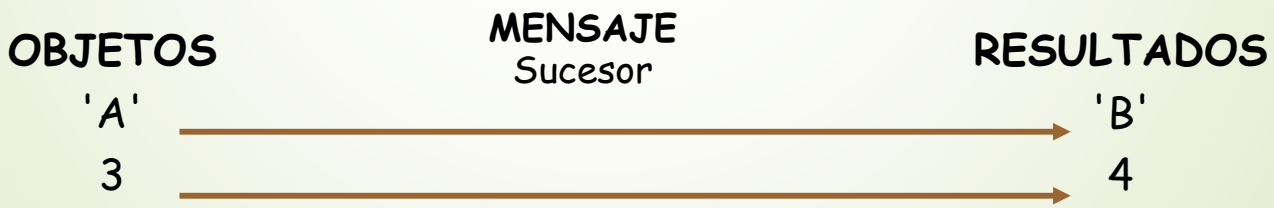
- Capacidad que permite a dos ***clases diferentes*** responder de forma distinta a un ***mismo mensaje***

- Esto significa que dos clases que tengan un método con el mismo nombre y que respondan al mismo tipo de mensaje (es decir, que reciban los mismo parámetros), ejecutarán acciones distintas

Polimorfismo

Ejemplo :

Si se tienen las clases **Entero** y **Char**, ambas responderán de manera distinta al mensaje **"Sucesor"**



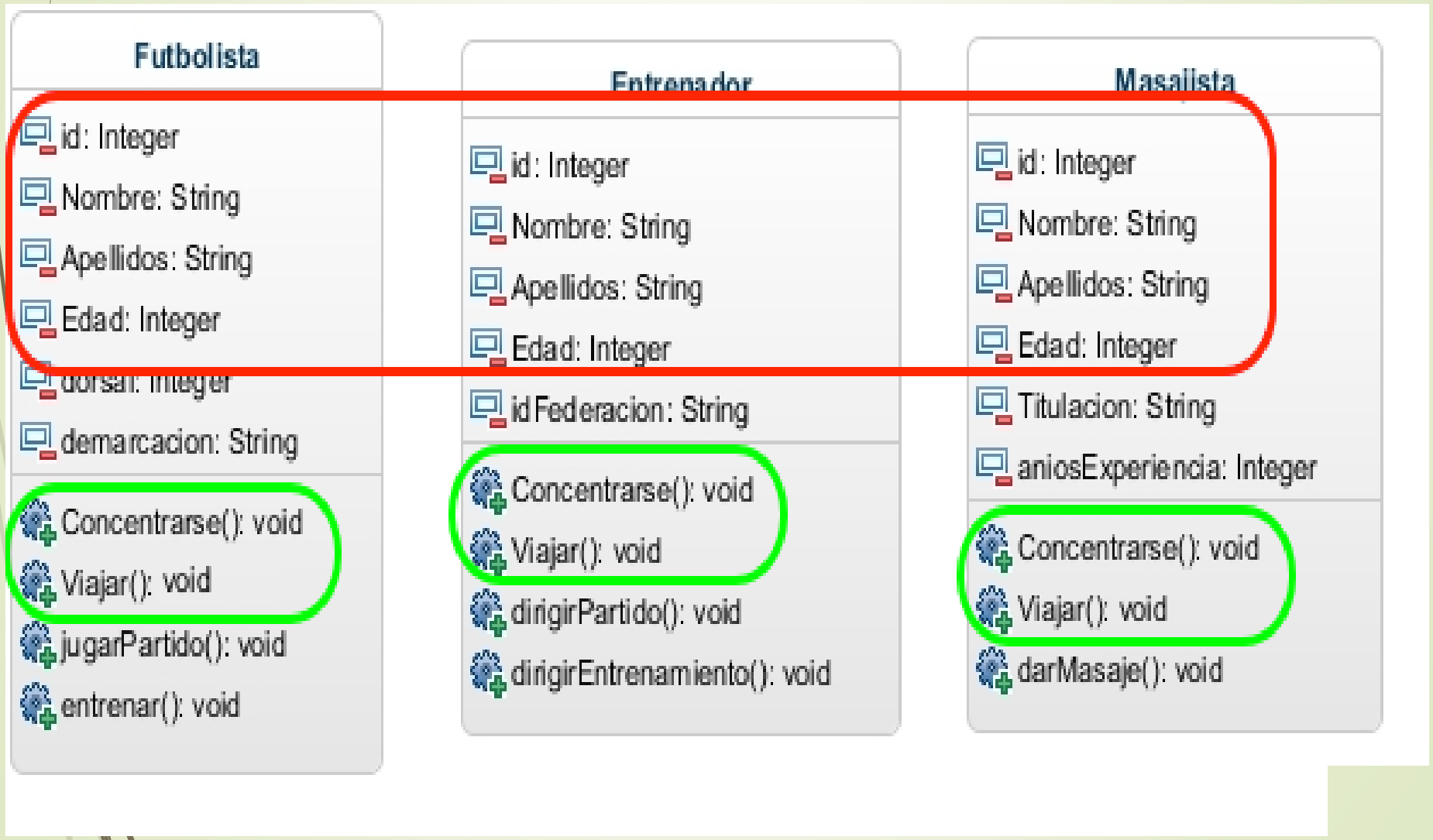
Herencia

- Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen.

- La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo.

- ***La Herencia*** es uno de los 4 pilares de la programación orientada a objetos (POO) junto con la ***Abstracción***, ***Encapsulación*** y ***Polimorfismo***.

Ejemplo


Futbolista
id: Integer
Nombre: String
Apellidos: String
Edad: Integer
dorsal: Integer
demarcacion: String
Concentrarse(): void
Viajar(): void
jugarPartido(): void
entrenar(): void
Entrenador
id: Integer
Nombre: String
Apellidos: String
Edad: Integer
idFederacion: String
Concentrarse(): void
Viajar(): void
dirigirPartido(): void
dirigirEntrenamiento(): void
Masajista
id: Integer
Nombre: String
Apellidos: String
Edad: Integer
Titulacion: String
aniosExperiencia: Integer
Concentrarse(): void
Viajar(): void
darMasaje(): void

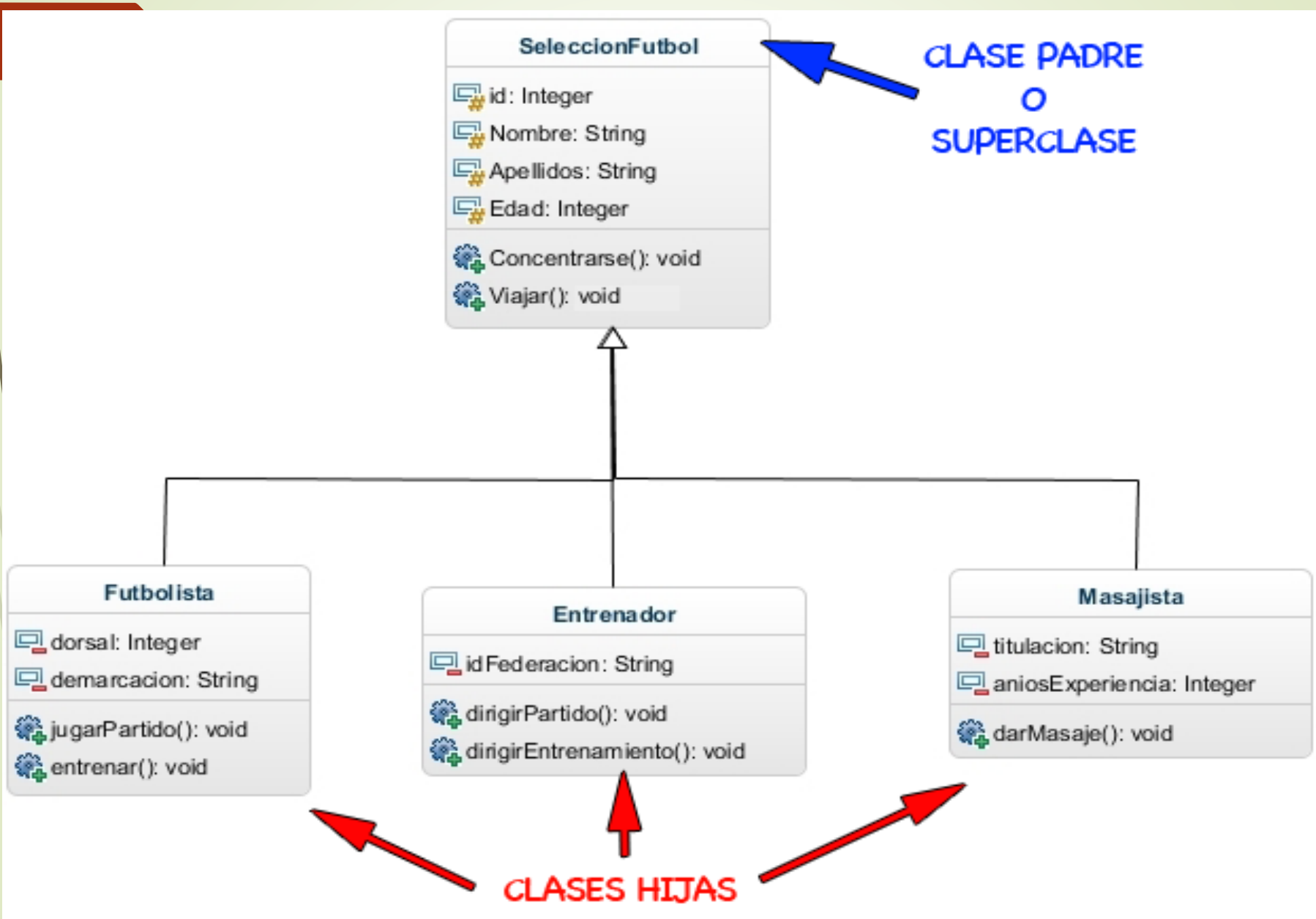
A nivel Código

```
public class Futbolista
{

        private int id;
        private String Nombre;
        private String Apellidos;
        private int Edad;
        private int dorsal;
        private String demarcacion;

        // constructor, getter y se
tter

        public void Concentrarse()
{
                ...
        }

        public void Viajar() {
                ...
        }

        public void jugarPartido()
{
                ...
        }

        public void entrenar() {
                ...
        }
}
```

```
public class Entrenador
{

        private int id;
        private String Nombre;
        private String Apellidos;
        private int Edad;
        private String
idFederacion;

        // constructor, getter y se
tter

        public void Concentrarse()
{
                ...
        }

        public void Viajar() {
                ...
        }

        public void
dirigirPartido() {
                ...
        }

        public void
dirigirEntreno() {
                ...
        }
```

```
public class Masajista
{

        private int id;
        private String Nombre;
        private String Apellidos;
        private int Edad;
        private String Titulacion;
        private int aniosExperiencia;

        // constructor, getter y sett
er

        public void Concentrarse() {
                ...
        }

        public void Viajar() {
                ...
        }

        public void darMasaje() {
                ...
        }
}
```

Solución - Herencia



En código

```
public class SeleccionFut
bol
{

        protected int id;
        protected String
Nombre;
        protected String
Apellidos;
        protected int Eda
d;

        // constructor, g
etter y setter

        public void Conce
ntrarse() {
                ...
        }

        public void Viaja
r() {
                ...
        }
}
```

```
public class Futbolista e
xtends SeleccionFutbol
{
        private int dorsa
l;
        private String de
marcacion;

        public
Futbolista() {
                super();
        }

        // getter y sette
r

        public void jugar
Partido() {
                ...
        }

        public void entre
nar() {
                ...
        }
}
```

```
public class Entrenador e
xtends SeleccionFutbol
{

        private String id
Federacion;

        public
Entrenador() {
                super();
        }

        // getter y sette
r

        public void dirig
irPartido() {
                ...
        }

        public void dirig
irEntreno() {
                ...
        }
}
```

```
public class Masajista ext
ends SeleccionFutbol
{

        private String Tit
ulacion;
        private int aniosE
xperiencia;

        public Masajista()
 {
                super();
        }

        // getter y setter

        public void darMas
aje() {
                ...
        }
}
```

- GRACIAS…