Tema 7: Programación Orientada a Objetos

- Definición
- Clasificación
- Encapsulación
- Clases
- Nomenclatura
- Constructores
- Sobrecarga

Definición

- La programación orientada a objetos (POO) permite cambiar el paradigma de progamación, prestando atención en el objeto y sus relaciones con el resto.
- Cambia la filosofía respecto a ala programación tradicional en la que el foco de desarrollo era la función.
- Ejemple POO – Tradicional (imprimir tabla)

Conceptos POO

- Clase: Métodos y atributos de un mismo comportamiento.
- Objeto: Instancia de una clase
- Subclase: Clase que deriva de otra clase existente. La clase derivada hereda todos sus métodos y atributos.
- Herencia: Capacidad de recibir métodos y atributos de las clases superiores.

Conceptos POO – II

- Encapsulación: Ocultar datos y métodos, para hacer más sencilla la utilización. Permitiendo el acceso a la parte pública.
- Polimorfismo: Capacidad de un objeto de asumir diferentes formas según sea el objeto existente.

Clasificación

- Organización de elementos con significado común.
- Produciendo una relación directa con los objetos con los que nos rodean. En P.O.O. permite definir clases que contienen:
  - Comportamiento: métodos (funciones)
  - Atributos: datos (variables)

Encapsulación

- Organizar diferentes elementos y dotarles de la misma estructura.
- Se emplea para:
  - Combinar métodos y datos en una clase → Definición (class)
  - Controlar el acceso los métodos y datos en una clase → Control de acceso (public, private)

Definición clases

```
tipoAcceso class nombre
{
    Métodos → Métodos.
    Variables→Campos
}

 public class Circulo
  {
     double radio;

     double Area()
     {
       return 3.1416 * radio * radio;
     }
  }
```


Control de acceso

- tipoAcceso
  - public: Accesible por exterior e interior de la clase.
  - private: Accesible por interior de la clase.
  - protected: Accesible por interior de la clase y los hijos.
    - Omisión: private

Contructores

- Clase: Clasificación de elementos que comparten métodos y atributos. Ej: Todas las personas tienen edad, nombre y pueden escribir su nombre.
- Objeto: Instanciación de un elemento de la clase. Ej: Puede haber 20 objetos de la clase persona con edades y nombres diferentes. Todos podrán escribir su nombre.

Constructores

- Para crear una variable hay que llamar al constructor. Según tipo de datos:
  - Primitivos: int,long,float, doucle, char, String.
  - Datos de clase (objetos): llamar a un método constructor.

```
<NombreClase>  <identidicador>;
<identidicador> = new Constructor;
```


Constructor

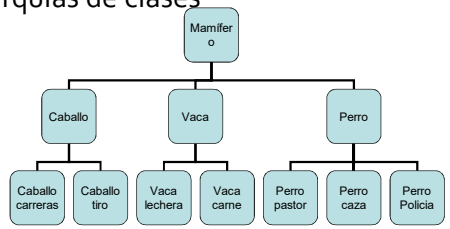
- Constructor por defecto:
  - Numeros a 0.
  - Ref a null.
  - Boolean a false.
- Toda clase tiene un constructor por defecto sin parámetros.

Sobrecarga

- Defínción de varios métodos que tengan el mismo identificador pero se diferencian por distintos parámetros: por número o por tipo diferente.
- Cuando se realza la llamada se determina el método según los parámetros.
- Cuando un mismo método tiene varias versiones con distintos parámetros se dice que está ***sobrecargado.***

Herencia

- Clasificación de clases con relaciones de tipo padre hija.
- Mediante la herencia se consiguen establecer jerarquías de clases



Herencia

- public class <claseHija> extends <clasePadre>
- La clase Hija **hereda** de la clase Padre, sólo se puede heredar de una única clase.
- Cuando una clase hija hereda de una clase padre, puede acceder a:
  - Sí puede acceder a atributos y métodos public.
  - No puede acceder a atributos y métodos private.
  - Sí puede acceder a atributos y métodos protected.

Herencia

```
Class Mamífero
{
   Int extremidades;
   String nombreCientifico;
   String nombreComun;
}
Class Caballo
{
   String colorCrin;
   String tipoBocado;

}
Class Vaca
{
   String tipoPelaje;
}
Class CaballoCarreras
{
   Int añosCorriendo;
   Int campeonatosGanado;
   Int velocidadMaxima;
}
Class CaballoTiro
{
   Int cargaMaxima;
   Int cargaHabitual;
}
cl
```