1. Clases
2. Objetos y métodos

Desarrollo web con PHP

Orientación a objetos

Técnica de programación que define clase y usa objetos y sus interacciones para diseñar aplicaciones de cómputo.

Orientación a objetos en PHP

- **Empezó en php4**
- **Tienen muchas maneras extravagantes de iniciar un objeto. Veremos las más apegadas a los otros lenguajes con paradigma orientados a objetos.**

Declaración de clases

En PHP se usa la palabra reservada class seguido del nombre de la clase. El nombre puede estar formado con letras y números, pero no puede empezar con número. Ejemplo:

```
class Persona {
        //cuerpo de la clase
}
```

Declaración de clases: atributos o propiedades

Sintáxis :

$<nombre Atributo> [= <valor por omisión>];

Ejemplo:

```
class Persona {
        public $nombre = “Pedro”;
        public $apellido_paterno = “Juárez”;
        public $apellido_materno = “Romero”;
}
```

Declaración de clases: métodos

Se le llama ***firma o encabezado*** de un método al nombre del método, al número y tipo de los parámetros.

```
public function Nombre($par1, …,$parN){
     // Código del método
}
```

En PHP sólo se puede tener un método con un mismo nombre, no importa si la firma es diferente.

Declaración de clases: métodos

```
class Persona {
    public $nombre = "Pedro";
    public $apellidoPat = "Juárez";
    public $apellidoMat = "Romero";

    public $correo;


    public function saludaA($nombre) {
       return "Hola ".$nombre."!";
    }

}
```

Modificadores de Visibilidad

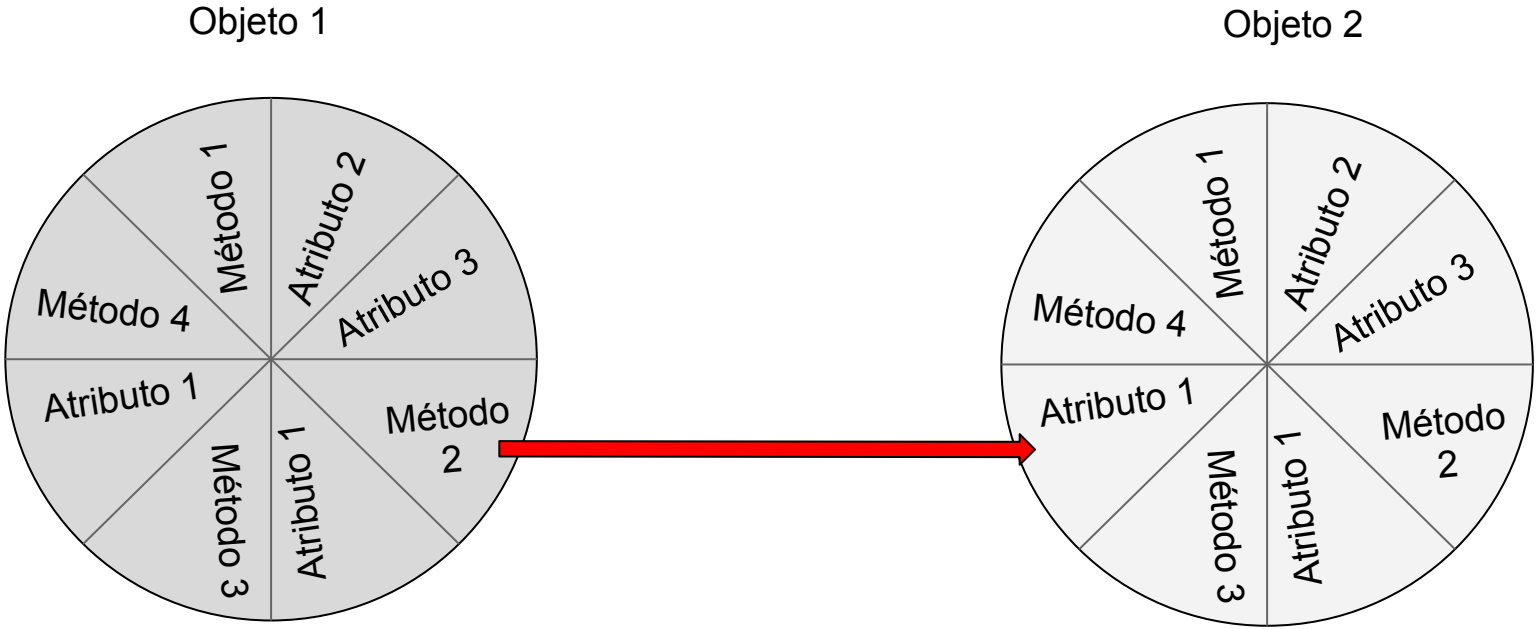
Son usados para determinar la visibilidad tanto de atributos como métodos. Es decir qué objetos, según la clase a la que pertenecen, pueden hacer uso de ellos (atributos y métodos). Son tres:

1. ***public***
2. ***protected***
3. ***private***

Modificadores de Visibilidad

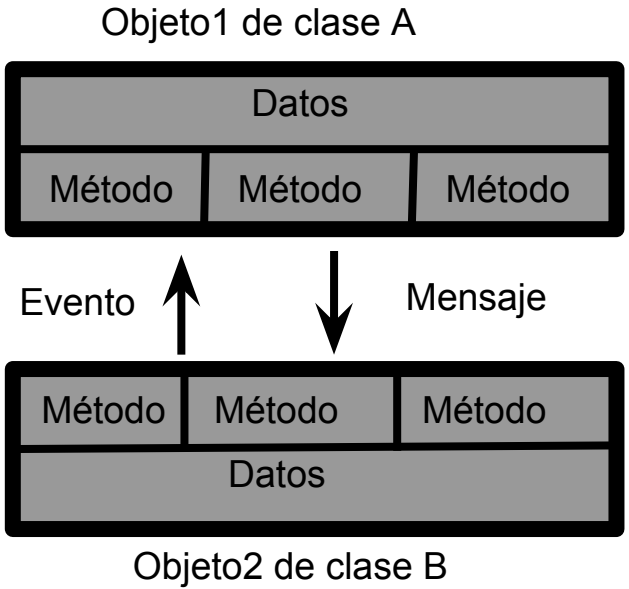
1. ***public***: acceso desde cualquier contexto (clase). Cualquier objeto puede hacer uso de este recurso.
2. ***protected***: acceso desde la clase propietaria y las subclases.
3. ***private***: acceso sólo desde la clase propietaria. Sólo los objetos de la misma clase pueden hacer uso del recurso.

Diagrama de Objetos con acceso público



Eventos y mensajes entre objetos

Un ***evento*** se genera cuando un objeto usa los métodos de otro objeto.
Mensaje es la respuesta del mensaje cuyo método fue invocado.



El acceso y modificación a los datos de un objeto debe hacerse a través de sus métodos.

Declaraciones con modificadores de visibilidad

```
class Persona {
    private $nombre = "Pedro";
    private $apellidoPat = "Juárez";
    private $apellidoMat = "Romero";

    public function saludaA($nombre) {
        return "Hola ".$nombre."!";
    }
}
```

Métodos constructores

- Su nombre debe ser **__construct()**
- Es llamado automáticamente cada vez que un objeto de la clase es creado.
- **No puede tener** un enunciado return.

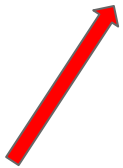
Métodos constructores

- PHP proporciona un método constructor por omisión cuando el programador no lo hace.
- No acepta como métodos constructores métodos que tengan el mismo nombre de la clase.
- Los métodos constructores no se invocan de manera explícita:

Instanciación

```
// Creación de un objeto

$personaNueva = new Persona();
```



Invocación del constructor

Métodos constructores: pseudovariable $this

$this es un mecanismo por el cual una clase se refiere a una instancia y tiene acceso a sus miembros.

Para acceder a las miembros del objeto se necesita el operador **->** seguido del nombre del atributo o método que se requiera.

Propiedades estáticas

Definición:

> Propiedades propias de la clase, es decir, no necesitan la instanciación de ningún objeto para existir.

Propiedades estáticas

Sintáxis:

<Modificador_de_visibilidad> static $<nombre> = <valor>;

Donde:

- **Modificador_de_visibiliad** es *public*, *protected* o *private*.
- **nombre** es una cadena de *caracteres*, *dígitos* y _ pero no puede empezar con dígito.
- **static** es una palabra reservada del lenguaje.
- **valor** debe ser un *valor constante*, no puede ser expresiones o resultados de funciones.

Propiedades estáticas: ejemplo

```
Class Persona{

        public static $poblacion = 0;

}

echo Persona::$poblacion;
```

Métodos estáticos

Definición:

> Métodos propios de la clase, es decir, no necesitan la instanciación de ningún objeto para poder ser usados.

Métodos estáticos

Sintáxis:

<Modificador_de_visibilidad> static function <nombre> ($arg1,...,$argN)

Donde:

- **Modificador_de_visibiliad** es *public*, *protected* o *private*.
- **static** es una palabra reservada del lenguaje.
- **function** es una palabra reservada del lenguaje.
- **nombre** es una cadena de *caracteres*, *dígitos* y _ pero no puede empezar con dígito.

Métodos estáticos: ejemplo

```
Class Persona{


        public static function getDefinicion(){
                return “Individuo de la especie humana.”;
        }

}

Persona::getDefinicion();
```

Métodos estáticos

La pseudovariable **$this** no está disponible dentro de un método estático ya que la declaración del método en esta forma supone que se mandará a llamar fuera desde la definición de la clase y no desde un objeto de la misma.

Valores constantes

Definición:

Son valores constantes propios de una clase que permanecen invariables.

Valores constantes

Sintáxis:

```
const <nombre> = <valor>;
```

Donde:

- **const** es una palabra reservada del lenguaje.
- **nombre** es una cadena de *caracteres*, *dígitos* y _ pero no puede empezar con dígito.
- **valor** debe ser un *valor constante*, no puede ser expresiones o resultados de funciones.

Operador de resolución de ámbito (::)

- Nombrado como *Paamayinm Nekudotayim*. Dos puntos en hebreo.
- Se utiliza para acceder a elementos *estáticos* o *constantes* de una clase desde fuera o dentro de ella.
  - Uso en contexto externo:

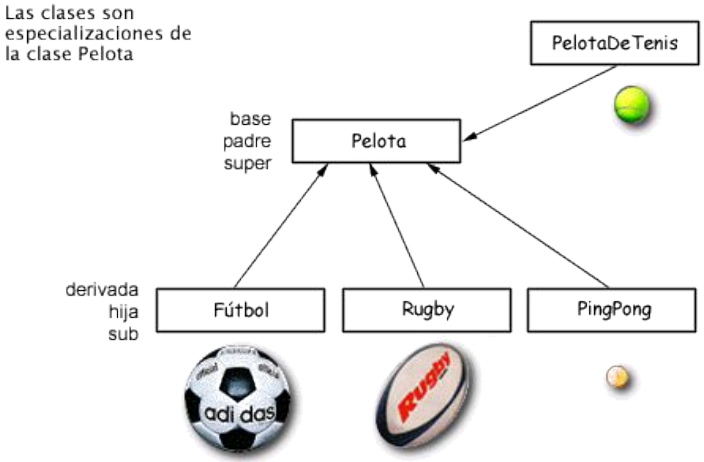
    <Nombre de la Clase>::<elemento>

  - Uso en contexto interno:

    self::<elemento>

Superclases y Subclases

La clase de la que se hereda se suele llamar **clase *base*, *clase padre* o *superclase***, mientras que la clase que la extiende se conoce como **clase *derivada*, *clase hija* o *subclase*.**



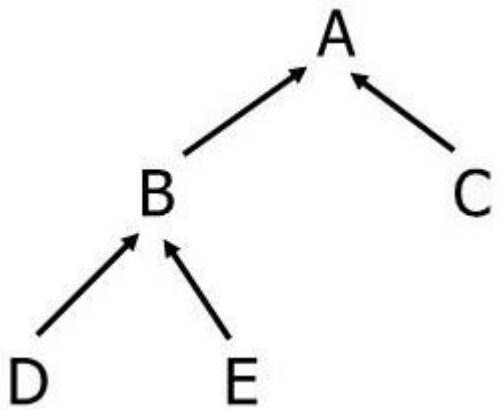
Tipos de Herencia

- Herencia simple

- Herencia múltiple

Herencia simple

Una clase hereda de una única clase



Herencia simple

- **La subclase copia el comportamiento del padre, siempre y cuando no sobreescriba métodos.**

- **La subclase puede agregar comportamiento específico.**

- **La subclase puede ocupar el constructor del padre dentro de su constructor si así lo requiere.**
  `parent::__construct($a, …, $n);`

HERENCIA: EJEMPLO

```
class Persona {

    public function __constructor(<parametrosP>){ //bla }

    public function saludar(){
         echo ”Hola”;
    }

    //Otros métodos
}
```

HERENCIA: EJEMPLO

```
class Italiano extends Persona {

    public function __construct(<parametrosP>,
                                <parametrosH>){
        parent::__construct(<parametrosP>);
        // algo más con <parametrosH>
    }

    public function saludar() {

        return “Ciao!”;

    }
```

Ejercicio

- **Implementa la clase usuario que se encuentra en el archivo Usuario.php**
- **Extiende la clase Exception (http://php.net/manual/en/language.exceptions.extending.php)**

Ejercicio 2

- **Extiende la clase Exception (http://php.net/manual/en/language.exceptions.extending.php) para pesonalizar el error de división entre 0 y utiliza la nueva excepción en la función de calcula.**