



Clase presencial



Sesion2C: Programación Orientada a Objetos

Instructor: David Paúl Porras Córdova

@iscodem



Objetivo General

- Aplicar las teorías y conceptos de Programación Orientada a Objetos.



Contenido de Agenda

- Clases y objetos
 - ▣ Clases
 - ▣ Objetos
 - ▣ Atributos, métodos y constantes
 - ▣ Constructores
 - ▣ Interfaces y diferencia con clases
- Principios básicos de la POO
 - ▣ Encapsulamiento
 - ▣ Herencia
 - ▣ Polimorfismo



Clases y objetos

- Clases
- Objetos
- Atributos, métodos y constantes
- Constructores
- Interfaces y diferencia con clases

1. Clases

Una Clase es una plantilla o prototipo que define los atributos y métodos de un objeto.

ClaseCelular
-color -peso -ancho -alto -grosor
+enviaMensaje() +recibeMensaje() +seleccionDeTono() +seleccionDeFondo() +llamada()



2. Objetos

Un objeto es una unidad de código con atributos y métodos predefinidos.



Laboratorio: 5.1. Creación de la clase Convierte

- Ejercicio 1:
 - ▣ Tiempo: 60 minutos
 - ▣ Mediante la teoría de clases y objetos, cree la clase **Convierte** que se encargue de convertir cadenas a valores y viceversa.

3. Atributos, métodos y constantes

```
private Double radio;
```

```
public Double areaCirculo() {  
    Double area = PI * Math.pow(getRadio(), 2);  
    return area;  
}
```

```
private final Double PI = Math.PI;
```


4. Constructores

- Son métodos de clase que se ejecutan automáticamente, cada vez que se instancia un objeto.

```
public MiembrosDeClase() {  
}  
  
public MiembrosDeClase(Double radio) {  
    this.radio = radio;  
}
```

5. Interfaces y diferencia con clases

- **Interface** es la planificación de la aplicación.
- **Clase** es la implementación de la aplicación.
- Para crear una interface se usa la palabra reservada **interface**.
- Para crear una clase se usa la palabra reservada **class**.

Principios básicos de la POO

- Encapsulamiento
- Herencia
- Polimorfismo

6. Encapsulamiento

- ❑ Modularidad
- ❑ Ocultamiento de la información
- ❑ Las clases proveen el beneficio de la reutilización
- ❑ Los objetos se transfieren datos entre sí (conversan).

7. Herencia

- Se puede extender una clase preservando sus características y operaciones para añadir funcionalidad.
- Mediante la palabra reservada **extends**, se extiende una clase Visual Studio.

```
public class Baldor extends Convierte
```

Laboratorio: 5.2. Creación de la clase Baldor

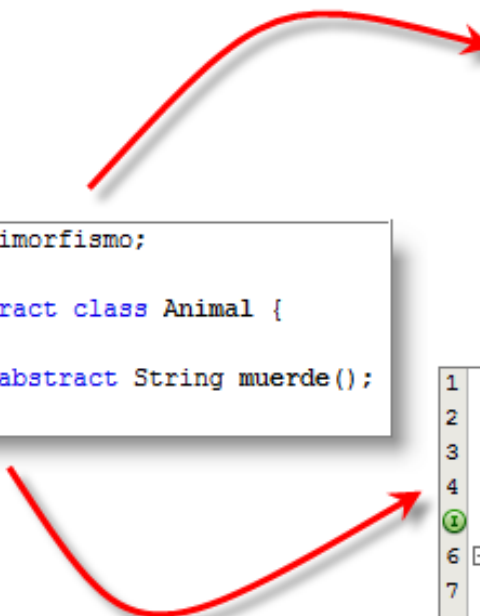
- Ejercicio 2:
 - ▣ Tiempo: 60 minutos
 - ▣ Mediante la teoría de clases y objetos, cree la clase Baldor extendida de la clase Convierte que se encargue de sumar, restar, multiplicar y dividir.

Laboratorio: 5.3. Creación de la clase Formato

- Ejercicio 2:
 - ▣ Tiempo: 60 minutos
 - ▣ Mediante la teoría de clases y objetos, cree la clase Formato extendida de la clase Convierte que se encargue de dar formato de Soles (S/.) a un valor y también formato con N decimales.

8. Polimorfismo

```
1 package polimorfismo;  
2  
3 public abstract class Animal {  
4  
5     public abstract String muerde();  
6 }
```



```
1 package polimorfismo;  
2  
3 public class Pulga extends Animal {  
4  
5     @Override  
6     public String muerde() {  
7         return "Pulga picando!";  
8     }  
9 }
```

```
1 package polimorfismo;  
2  
3 public class Tiranosaurio extends Animal {  
4  
5     @Override  
6     public String muerde() {  
7         return "Tiranosaurio mordiendo!";  
8     }  
9 }
```




Resumen del Capítulo

- La programación orientada a objetos, trata de emular el comportamiento humano de tal forma, que programar sea algo común para nosotros.