

Clases abstractas y polimorfismo




Clases abstractas

- Es una clase que cuenta, al menos, con un método abstracto, que es aquel que está declarado en la clase pero no implementado.
- Tanto la clase como los métodos abstractos se definen con la palabra *abstract*

```
abstract class Clase1{  
    public abstract int calculo();  
}
```

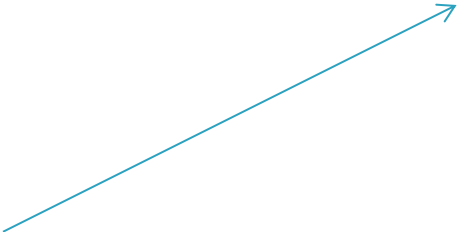
El método abstracto no tiene código, solo se declara

Características clases abstractas


- No es posible crear objetos de una clase abstracta.
 - Además de métodos abstractos, las clases abstractas pueden incluir atributos, constructores y métodos estándares
 - Una clase que herede una clase abstracta está obligada a sobrescribir los métodos abstractos heredados (o declararse también como abstract)
 - El objetivo de los métodos abstractos es forzar a que todas las subclases tengan el mismo formato de método
- 

Ejemplo

```
abstract class Figura{  
    private String color;  
    public Figura(String color){  
        this.color=color;  
    }  
    public abstract double area();  
}
```



```
class Circulo extends Figura{  
    private int radio;  
    public Circulo(String color, int radio){  
        super(color);  
        this.radio=radio;  
    }  
    public double area(){  
        return Math.PI*radio*radio;  
    }  
}
```

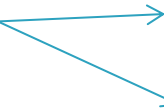


```
class Triangulo extends Figura{  
    private int base,altura;  
    public Triangulo(String color, int base, int altura){  
        super(color);  
        this.base=base;  
        this.altura=altura;  
    }  
    public double area(){  
        return base*altura/2;  
    }  
}
```

Polimorfismo

- Se basa en la asignación de referencias a objetos de subclases en variables de su superclase (abstractas o no)
- Consiste en utilizar una misma instrucción para llamar a diferentes versiones de un mismo método

Misma instrucción,
llama a diferentes
métodos



```
Figura f=new Triangulo(...);  
f.area();// método área de Triangulo  
f=new Circulo(...);  
f.area();// método área de Circulo
```

- Principales ventajas: reutilización de código, flexibilidad, dinamismo

Métodos abstractos vs finales

- Lo contrario a un método abstracto es un método final.
- Un método final es aquel que no puede ser sobrescrito. El modificador *final* se utiliza delante del tipo

```
class Clase1{  
    public final int calculo(){}  
}  
class Clase2 extends Clase1{  
    public int calculo{} //error de compilación  
}
```