

Herencia

Definición

- Herencia es una característica que permite crear nuevas clases a partir de clases ya existentes, de forma que la nueva clase adquiera (herede) los miembros de la ya existente.
- A la clase “padre” se le conoce como superclase, mientras que la “hija” es la subclase
- Se emplea **extends**:

```
class Clase1{  
    public void metodo(){}  
}  
class Clase2 extends Clase1{  
    //automaticamente adquiere metodo()  
}
```

El principal beneficio de la herencia es la reutilización de código

Consideraciones

- Una clase solo puede heredar otra clase, aunque la superclase puede heredar a su vez a otra y así hasta n niveles.
- Varias clases pueden heredar la misma clase.
- Los miembros privados de la superclase no son accesibles directamente desde la subclase
- Si queremos que una clase no se pueda heredar, la definiremos con *final*:

```
final class Clase1{ //no podrá ser heredada
    public void metodo(){}
}
class Clase2 extends Clase1{} //error de compilación
```

Herencia de Object

- Todas las clases Java heredan Object
- Si una clase no hereda explícitamente otra clase, implícitamente heredarán Object

`class Test{`
`}`  `class Test extends Object{`
`}`

- Todas las clases disponen de los métodos de Object, entre ellos: `toString()`, `equals()` y `hashCode()`

Clases abstractas


Clases abstractas

- Es una clase que puede incluir métodos abstractos, que son aquellos que están declarados en la clase pero no implementados.
- Tanto la clase como los métodos abstractos se definen con la palabra *abstract*

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

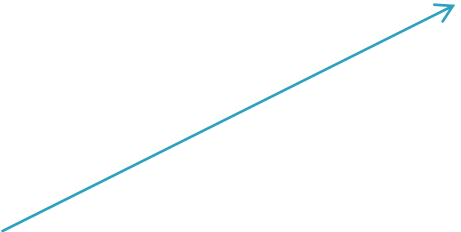
El método abstracto no tiene código, solo se declara

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)
 - Una clase abstracta no está obligada a tener métodos abstractos
- 

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;  
    }  
}
```


Interfaces

Definición

- Una interfaz es un conjunto de métodos abstractos.
- Su objetivo es definir el formato de ciertos métodos, que posteriormente las clases se encargarán de implementar.
- También puede incluir constantes, que serán públicas y estáticas

Creación de una interfaz

- Una interfaz se crea, al igual que las clases, en archivos .java.
- Se define con la palabra reservada *interface*

Como los métodos solo pueden ser públicos y abstractos, se pueden omitir las palabras *abstract* y *public*

```
public interface Operaciones{  
    int k=10;  
    void girar(int grados);  
    int invertir();  
}
```

En el caso de las constantes, se omiten las palabras *public*, *final* y *static*

Implementación de una interfaz

- Una clase que implementa una interfaz está obligada a sobrescribir (implementar) todos los métodos de la misma.

Operaciones



Test

Al implementar los métodos en la clase, es obligatorio indicar el modificador *public*.

```
public class Test implements Operaciones{  
    public void girar(int grados){  
        :  
    }  
    public int invertir(){  
        :  
    }  
}
```

Herencia múltiple en interfaces

➤ Una interfaz puede heredar una o varias interfaces:

```
public interface Operaciones{  
    void girar(int grados);  
    int invertir();  
}  
public interface Inter1{  
    int miMetodo();  
}  
public interface InterFin extends Operaciones, Inter1{  
    void nuevoMetodo();  
}
```

```
public class Prueba implements InterFin{  
    public void girar(int grados){...}  
    public int invertir(){...}  
    public int miMetodo(){...}  
    public void nuevoMetodo(){...}  
}
```

La clase está obligada a implementar los métodos de la interfaz que implementa y los de las interfaces que esta hereda