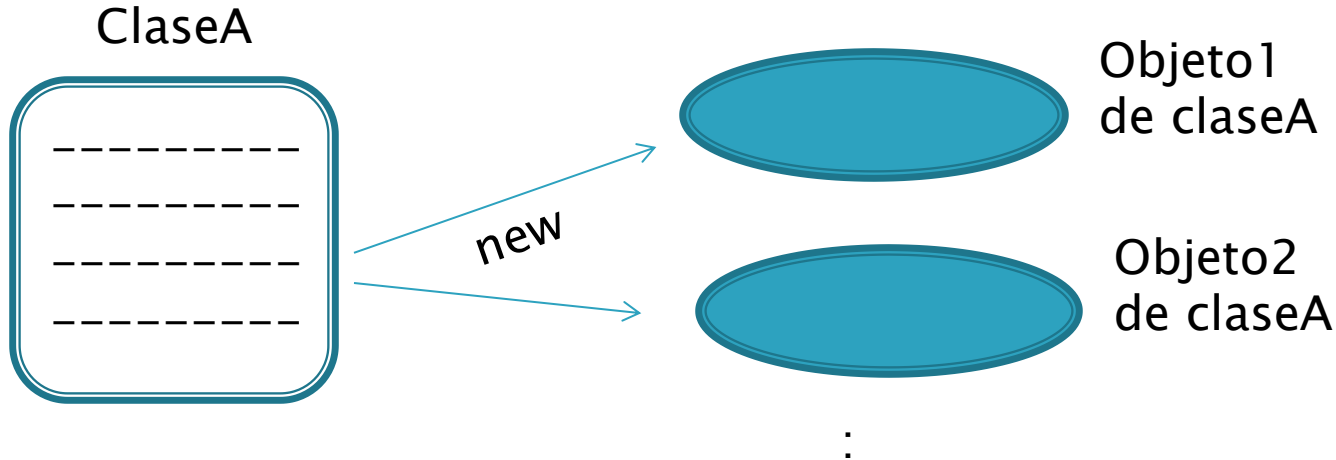


# Clases, métodos, constructores

# Clases y objetos

- Una clase define el comportamiento de un determinado tipo de objeto.
- La clase es el molde y el objeto el elemento “físico” obtenido del molde.



```
ClaseA obj=new ClaseA();  
obj.metodo();
```

# Sobrecarga de métodos

➤ Una clase puede contener varios métodos con el mismo nombre, pero deben diferenciarse en el número y/o tipo de parámetros:

```
public int sumar(int a, int b){..}  
public int sumar(int a){..}  
public int sumar(long b){..}
```

➤ El tipo de devolución no afecta en la sobrecarga, puede ser el mismo o diferente

# Llamadas a métodos sobrecargados

➤ La versión del método que será llamado se determina en función de los argumentos de la llamada:

sumar(3,9); —————> public int sumar(int a, int b){..}

sumar(10); —————> public int sumar(int a){..}

sumar(7L); —————> public int sumar(long b){..}

# Ejemplos

*Casos válidos  
de sobrecarga:*

```
public int imprimir(int a){..}
```

```
public void imprimir(){..}
```

```
public int imprimir(long b){..}
```

*Casos no válidos  
de sobrecarga:*

```
public int imprimir(int a){..}
```

```
//error compilación
```

```
public void imprimir(int s){..}
```

```
//no error, pero no sobrecarga
```

```
public int Imprimir(int b){..}
```



# Precaución

➤ Cuando hay varios posibles métodos que se pueden ejecutar en una llamada: primero se intenta coincidencia exacta, después promoción de tipos y en último lugar autoboxing

metodo(4); —————> void metodo(int a)  
void metodo(Integer e);

metodo(4); —————> void metodo(long a)  
void metodo(Integer e);

metodo(4); —————> void metodo(Long a)  
void metodo(Integer e);

# Métodos estáticos

➤ Son métodos que no están asociados a ningún objeto particular de la clase. Pertenecen a la clase, no a la instancia

➤ Se declaran con la palabra `static`:

```
class Calc{  
    public static int cuadrado(int a){  
        return a*a;  
    }  
}
```

➤ No es necesario crear un objeto para llamar a estos métodos, se utiliza el nombre de la clase:

```
int r=Calc.cuadrado(4);
```

Aunque es la forma habitual de usarlos, también se les puede llamar con cualquier instancia de la clase

# Consideraciones métodos estáticos

- Solo pueden llamar a otros miembros de su misma clase que también sean static

```
class Test{  
    int a=2;  
    static int b=5;  
    public static int metodo(){  
        int c=a*3; // error de compilación  
        int n=b+1; //ok  
        imprime(n); //ok  
    }  
    static void imprime(int s){..  
}
```

- No se puede usar en su interior ni *this* ni *super*



# Atributos estáticos

- Son compartidos por todos los objetos de la clase. Se accede a ellos a través de la clase
- Se definen con la palabra `static`

```
class Test{  
    static int n=0;  
    public void inc(){  
        n++;  
    }  
    public int getN(){return n;}  
}
```

```
class Prueba{  
    public static void main(String[] ar){  
        Test t1=new Test();  
        t1.inc();  
        Test t2=new Test();  
        t2.inc();  
        System.out.println(t1.getN()); //2  
        System.out.println(t2.getN()); //2  
    }  
}
```

# Bloque estáticos

- Se ejecutan una vez durante la vida de una clase
- Solo puede acceder a otros miembros estáticos

```
class Test{  
    static int n=0;  
    static{  
        n++;  
    }  
    public int getN(){return n;}  
}
```


```
class Prueba{  
    public static void main(String[] ar){  
        Test t1=new Test();  
        Test t2=new Test();  
        System.out.println(t1.getN()); //1  
        System.out.println(t2.getN()); //1  
    }  
}
```

# Constructores

- Bloques de código que se ejecutan al crear un objeto de la clase.
- Como los métodos, pueden recibir parámetros, aunque no tienen tipo de devolución y su nombre siempre es igual al de la clase:


```
class Calc{  
    public Calc(){  
  
    }  
}
```

Calc c=new Calc();



```
class Test{  
    public Test(int n){  
  
    }  
}
```

Test t=new Test(10);



# Constructor por defecto

➤ Si no se define un constructor de forma explícita en una clase, el compilador añade el llamado constructor por defecto, que no tiene parámetros y tampoco ninguna instrucción:

```
class Test{  
}
```



```
class Test{  
    public Test(){}  
}
```

```
Test t=new Test(); //ok
```

➤ Si se define explícitamente un constructor, el compilador ya no crea el constructor por defecto:

```
class Test{  
    public Test(int m){}  
}
```

```
Test t=new Test(); //error compilación
```

# Sobrecarga de constructores

- Una clase puede incluir varios constructores que permitan inicializar los objetos de diferente forma
- Se siguen las mismas reglas que con la sobrecarga de métodos.
- Ejemplo:

```
Test t1=new Test();  
Test t2=new Test(5);  
Test t3=new Test(3, 1);
```

```
class Test{  
    public Test(){}  
    public Test(int a){}  
    public Test(int a, int b){}  
}
```

