

# MÉTODOS

- Un **método** en Java es un conjunto de instrucciones definidas dentro de una clase, que realizan una determinada tarea y a las que podemos invocar mediante un nombre.
- Algunos métodos que hemos utilizado hasta ahora:
  - Math.pow()
  - Math.sqrt()
  - System.out.println();
- Cuando se llama a un método, la ejecución del programa pasa al método y cuando éste acaba, la ejecución continúa a partir del punto donde se produjo la llamada.
- Utilizando métodos:
  - Podemos construir programas modulares.
  - Se consigue la reutilización de código.

```
1
2 public class ClasePrincipal {
3     public static void main(String args[]){
4         ClasePrincipal objeto = new ClasePrincipal();
5         objeto.xxxValue();
6         objeto.parseValueOf();
7         objeto.metodoToString();
8     }
9
10    public void metodoToString(){
11        Double d= new Double("1234");
12        String dCadena = d.toString();
13        System.err.println(dCadena);
14    }
15
16    public void parseValueOf(){
17        double d4 = Double.parseDouble("3.14");
18        long primitivoLong = Long.parseLong("323");
19        Long objetoLong=Long.valueOf("456");
20        System.err.println(d4 + 1);
21        System.err.println(primitivoLong + 1);
22        System.err.println(objetoLong.longValue());
23    }
24
25    public void xxxValue(){...13 lines }
26
27 }
28
29
```

# MÉTODOS

- SU SINTAXIS ES LA SIGUIENTE:

## Syntax

```
[modifiers] return_type method_identifier ( [arguments] ) {  
    method_code_block  
}
```

- **Especificadores o modifiers** (opcional): determinan el tipo de acceso al método. Se verán en detalle más adelante.
- **tipoDevuelto o return\_type**: indica el tipo del valor que devuelve el método. Si el método no devuelve ningún valor este tipo será void.
- **nombreMetodo o method\_identifier**: es el nombre que se le da al método. Para crearlo hay que seguir las mismas normas que para crear nombres de variables.
- **Lista de parámetros o arguments** (opcional): después del nombre del método y siempre entre paréntesis puede aparecer una lista de parámetros (también llamados argumentos) separados por comas.

**return:** se utiliza para devolver un valor. El tipo del valor de retorno debe coincidir con el tipoDevuelto que se ha indicado en la declaración del método. Si el método no devuelve nada (tipoDevuelto = void) la instrucción return es opcional.

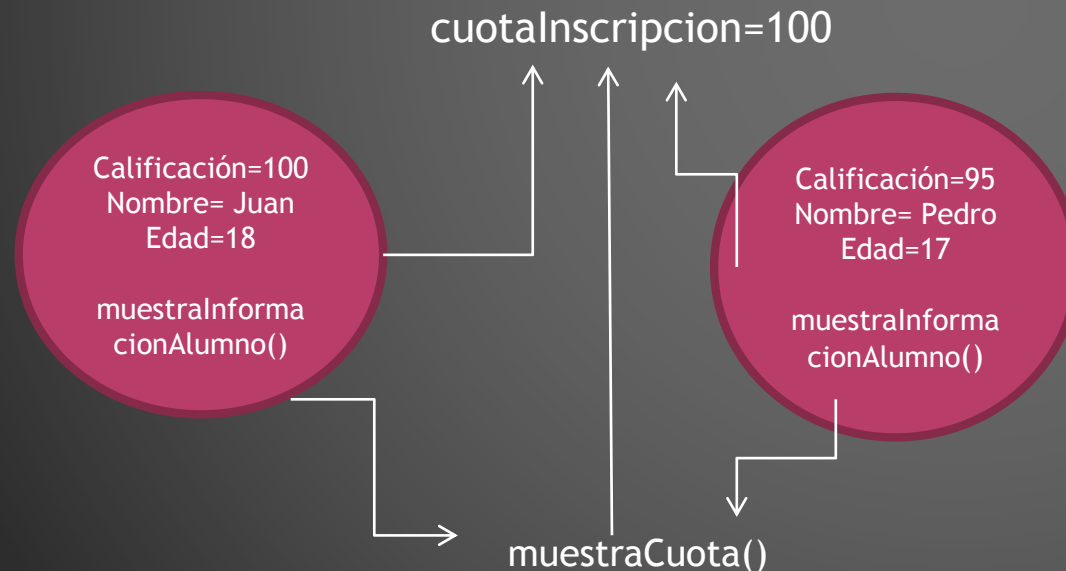
# MÉTODOS

## CONSIDERACIONES EN EL USO DE MÉTODOS:

- NO HAY LIMITE EN LAS LLAMADAS A LOS MÉTODOS
- SE PUEDEN LLAMAR MÉTODOS DE LA MISMA CLASE O DE DIFERENTE CLASE
- SI ESTAMOS EN LA MISMA CLASE SE PUEDE LLAMAR LOS MÉTODOS SIN CREAR UN OBJETO, SIN EMBARGO CUANDO ES DE OTRA CLASE, TENEMOS QUE CREAR UN OBJETO.
- NO EXISTE UN ORDEN EN LA CREACIÓN DE LOS MÉTODOS
- EN JAVA UN MÉTODO SIEMPRE PERTENECE A UNA CLASE

# MÉTODOS Y ATRIBUTOS ESTÁTICOS

- Podemos definir estos elementos como pertenecientes a la clase, en lugar de pertenecientes a la instancia, por eso se les llama elementos de la clase.
- Un método de clase es aquel que puede ser invocado sin existir una instancia.
- Se puede usar en variables y en métodos.
- Existe el concepto estático y dinámico
- Contexto Dinámico puede acceder al estático, y el estático no accede al dinámico



```
public class Alumno {  
    int calificacion;  
    String nombre;  
    int edad;  
    static double cuotaSemestre;  
    public Alumno(int cal,String nomb, int ed){  
        calificacion=cal;  
        nombre = nomb;  
        edad = ed;  
    }  
    public void muestraInformacionAlumno(){  
        System.out.print("Nombre: " + nombre);  
        System.out.println("    Calificación: " + calificacion);  
        System.out.println("    Edad: " + edad);  
    }  
    public static void muestraCuota(){  
        System.out.println("La cuota por alumno para este semestre"  
            + "es de: " + cuotaSemestre);  
    }  
}
```

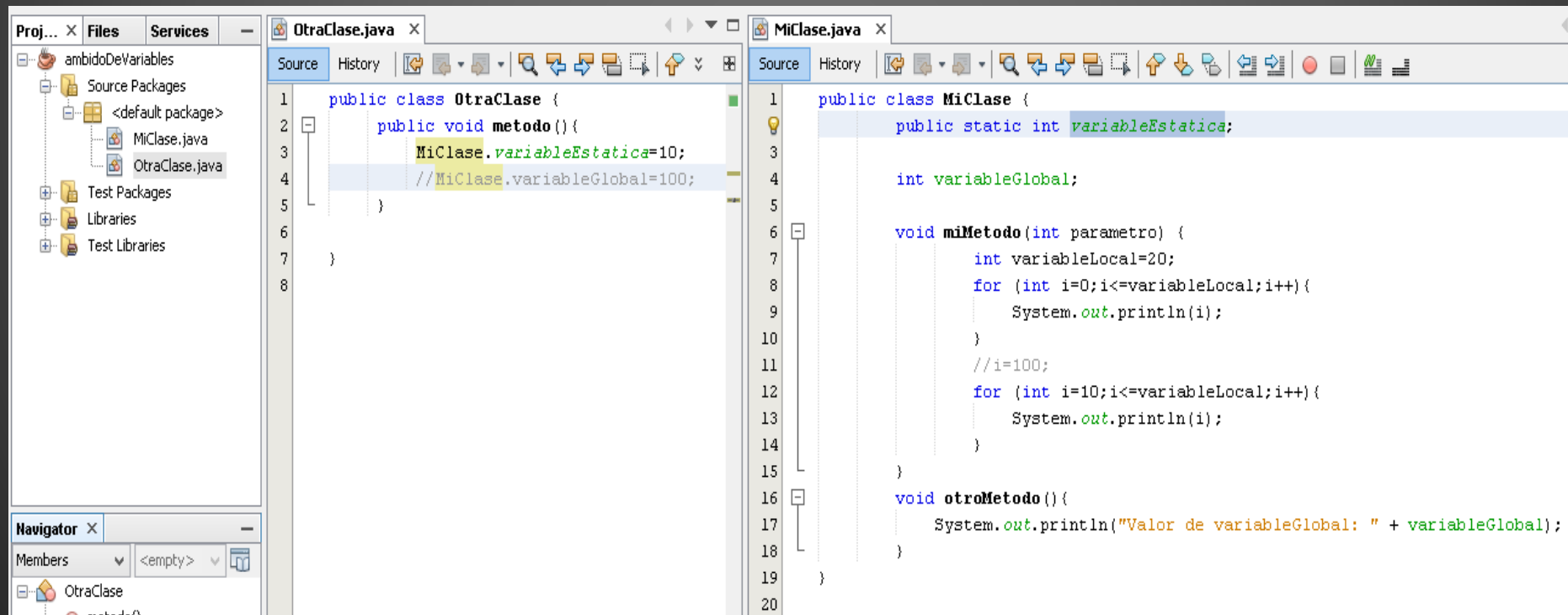
# SOBRECARGA DE MÉTODOS EN JAVA (OVERLOADING)

- Un método sobrecargado se utiliza para reutilizar el nombre de un método pero con diferentes argumentos (opcionalmente un tipo diferente de retorno).
- Los métodos sobrecargados deben de cambiar la lista de argumentos
- Pueden cambiar el tipo de retorno.
- Lo que define qué método es el que se va a llamar son los argumentos que se envían al mismo durante la llamada.
- También existe la sobrecarga de constructores.

```
//SOBRECARGA CONSTRUCTOR
public Alumno()
public Alumno(String nomb)
//SOBRECARG MÉTODOS
public void setDatosAlumno(String nomb,int ed, int cal)
public void setDatosAlumno(String nomb)
public void setDatosAlumno(String nomb, int ed)
public void setDatosAlumno(int cal)
```

# ALCANCE DE VARIABLES

- El *ámbito* de una variable define su *alcance de uso*.
- Existen tres tipos de ámbito
  1. Local
  2. Global
  3. Estático.



```
OtraClase.java
1 public class OtraClase {
2     public void metodo() {
3         MiClase.variableEstatica=10;
4         //MiClase.variableGlobal=100;
5     }
6 }
7
8

MiClase.java
1 public class MiClase {
2     public static int variableEstatica;
3
4     int variableGlobal;
5
6     void miMetodo(int parametro) {
7         int variableLocal=20;
8         for (int i=0;i<=variableLocal;i++){
9             System.out.println(i);
10        }
11        //i=100;
12        for (int i=10;i<=variableLocal;i++){
13            System.out.println(i);
14        }
15    }
16    void otroMetodo(){
17        System.out.println("Valor de variableGlobal: " + variableGlobal);
18    }
19 }
20
```