**Overloading names**

/\*\* = the absolute value of b. \*/

static double abs​(double b)

/\*\* = the absolute value of b. \*/

static float abs(float b)

/\*\* = the absolute value of b. \*/

static int abs​(int b)

/\*\* = the absolute value of b. \*/

static long abs​(long b)

Class Math in package java.lang has lots of functions for performing basic numeric operations. You will no doubt use class Math often. Four of its functions are shown to the right. They calculate the absolute value of a **double** value, a **float** value, an **int** value, and a **long** value.

All four function are named abs! *g*. In certain circumstances, several methods can have the same name. This is called *over­loading*.

This is also an example of polymorphism: the four functions do the same thing —calculate an absolute value— but they have different forms. Remember, polymorphism is the capability of assuming different forms.

Overloading like this is called *ad-hoc* polymorphism.

When there is a call like Math.abs(-5) in a program, the argument, -5, is used to determine which function to call. Since -5 has type **int**, the function with an **int** parameter will be called.

You may wonder whether *you* can use overloading in your programs. Yes, of course! We illustrate this with class Counter to the right, which *you* might have written. An instance of Counter maintains a counter. The counter is initially 0, and it can be incremented with a call on procedure increment.

/\*\* an instance maintains a counter,  
 which is initially 0. \*/

class Counter {

private int ctr; // the counter

/\*\* = the value of the counter \*/

public int ctr() { return ctr; }

/\*\* increment the counter \*/

public void increment()  
 { ctr= ctr + 1; }

/\*\* reset the counter to 0 \*/

public void reset() { ctr= 0; }

/\*\* reset the counter to i \*/

public void reset(int i) { ctr= i; }

}

}

There are two reset procedures. One sets the counter to 0, the other sets the counter to its parameter i. The name reset is overloaded. Both procedures reset the counter, but they have different forms. That is polymorphism.

There is another form of polymorphism and overloading in class Counter. The field name is ctr, and method ctr () returns the value of field ctr. Yes indeed, the field name can be the same as a method name.

**+ is overloaded**

Here is another form of overloading. The value of expression 2 + 3 is the integer 5; operator + stands for **int** addition. The value of the expression "2" + 3 is "23"; here, since at least one operand is a String, + means string catenation.

The programmng language Python allows the programmer to overload operators like + and – and \* with other meanings. Java, however, does not allow this.