```
 1 public class MinTest
 2 {
 3     public static void main( String [ ] args )
 4     {
 5         int a = 3;
 6         int b = 7;
 7
 8         System.out.println( min( a, b ) );
 9     }
10
11     // Method declaration
12     public static int min( int x, int y )
13     {
14         return x < y ? x : y;
15     }
16 }
```

Figure 1.6 Illustration of method declaration and calls

The *`return` statement* is used to return a value to the caller. If the return type is `void`, then no value is returned, and `return;` should be used.

The `return` *statement* is used to return a value to the caller.

1.6.1 Overloading of Method Names

Suppose we need to write a routine that returns the maximum of three `int`s. A reasonable method header would be

```
int max( int a, int b, int c )
```

In some languages, this may be unacceptable if `max` is already declared. For instance, we may also have

```
int max( int a, int b )
```

Java allows the *overloading* of method names. This means that several methods may have the same name and be declared in the same class scope as long as their *signatures* (that is, their parameter list types) differ. When a call to `max` is made, the compiler can deduce which of the intended meanings should be applied based on the actual argument list. Two signatures may have the same number of parameters, as long as at least one of the parameter list types differs.

Overloading of a method name means that several methods may have the same name as long as their parameter list types differ.

Note that the return type is not included in the signature. This means it is illegal to have two methods in the same class scope whose only difference is the return type. Methods in different class scopes may have the same names, signatures, and even return types; this is discussed in Chapter 3.