Types Are Defined Implicitly

Although everything in Smalltalk, even classes, is an object, classes do not correspond to types in the way they do in languages such as C++ and Java. Types are defined implicitly by what they do, and by their interfaces. This is described by names such as *latent typing* or *duck typing*.

Latent typing is the only typing mechanism in Smalltalk (and also in some other dynamically typed languages), but that does not mean it is of no importance to strongly typed languages. In C++, for instance, latent typing is the basis of generic programming via templates. It makes sense to see it first in that language. Take a look at the following introductory example of C++ templates (Vandervoorde and Josuttis 2002, 2.4):

```
// maximum of two int values
inline int const& max (int const& a, int const& b)
{
    return  a < b ? b : a;
}

// maximum of two values of any type
template <typename T>
inline T const& max (T const& a, T const& b)
{
    return  a < b ? b : a;
}

// maximum of three values of any type
template <typename T>
inline T const& max (T const& a, T const& b, T const& c)
{
    return ::max (::max(a,b), c);
}

int main()
{
    ::max(7, 42, 68);     // calls the template for three arguments
    ::max(7.0, 42.0);     // calls max<double> (by argument deduction)
    ::max('a', 'b');      // calls max<char> (by argument deduction)
    ::max(7, 42);         // calls the nontemplate for two ints
    ::max<>(7, 42);       // calls max<int> (by argument deduction)
    ::max<double>(7, 42); // calls max<double> (no argument deduction)
    ::max('a', 42.7);     // calls the nontemplate for two ints
}
```

As we see in `main()`, the function `::max` will work for any type that implements the comparison operator. In C++, that type may be a primitive, or it may be user-defined. There are no restrictions that it should inherit from a specific class. It can be of any type, as long as it obeys the basic requirement on comparison. The implicit type definition is: anything for which operator < makes sense.

In school we learned that there are two ways to define a set. One is explicit, by enumerating its elements. This is called *extensional definition*. The set of natural numbers we know is