{1, 2, 3, ...}. The other way to define a set is by describing what the members of the set have in common. This is called *intentional definition*. The intensional definition of the set of even natural numbers is "whole numbers greater than zero." When we declare an object in C++, we are effectively using an extensional definition: we say that the object belongs to the given type. When we use templates, however, we are effectively using an intensional definition: we say that the set of objects for which this code is applicable is the set of objects that have the given properties (that is, they offer the required operations).

The situation is unfortunately muddled in Java. Java offers generics, but they bear only a surface syntactic resemblance to C++ templates. An eloquent explanation of the problem has been given by Bruce Eckel (see *http://www.mindview.net/WebLog/log-0050*). In a language such as Python, which supports latent typing, we can do this:

```
class Dog:
    def talk(self):  print "Arf!"
    def reproduce(self): pass

class Robot:
    def talk(self): print "Click!"
    def oilChange(self): pass

a = Dog()
b = Robot()
speak(a)
speak(b)
```

The two invocations of `speak()` will work without caring for the type of its argument. We can do the same with C++:

```
class Dog {
public:
    void talk() { }
    void reproduce() { }
};

class Robot {
public:
    void talk() { }
    void oilChange() { }
};

template<class T> void speak(T speaker) {
    speaker.talk();
}

int main() {
    Dog d;
    Robot r;
    speak(d);
    speak(r);
}
```