1.5 利用 Java 5 泛性实现泛型特性成分

Java 5 支持泛型类，这些类很容易使用。然而，编写泛型类却需要多做一些工作。本节将叙述编写泛型类和泛型方法的基础。我们不打算涉及语言的所有结构，那样将是相当复杂的，而且有时是很难处理的。我们将介绍用于全书的语法和习语。

1.5.1 简单的泛型类和接口

图 1-9 是前面图 1-5 描述的 MemoryCell 的泛型版代码。这里，我们把名字改成了 GenericMemoryCell，因为两个类都不在包中，所以名字也就不能相同。

```
1  public class GenericMemoryCell<AnyType>
2  {
3      public AnyType read( )
4        { return storedValue; }
5      public void write( AnyType x )
6        { storedValue = x; }
7
8      private AnyType storedValue;
9  }
```

图 1-9 MemoryCell 类的泛型实现

当指定一个泛型类时，类的声明则包含一个或多个类型参数，这些参数被放在类名后面的一对尖括号内。第 1 行指出，GenericMemoryCell 有一个类型参数。在这个例子中，对类型参数没有明显的限制，所以用户可以创建像 GenericMemoryCell<String>和 GenericMemoryCell<Integer>这样的类型，但是不能创建 GenericMemoryCell<int>这样的类型。在 GenericMemoryCell 类声明内部，我们可以声明泛型类型的域和使用泛型类型作为参数或返回类型的方法。例如在图 1-9 的第 5 行，类 GenericMemoryCell<String>的 write 方法需要一个 String 类型的参数。如果传递其他参数那将产生一个编译错误。

也可以声明接口是泛型的。例如，在 Java 5 以前，Comparable 接口不是泛型的，而它的 compareTo 方法需要一个 Object 作为参数。于是，传递到 compareTo 方法的任何引用变量即使不是一个合理的类型也都会编译，而只是在运行时报告 ClassCastException 错误。在 Java 5 中，Comparable 接口是泛型的，如图 1-10 所示。例如，现在 String 类实现 Comparable<String>并有一个 compareTo 方法，这个方法以一个 String 作为其参数。通过使类变成泛型类，以前只有在运行时才能报告的许多错误如今变成了编译时的错误。

```
1  package java.lang;
2
3  public interface Comparable<AnyType>
4  {
5      public int compareTo( AnyType other );
6  }
```

图 1-10 Java 5 版本的 Comparable 接口，它是泛型接口

1.5.2 自动装箱/拆箱

图 1-7 中的代码写得很麻烦，因为使用包装类需要在调用 write 之前创建 Integer 对象，然后才能使用 intValue 方法从 Integer 中提取 int 值。在 Java 5 以前，这是需要的，因为如果一个 int 型的量被放到需要 Integer 对象的地方，那么编译将会产生一个错误信息，而如果将一个 In-