第18章

原型（Prototype）模式

设计一个类时，通常会提供构造函数，使得客户端应用程序能够通过它去创建对象。但在某些情形下，可能不允许类的调用者直接调用构造函数。在构造型设计模式中，如构建者模式、工厂方法模式、抽象工厂模式都可以避免客户端直接调用构造函数。这些模式都是通过引入新类，代表客户端代码实例化合适的类对象。原型模式创建对象的方式与这些模式殊途同归。

原型模式的意图是通过复制一个现有的对象来生成新的对象，而不是通过实例化的方式。

作为工厂的原型

假定你在 Oozinoz 系统中使用抽象工厂模式为不同的应用场景提供用户界面。图 18.1 列出了所用到的用户界面工厂。

Oozinoz 系统可以为不同用户在不同环境下提供合适的界面。但是，一旦用户需要更多的用户界面工具包，则为每种应用场合创建一个新的用户界面工厂类的工作，就会变得非常繁重。因此，为了避免用户界面工厂类数量的不断膨胀，Oozinoz 系统的开发者建议采用原型模式：

- 删去 `UIKit` 的子类。
- 让 `UIKit` 类的每个实例成为能够分发控件原型副本的用户界面工厂。
- 把静态方法中创建的新 `UIKit` 对象的代码转移到 `UIKit` 类中。

在这种设计下，`UIKit` 对象将拥有完整的原型实例变量集合：一个按钮对象、一个网格对

象、一个面板对象等。创建新 UIKit 对象的代码将会为这些原型控件设置相应的属性去实现期望的界面风格。create-()方法将会返回这些原型控件的副本。

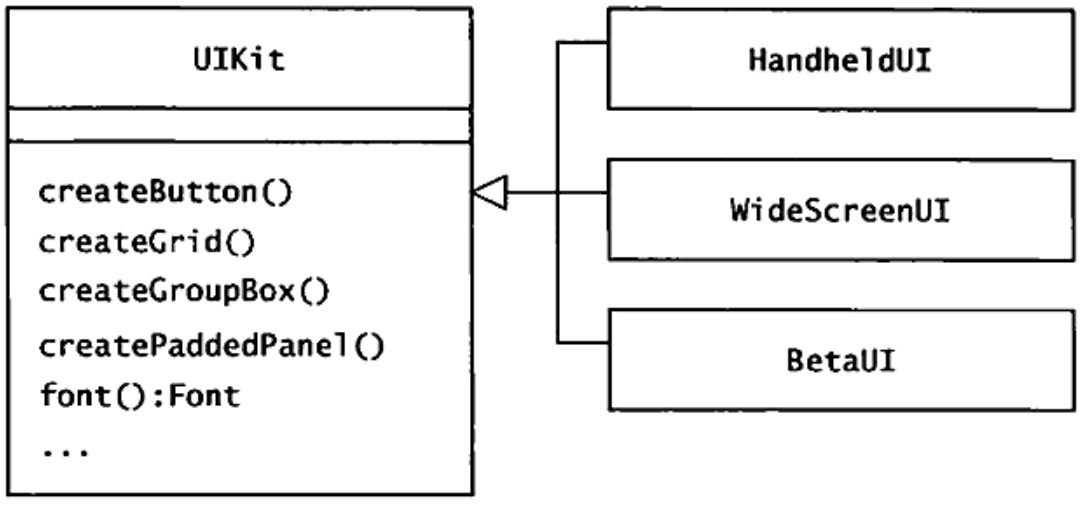

图 18.1　三种抽象工厂（工具包）产生三种不同的用户界面风格

例如，我们可以在 UI 类中定义一个静态的 handheldUI()方法。它将实例化 UIKit 对象，设置适合手持设备的对象实例变量，同时，返回用做 GUI 工具集的对象。

挑战 18.1

原型模式的使用会减少 Oozinoz 公司系统用来维护多个不同 GUI 工具箱的类数量。请列出这种设计方法的其他两个优点和缺点。

答案参见第 341 页

创建对象的常规方法是调用类的构造函数。原型模式提供了灵活的解决方案，你可以在运行时决定使用哪个对象作为一个模板去创建新的对象。然而，Java 程序开发中的原型方法并不允许对象拥有与父对象不同的方法。因此，可能需要重新考虑原型模式的利弊，甚至需要先试试原型模式是否符合需求。为了使用原型模式，需要掌握 Java 语言的复制对象机制。

利用克隆进行原型化

原型模式的意图是通过复制一个样本来创建新对象。当复制一个对象时，新对象将拥有与原对象相同的属性和方法。当然，这个新对象还可以继承部分甚至全部原对象的数据值。例如，

面板的副本应该具有与原面板相同的属性。

我们需要考虑一个关键问题：复制对象时，复制操作会提供原始对象属性值的副本，还是副本与原始对象共享这些属性值？作为一个应用系统开发者，这些问题往往被忽略甚至回答错误。如果开发者错误地理解复制对象机制，系统缺陷便会应运而生。Java 类库中的很多类都提供复制机制，但是，作为一个应用系统开发者，你必须理解复制的工作机制，尤其是当你想使用原型模式时。

挑战 18.2

所有的对象都可以从 Object 类继承 clone()方法。如果你不熟悉这个方法，可以从联机帮助或相关文档中了解。然后写下你对这个方法的理解。

答案参见第 341 页

挑战 18.3

假定 Machine 类有两个属性：整型的 ID 和 Location 对象，其中，Location 是一个单独的类。

设计一个对象图，说明 Machine 对象及它的 Location 对象，还有其他对象都是从 Machine 对象的 clone()方法创建而来的。

答案参见第 342 页

clone()方法使我们能很容易地往类中添加 copy()方法。比如，你可以使用如下代码创建可克隆面板的类：

```
package com.oozinoz.ui;
import javax.swing.JPanel;

public class OzPanel extends JPanel implements Cloneable {
    // 危险!
    public OzPanel copy() {
        return (OzPanel) this.clone();
    }
```

```
    // ...
}
```

上面代码中的 copy()方法使得大家都可以使用克隆功能，并且复制对象为需要的类型。然而，代码存在的问题是，clone()方法会创建某 JPanel 对象所有属性的副本，而不管你是否理解这些属性的功能。注意，JPanel 类的属性包括它的超类的所有属性。如图 18.2 所示。

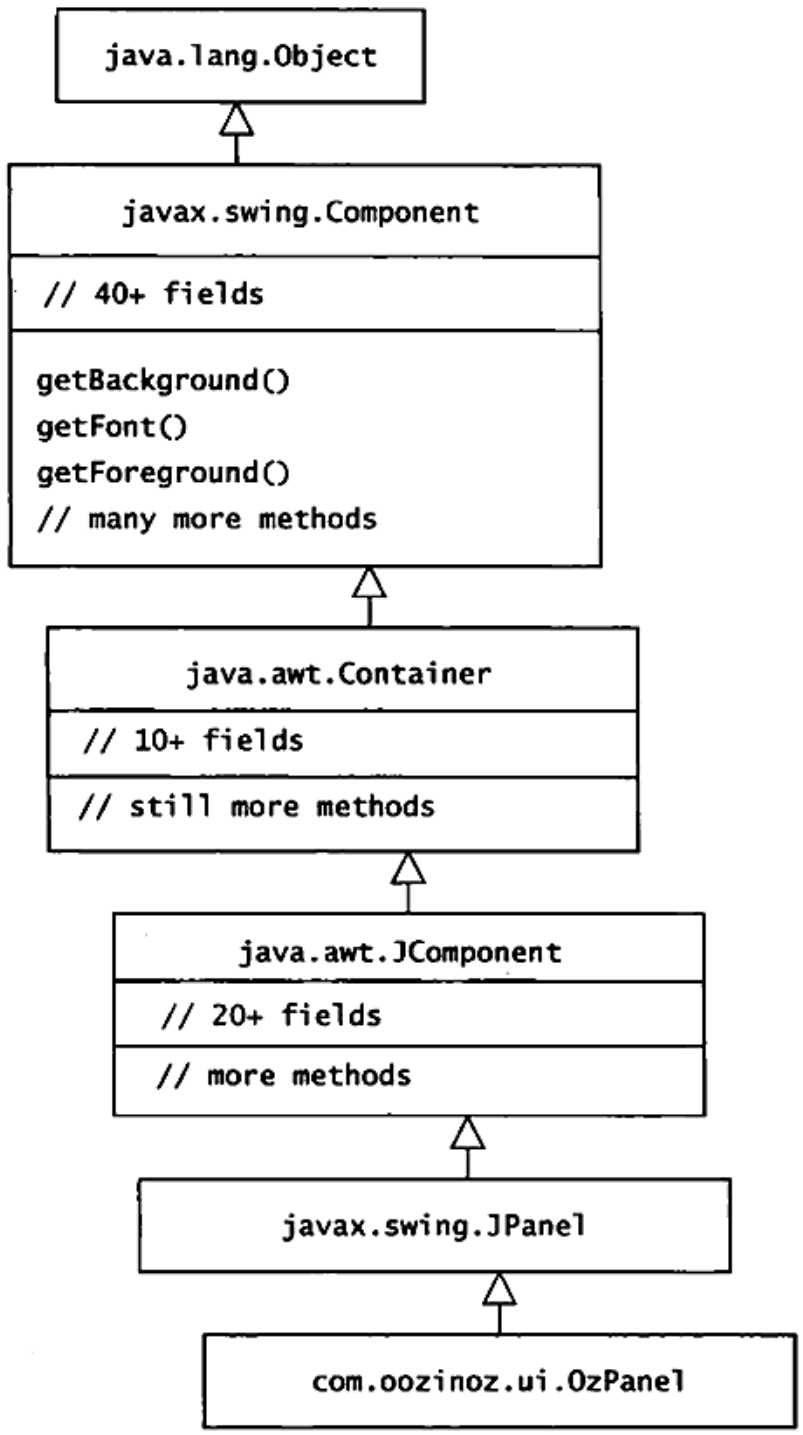

图 18.2　OzPanel 类从超类中继承了大量的属性字段和变量

在图 18.2 中，OzPanel 类从 Component 类继承了大量的属性，这些也是使用 GUI 对象时需要复制的属性。

挑战 18.4

不依赖 clone()方法，写一个 OzPanel.copy2()方法复制一个面板。假定需要复制的重要属性是 background、font 和 foreground。

答案参见第 342 页

小结

原型模式使得用户可以通过复制对象样本来创建新对象。与通过调用构造函数创建对象相比，二者的主要区别在于：通过复制创建的新对象一般会包含原始对象的某些状态。你可以充分利用这一特点，尤其是当多个对象的类在属性上仅存在细微差别，方法上却完全相同的时候。在这种情形下，通过为用户提供一个可以复制的对象原型，就可以在运行时动态创建新类。

当需要创建一个副本时，可以借助 Object.clone()方法。但是必须记住，该方法创建的新对象与对象原型的属性字段是完全相同的。所以，通过这种复制方法获得的新对象或许并不是应用系统开发者所期望的副本，还需要执行更进一步的复制工作。如果对象原型的字段太多，那么我们可以选择先实例化一个新对象，然后将对象原型中需要的字段赋给新对象的对应字段。