

软件体系结构

Zhenyan Ji

— Beijing Jiaotong University —

创建型模式: 原型模式

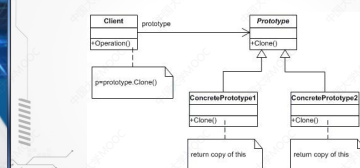
原型模式

» 当创建一个类的实例非常耗时或者非常复杂时, 可使用原型模式: 创建原型实例的副本并适当对其进行修改。

» 定义/意图:

- 定义可通过原型实例创建的对象类型
- 通过复制原型创建新的对象

原型模式



原型模式

» 参与者

- Prototype
声明用于克隆自身的接口
- ConcretePrototype
实现克隆自身的操作

原型模式

- Client
通过让原型克隆自身创建新对象

克隆方法

- 利用clone方法创建任意java对象的副本。
- `Job j1 = (Job)j0.clone();`
- 克隆方法的四个限制
 - 克隆方法总是返回一个Object类型的对象，因此必须将其强制类型转换为正在克隆对象的实际类型。

克隆方法的四个限制

克隆方法

- 2. 该方法为`protected`类型方法，只能在类内部或在包含该类的包内调用。
- 3. 只能克隆实现了`Cloneable`接口的对象。
- 4. 不能被克隆的对象会抛出`CloneNotSupportedException`异常。

- 2. 该方法为`protect`类型方法，只能在类内部或在包含该类的包内调用。
- 3. 只能克隆实现了`Cloneable`接口的对象。
- 4. 不能被克隆的对象会抛出`CloneNotSupportedException`异常。

The screenshot shows a Java Swing window titled "例子". Inside the window, there is a text area containing a "Prototype example" and a list of names. The text area has a "Close" button and a "Refresh" button. The list of names is:

- Kristin Floyd 26.31
- Kimberly Wallace 27.37
- Jason Carey 27.53
- Megan Crapper 27.63
- Karlyn Arment 28.2
- Lucian Rogers 28.88
- Emily McLaughlin 28.8
- Erin Taylor 28.95
- Aurora Lee 28.88
- Kate Isosue 28.91
- Luke Meister 24.88
- Stephen Carme 27.89
- Jeffrey Soltzberg 26.24
- Ernest Vance 26.46
- David Leibowitz 26.78
- Ryan Ronowiczke 28.83
- Emily Renner 28.88
- Aurora Lee 28.88
- Kate Isosue 28.91
- Matthew Conish 28.95
- Christopher Priss 29.02
- Charles Baker 28.96
- Matthew Swetzel 29.1

例子

- ❖ 左边的列表框
 - 程序启动时被加载
 - 显示最初数据
 - 名字按性别排序，性别一致按时间排序
- ❖ 右边的列表框
 - 点击Clone按钮时被加载
 - 显示排序后数据
 - 名字仅按时间排序

- ▶▶ 左边的列表框
 - 程序启动时被加载
 - 显示最初数据
 - 名字按性别排序, 性别一致按时间排序
- ▶▶ 右边的列表框
 - 点击Clone按钮时被加载
 - 显示排序后数据
 - 名字仅按时间排序

例子：原型模式

```
public class SwimData implements Cloneable {  
    public Object clone() {  
        try {  
            return super.clone();  
        } catch (Exception e) {  
            System.out.println(e.getMessage());  
            return null;  
        }  
    }  
}
```

例子：原型模式

» 在方法内进行类型转换

```
public SwimData cloneMe() {  
    try {  
        return (SwimData)super.clone();  
    } catch (Exception e) {  
        System.out.println(e.getMessage());  
        return null;  
    }  
}
```

```
public SwimData cloneMe() {
    try {
        return (SwimData)super.clone();
    } catch (Exception e) {
        System.out.println(e.getMessage());
        return null;
    }
}
```

例子:

```
class Swimmer {  
    String name;  
    int age;  
    String club;  
    float time;  
    boolean female;  
}
```

例子: 原型模式

```
public class SwimData implements  
    Cloneable {  
    Vector swimmers;  
    public SwimData(String filename) {  
        String s = "";  
        swimmers = new Vector();  
        //open data file
```

例子: 原型模式

```
InputFile f = new InputFile(filename)  
s= f.readLine();  
//read in and parse each line  
while(s != null) {  
    swimmers.addElement(new Swimmer(s));  
    s= f.readLine();  
}  
f.close(); }
```

例子

```
swList.removeAll(); //clear list  
for (int i = 0; i < sdata.size(); i++) {  
    sw = sdata.getSwimmer(i);  
    swList.addItem(sw.getName()+"  
        "+sw.getTime());  
}
```

例子

```
sxdata = (SwimData)sdata.clone();  
sxdata.sortByTime(); //re-sort  
cloneList.removeAll(); //clear list  
//now display sorted values from clone  
for(int i=0; i< sxdata.size(); i++) {  
    sw = sxdata.getSwimmer(i);  
    cloneList.addItem(sw.getName()+"  
        "+sw.getTime());  
}
```

浅克隆

Java中的克隆是原始类的浅克隆

Prototype example	Clone
John Miller 24.10	John Miller 24.10
Michael Frost 26.11	Michael Frost 26.11
Andrew Pugh 27.07	Andrew Pugh 27.07
Jason Carter 27.83	Jason Carter 27.83
Robert Coleman 27.85	Robert Coleman 27.85
Stephen Coates 27.89	Stephen Coates 27.89
Robert Bennett 28.2	Robert Bennett 28.2
Jeffrey Daulton 28.24	Jeffrey Daulton 28.24
David Jackson 28.46	David Jackson 28.46
Robert Cooper 28.66	Robert Cooper 28.66
David Larkins 28.70	David Larkins 28.70
David McLaughlin 28.8	David McLaughlin 28.8
Robert Dymkowski 28.83	Robert Dymkowski 28.83
David Carter 28.87	David Carter 28.87
Aurora Lee 28.89	Aurora Lee 28.89
Paul Simpson 28.91	Paul Simpson 28.91
Matthew Cooper 28.95	Matthew Cooper 28.95
Christopher Pugh 28.95	Christopher Pugh 28.95
Matthew Cooper 28.96	Matthew Cooper 28.96
Matthew Daulton 28.1	Matthew Daulton 28.1

深克隆

- » 通过Serializable接口实现数据的深克隆
- » 深克隆允许拷贝具有任意复杂度的类的实例，并且两份拷贝实例间具有完全独立的数据。

深克隆

```
public class SwimData implements Cloneable, Serializable  
class Swimmer implements Serializable  
public Object deepClone() {  
    try {  
        ByteArrayOutputStream b = new ByteArrayOutputStream();  
        ObjectOutputStream out = new ObjectOutputStream(b);  
        out.writeObject(this);  
    }
```

深克隆

```
ByteArrayInputStream bIn = new  
    ByteArrayInputStream(b.toByteArray());  
ObjectInputStream oi = new ObjectInputStream(bIn);  
return (oi.readObject());  
} catch (Exception e) {  
    System.out.println("exception:" + e.getMessage());  
    return null; }  
}
```

创建型模式总结

- » 工厂模式根据提供给工厂的外部数据或环境变量，确定创建哪个类的实例。
- » 抽象工厂模式是工厂的工厂，可以用来创建相关的或互相依赖的一系列产品。

创建型模式总结

- » 建造者模式可以用统一的构建过程构造出不同的复杂对象。
- » 单例模式可以确保只有一个对象实例，并且该实例可被全局访问。
- » 原型模式：当创建新实例成本较高时，复制或克隆已存在类实例而不是创建新的实例。