原型模式 - 杰尔马 66 军团

发表于 2022-09-01 | 更新于 2023-04-06 | 设计模式

| 字数总计: 1.1k | 阅读时长: 3 分钟 | 阅读量: 640 | 评论数: 0

配套视频课程已更新完毕，大家可通过以下两种方式观看视频讲解：

关注公众号： 爱编程的大丙 ，或者进入 大丙课堂 学习。

苏丙榅

合抱之木，生于毫末；九层之台，起于垒土；千里之行，始于足下。

1. 克隆人

在海贼王世界中， 杰尔马王国 拥有一支强大的科学作战部队 – 杰尔马66军团 ， 其先锋是 文斯莫克家族 ， 他们作为雇佣军活跃在世界各地。

这支部队战斗力强悍，没有情感，不畏生死，勇往直前。从某种意义上来讲，他们不能被称之为人，因为他们是科学的结晶，他们都出自 文斯莫克·伽治 之手。

伽治曾和海贼世界中的顶级科学家贝加庞克是同事，一起发现了血统因子，于是才有了现在这么多的杰尔马士兵，对，你没有看错他们都是被克隆出来的。

克隆是一种最直接、最快捷的创建新对象的方式，它不仅隐藏了创建新对象的诸多细节，还保留了源对象的属性信息，保证了这两个对象能够一模一样。

从 血统因子 到 士兵 这是一个复杂而又艰辛的过程，一旦研发成功，之后的事情就是基于母体进行复制（克隆）。我猜伽治不仅是一名科学家，可能还是一位架构师，因为他懂设计模式，这种制作克隆人的模式就是 原型模式 。

> 原型模式就是能够复制已有的对象，而又无需使代码依赖它们所属的类。换种说法，就是通过已有对象克隆出另一个新的对象，并且克隆这个对象不需要使用构造函数。

CMake 保姆级教程（下）
2023-03-15

CMake 保姆级教程（上）
2023-03-06

访问者模式 - 再见，香波地群岛
2022-09-22

模板方法模式 - 和平主义者
2022-09-21

状态模式 - 文斯莫克・山治
2022-09-20

2. 这是在脱了裤子放屁吗

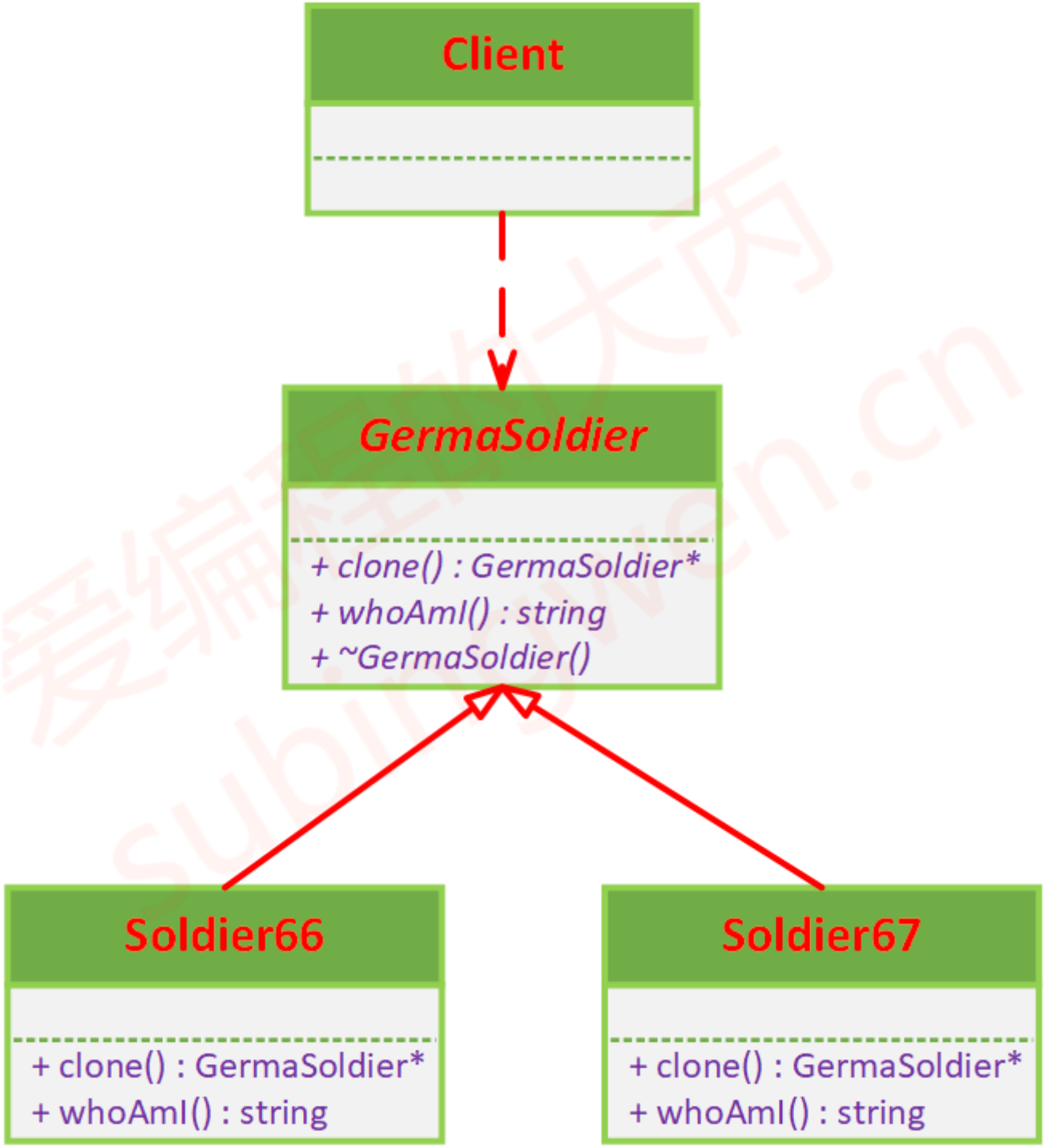
懂 C++ 的亲们看了上面关于原型模式的描述肯定是满脑子问号。因为在 C++ 中只要定义一个类，这个类就默认自带六大函数，其中一个就是 拷贝构造函数 ，这个函数的作用就是通过一个已有对象克隆出一个新的对象。一个拷贝构造函数就能搞定的事情为啥还要搞出一种设计模式呢？

这是脱了裤子放屁吗？肯定不是，因为这里边还隐藏着一个细节。

- 对于伽治来说，他可能想通过一代士兵克隆出更优秀的二代士兵
- 对于程序猿来说，我们可能想要通父类指针或引用把指向的子类对象克隆出来

通过这个描述，就可以从里面挖掘出一个重要的信息： 克隆可能会在父类和子类之间进行，并且可能是动态的，很明显通过父类的拷贝构造函数无法实现对子类对象的拷贝 ，其实这就是一个多态， 我们需要给父类提供一个克隆函数并且是一个虚函数 。

现在逻辑关系已经说明白了，来看一下对应 UML 类图


Client
GermaSoldier
+ clone() : GermaSoldier*
+ whoAmI() : string
+ ~GermaSoldier()
Soldier66
+ clone() : GermaSoldier*
+ whoAmI() : string
Soldier67
+ clone() : GermaSoldier*
+ whoAmI() : string

3. 量产士兵

根据上面的 UML 类图，我们就可以把对应的代码写出了，示例代码如下：

C++

```cpp
#include <iostream>
using namespace std;

class GermaSoldier
{
public:
    virtual GermaSoldier* clone() = 0;
    virtual string whoAmI() = 0;
    virtual ~GermaSoldier() {}
};

class Soldier66 : public GermaSoldier
{
public:
    GermaSoldier* clone() override
    {
        return new Soldier66(*this);
    }
    string whoAmI() override
    {
        return string("我是杰尔马66的超级士兵!!!");
    }
};

```

```
25  class Soldier67 : public GermaSoldier
26  {
27  public:
28      GermaSoldier* clone()
29      {
30          return new Soldier67(*this);
31      }
32      string whoAmI() override
33      {
34          return string("我是杰尔马67的超级士兵!!!");
35      }
36  };
37
38  int main()
39  {
40      GermaSoldier* obj = new Soldier66;
41      GermaSoldier* soldier = obj→clone();
42      cout << soldier→whoAmI() << endl;
43      delete soldier;
```

代码中的 `main()` 函数对应的就是 UML 类图中的客户端角色。

- 第41行 通过父类指针克隆了子类 `Soldier66` 的对象
- 第47行 通过父类指针克隆了子类 `Soldier67` 的对象
- 在这两个士兵子类的 `clone()` 函数体内部是通过当前子类的拷贝构造函数复制出了一个新的子类对象。

程序执行的结果如下：

C++

```
我是杰尔马66的超级士兵!!!
我是杰尔马67的超级士兵!!!
```

通过输出的结果可以看到通过父类指针克隆子类的对象成功了。

文章作者: 苏丙榅

设计模式

打赏

相关推荐

评论

昵称 邮箱 网址(http://)

来都来了, 说点什么吧...

提交

来发评论吧 ~

Powered By Valine
v1.5.1