

工厂模式 - 人造恶魔果实工厂 2

📅 发表于 2022-08-30 | ⌚ 更新于 2023-04-06 | 📄 设计模式

| 📄 字数总计: 1.3k | ⌚ 阅读时长: 5 分钟 | 👁 阅读量: 1403 | 💬 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。



苏丙楦

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. 简单工厂模式的弊端

在上一节 [简单工厂模式](#) 中，创建了一个工厂类，用于生产需要的对象，但是这种方式有一个弊端，它违反了设计模式中的 [开放 - 封闭原则](#)，先来看相关的代码：

```
1 // 恶魔果实工厂类
2 enum class Type:char{SHEEP, LION, BAT};
3 class SmileFactory
4 {
5 public:
6     SmileFactory() {}
7     ~SmileFactory() {}
8     AbstractSmile* createSmile(Type type)
9     {
10         AbstractSmile* ptr = nullptr;
11         switch (type)
12         {
13             case Type::SHEEP:
14                 ptr = new SheepSmile;
15                 break;
16             case Type::LION:
17                 ptr = new LionSmile;
18                 break;
19             case Type::BAT:
20                 ptr = new BatSmile;
21                 break;
```

文章	标签	分类
134	37	12

大丙课堂



公告

微信公众号 爱编程的大丙 和
大丙课堂 上线了，可
点击上方 图标关注 ~ ~ ~

目录

1. 简单工厂模式的弊端
2. 工厂模式

最新文章

```
22         default:
23             break;
24     }
25     return ptr;
26 }
27 };
```

在上面的工厂函数中需要生成三种人造恶魔果实，现在如果想要生成更多，那么就需要在工厂函数的 `switch` 语句中添加更多的 `case`，很明显这违背了 **封闭** 原则，也就意味着需要基于 **开放** 原则来解决这个问题。

使用工厂模式可以很完美的解决上述的问题，简单工厂模式是只有一个工厂类，而工厂模式是有很多的工厂类：

- 一个基类，包含一个虚工厂函数，用于实现多态。
- 多个子类，重写父类的工厂函数。每个子工厂类负责生产一种恶魔果实，这相当于再次解耦，将工厂类的职责再次拆分、细化，如果要生产新品种的恶魔果实，那么只需要添加对应的工厂类，无需修改原有的代码。

2. 工厂模式

我们先修改一下简单工厂模式中工厂类相关的代码：



C++



```
1 // 恶魔果实工厂类
2 class AbstractFactory
```



CMake 保姆级教程
(下)

2023-03-15



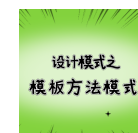
CMake 保姆级教程
(上)

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

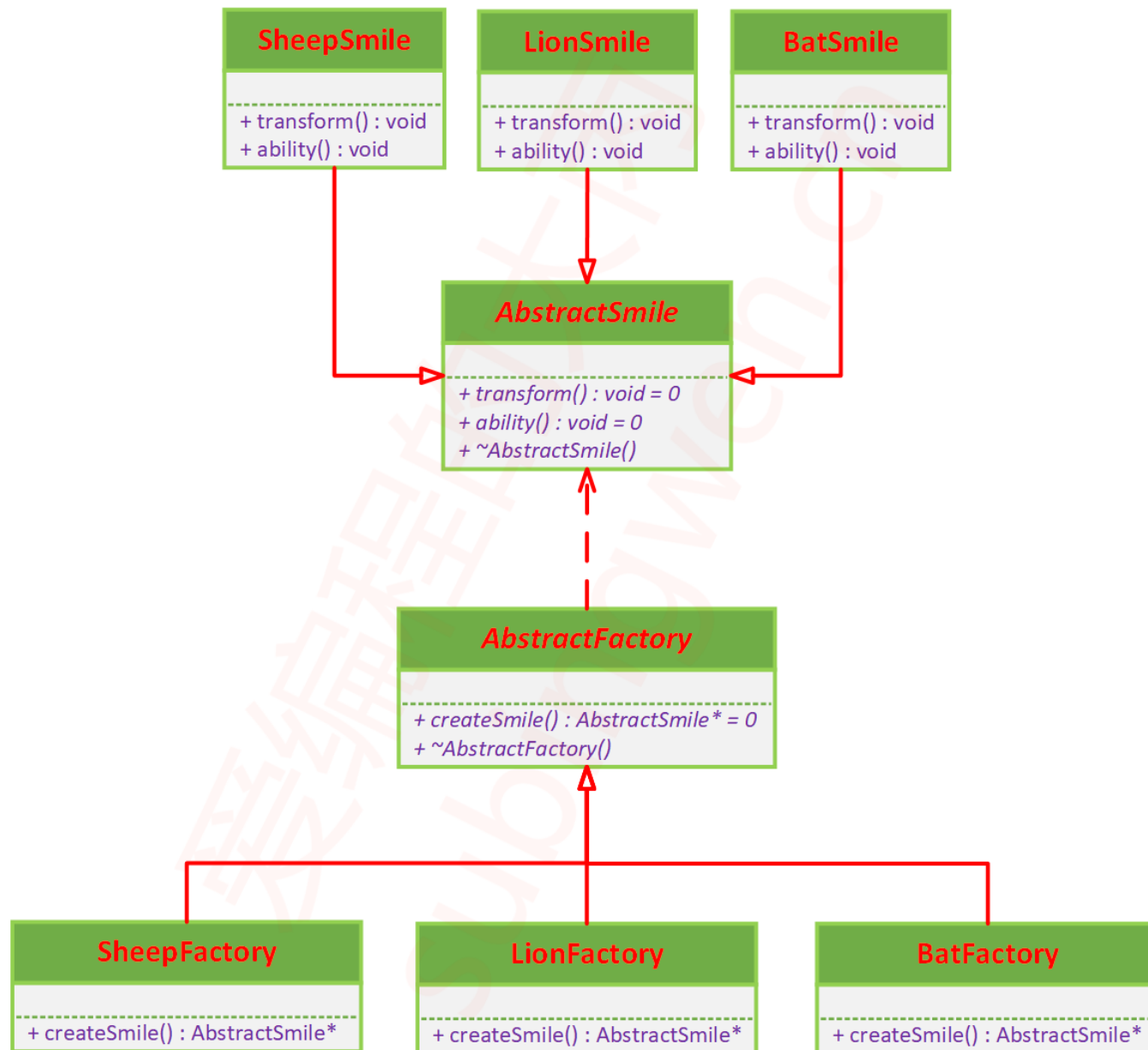
```
3  {
4  public:
5      virtual AbstractSmile* createSmile() = 0;
6      virtual ~AbstractFactory() {}
7  };
8
9  class SheepFactory : public AbstractFactory
10 {
11 public:
12     AbstractSmile* createSmile() override
13     {
14         return new SheepSmile;
15     }
16     ~SheepFactory()
17     {
18         cout << "释放 SheepFactory 类相关的内存资源" << endl;
19     }
20 };
21
22 class LionFactory : public AbstractFactory
23 {
24 public:
25     AbstractSmile* createSmile() override
26     {
27         return new LionSmile;
28     }
29     ~LionFactory()
30     {
31         cout << "释放 LionFactory 类相关的内存资源" << endl;
32     }
33 }
```

```
34 };
35
36 class BatFactory : public AbstractFactory
37 {
38 public:
39     AbstractSmile* createSmile() override
40     {
41         return new BatSmile;
42     }
```



通过示例代码可以看到，每个工厂类其实都不复杂，在每个子工厂类中也只是重写了父类的工厂方法而已，每个子工厂类生产一种恶魔果实，但是工厂函数的返回值确是恶魔果实类的基类类型，相当于是使用父类指针指向了子类对象，此处也是用到了多态。通过这样的处理，工厂函数也就不再需要参数了。

根据简单工厂模式的代码和上面的修改就可以把工厂模式的 UML 类图画出来了：



完整的代码应该是这样的:



C++



```
1  #include <iostream>
2  using namespace std;
3
4  class AbstractSmile
5  {
6  public:
7      virtual void transform() = 0;
8      virtual void ability() = 0;
9      virtual ~AbstractSmile() {}
10 };
11 // 人造恶魔果实·绵羊形态
12 class SheepSmile : public AbstractSmile
13 {
14 public:
15     void transform() override
16     {
17         cout << "变成人兽 -- 山羊人形态..." << endl;
18     }
19     void ability() override
20     {
21         cout << "将手臂变成绵羊角的招式 -- 巨羊角" << endl;
22     }
23 };
24
25 // 人造恶魔果实·狮子形态
26 class LionSmile : public AbstractSmile
27 {
28 public:
29     void transform() override
30     {
31         cout << "变成人兽 -- 狮子人形态..." << endl;
```

```
32     }
33     void ability() override
34     {
35         cout << "火遁· 豪火球之术..." << endl;
36     }
37 };
38
39 class BatSmile : public AbstractSmile
40 {
41 public:
42     void transform() override
43     {
```



在 `main()` 函数中的这句代码是实例化了一个生成蝙蝠恶魔果实的工厂对象:



C++



```
1 AbstractFactory* factory = new BatFactory;
```

在真实的项目场景中，要生成什么类型的恶魔果实其实是通过客户端的操作界面控制的，它对应的可能是一个按钮或者是一个选择列表，用户做出了选择，程序就可以根据该需求去创建对应的工厂对象，最终将选择的恶魔果实生产出来。



在上面的例子中，不论是恶魔果实的基类，还是工厂类的基类，它们的虚函数可以是纯虚函数，也可以是非纯虚函数。这样的基类在设计模式中就可以称之为抽象类（此处的抽象类和 C++ 中对抽象类的定义有一点出入）。

文章作者: 苏丙楦



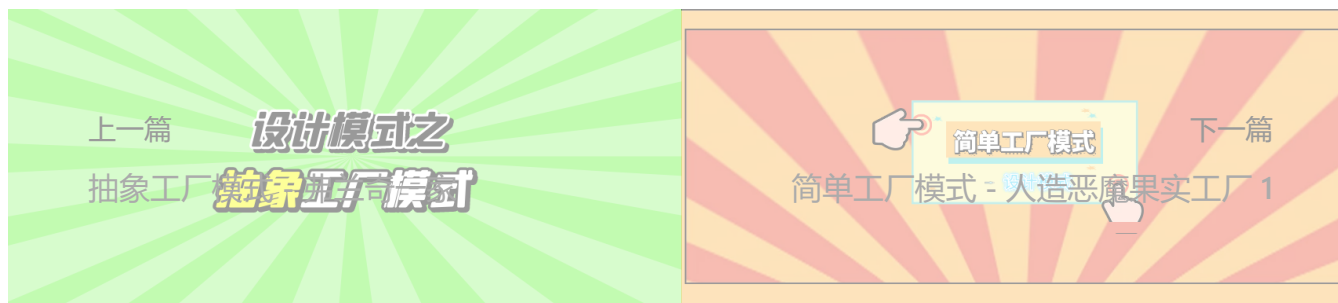
文章链接: <https://subingwen.cn/design-patterns/factory/>

版权声明: 本博客所有文章除特别声明外，均采用 [CC BY-NC-SA 4.0](#) 许可协议。转载请注明来自 爱编程的大丙！

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

©2021 - 2023 By 苏丙楹

冀 ICP 备 2021000342 号 - 1



冀公网安备 13019902000353 号