简单工厂模式 - 人造恶魔果实工厂 1

发表于 2022-08-29 | 更新于 2023-04-06 | 设计模式

| 字数总计: 2.2k | 阅读时长: 7 分钟 | 阅读量: 2126 | 评论数: 3

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. 工厂模式的特点

在海贼王中，作为原王下七武海之一的多弗朗明哥，可以说是新世界最大的流氓头子，拥有无上的权利和无尽的财富。他既是德雷斯罗萨国王又是地下世界的中介，控制着世界各地的诸多产业，人造恶魔果实工厂就是其中之一。

人造恶魔果实的最大买家是四皇之一的 凯多 ， 凯多 其实很明智，他没有自己去生产，可能有这么几个因素：

1. 凯多手下没有像 凯撒·库朗 一样的科学家，无法掌握生产人造恶魔果实这种顶级的科学技术

【意味着构造一个对象有时候需要经历一个非常复杂的操作流程，既然麻烦那索性就不干

文章 134 标签 37 分类 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. 工厂模式的特点
2. 生产的产品
3. 如何生产
最新文章

了】。

2. 有需求下单就行，只需关心结果，无需关心过程【实现了解耦合】。
3. 人造恶魔果实出了问题，自己无责任，售后直接找明哥【便于维护】。

在我们现实生活中也是一样，买馒头和自己蒸馒头、去饭店点一份大盘鸡和自己养鸡，杀鸡，做大盘鸡，这是全然不同的两种体验：

- 自己做麻烦，而且有失败的风险，需要自己承担后果。
- 买现成的，可以忽略制作细节，方便快捷并且无风险，得到的肯定是美味的食物。

对于后者，就相当于是一个加工厂，通过这个工厂我们就可以得到想要的东西，在程序设计中，这种模式就叫做工厂模式，工厂生成出的产品就是某个类的实例，也就是对象。关于工厂模式一共有三种，分别是：简单工厂模式、工厂模式、抽象工厂模式。

通过上面人造恶魔果实的例子，我们能够了解到，不论使用哪种工厂模式其主要目的都是 实现类与类之间的解耦合，这样我们在创建对象的时候就变成了拿来主义，使程序更加便于维护。 在本节中，先介绍简单工厂模式。

基于简单工厂模式去创建对象的时候，需要提供一个工厂类，专门用于生产需要的对象，这样关于对象的创建操作就被剥离出去了。

简单工厂模式相关类的创建和使用步骤如下：

1. 创建一个新的类，可以将这个类称之为工厂类。对于简单工厂模式来说，需要的工厂类只有一个。

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. 在这个工厂类中添加一个公共的成员函数，通过这个函数来创建我们需要的对象，关于这个函数一般将其称之为工厂函数。
3. 关于使用，首先创建一个工厂类对象，然后通过这个对象调用工厂函数，这样就可以生产出一个指定类型的实例对象了。

2. 生产的产品

在海贼世界中， 凯撒·库朗 研制的人造恶魔果实是有瑕疵的，吃下人造恶魔果实的失败品没能成功获得果实能力的人，会被剥夺除笑以外的一切表情，所以人造恶魔果实被称为 `SMILE` 。

下面是明哥的 SMILE 工厂要生产的众多人造动物系恶魔果实中的三种:

C++

```cpp
// 人造恶魔果实· 绵羊形态
class SheepSmile
{
public:
    void transform()
    {
        cout << "变成人兽 -- 山羊人形态..." << endl;
    }
    void ability()
    {
        cout << "将手臂变成绵羊角的招式 -- 巨羊角" << endl;
    }
};

// 人造恶魔果实· 狮子形态
class LionSmile
{
public:
    void transform()
    {
        cout << "变成人兽 -- 狮子人形态..." << endl;
    }
    void ability()
    {
        cout << "火遁· 豪火球之术..." << endl;
    }
};

// 人造恶魔果实· 蝙蝠形态
class BatSmile
```

```
{
public:
    void transform()
    {
        cout << "变成人兽 -- 蝙蝠人形态..." << endl;
    }
    void ability()
    {
        cout << "声纳引箭之万剑归宗..." << endl;
    }
};
```

不论是吃了那种恶魔果实，获得了相应的能力之后，可以做的事情大体是相同的，那就是形态变化 `transform()` 和 使用果实能力 `alility()`。

另外，生产这些恶魔果实的时候可能需要极其复杂的参数，在此就省略了【也就是说这些类的构造函数的参数在此被省略了】。

3. 如何生产

如果想要生产出这些恶魔果实，可以先创建一个工厂类，然后再给这个工厂类添加一个工厂函数，又因为我们要生成三种不同类型的恶魔果实，所以可以给工厂函数添加一个参数，用以控制当前要生产的是哪一类。

C++

```
enum class Type:char{SHEEP, LION, BAT};
// 恶魔果实工厂类
```

```
class SmileFactory
{
public:
    enum class Type:char{SHEEP, LION, BAT};
    SmileFactory() {}
    ~SmileFactory() {}
    void* createSmile(Type type)
    {
        void* ptr = nullptr;
        switch (type)
        {
        case Type::SHEEP:
            ptr = new SheepSmile;
            break;
        case Type::LION:
            ptr = new LionSmile;
            break;
        case Type::BAT:
            ptr = new BatSmile;
            break;
        default:
            break;
        }
        return ptr;
    }
};

int main()
{
    SmileFactory* factory = new SmileFactory;
    BatSmile* batObj = (BatSmile*)factory->createSmile(Type::BAT);
```

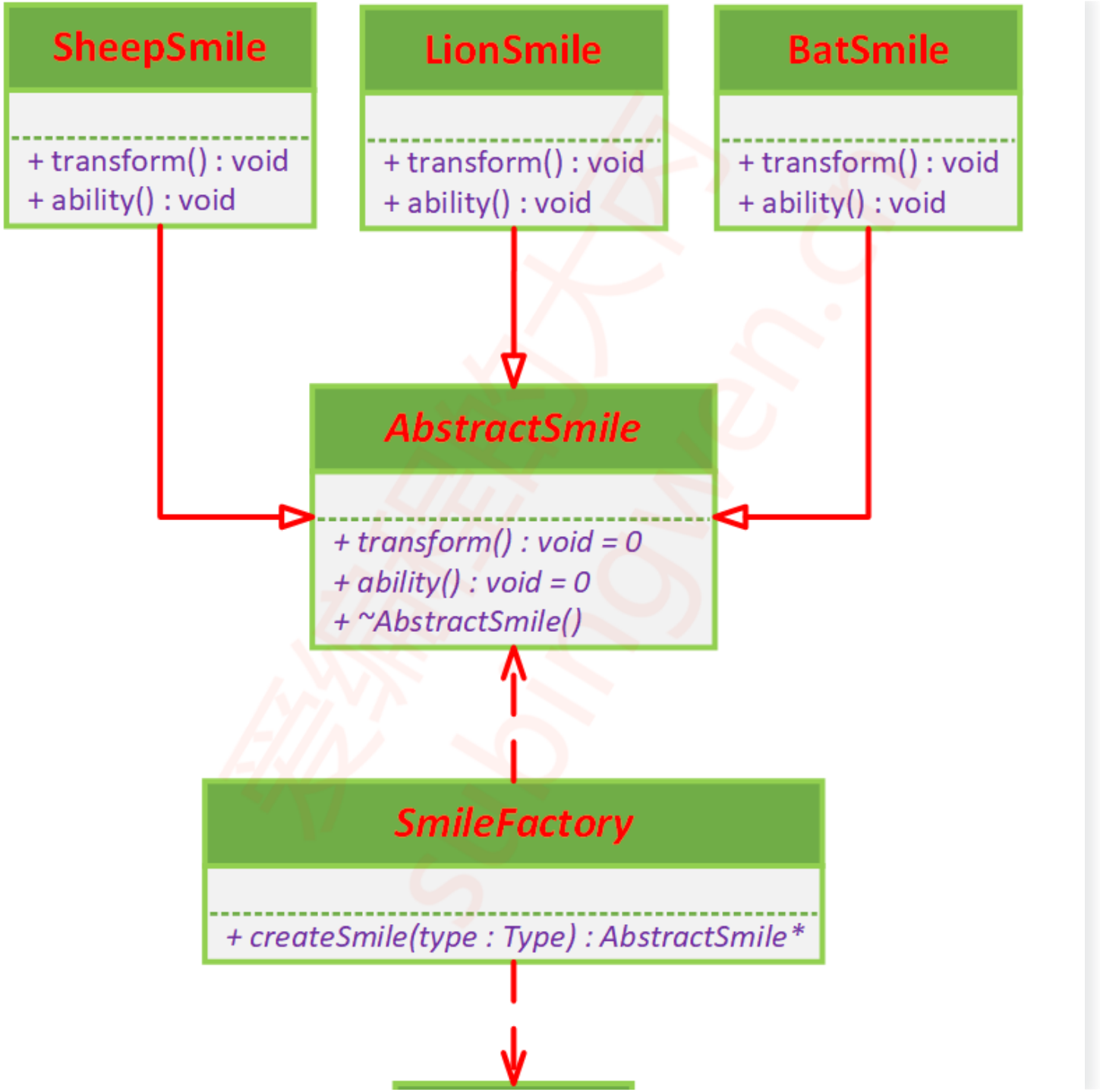
```
34      return 0;
35  }
```

- 关于恶魔果实的类型，上面的类中用到了 强类型枚举（C++11新特性），增强了代码的可读性，并且将枚举元素设置为了 `char` 类型，节省了内存。
- 函数 `createSmile(Type type)` 的返回值是 `void*` 类型，这样处理主要是因为每个 `case` 语句 创建出的对象类型是不一样的，为了实现兼容，故此这样处理。
- 得到函数 `createSmile(Type type)` 的返回值之后，还需要将其转换成实际的类型，处理起来还是比较繁琐的。

关于工厂函数的返回值，在 C++ 中还有一种更好的解决方案，就是使用 多态 。如果想要实现多态，需要满足三个条件：

- 类和类之间有继承关系。
- 父类中有虚函数，并且在子类中需要重写这些虚函数。
- 使用父类指针或引用指向子类对象。

所以，我们需要给人造恶魔果实提供一个基类，然后让上边的三个类 `SheepSmile` 、 `LionSmile` 、 `BatSmile` 作为子类继承这个基类。根据分析我们就有画出简单工厂模式的 UML 类图了：


SheepSmile
+ transform() : void
+ ability() : void
LionSmile
+ transform() : void
+ ability() : void
BatSmile
+ transform() : void
+ ability() : void
AbstractSmile
+ transform() : void = 0
+ ability() : void = 0
+ ~AbstractSmile()
SmileFactory
+ createSmile(type : Type) : AbstractSmile*

根据 UML 类图，编写出的代码如下：

C++

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

class AbstractSmile
{
public:
    virtual void transform() {}
    virtual void ability() {}
    virtual ~AbstractSmile() {}
};
// 人造恶魔果实· 绵羊形态
class SheepSmile : public AbstractSmile
{
public:
    void transform() override
    {
        cout << "变成人兽 -- 山羊人形态..." << endl;
    }
    void ability() override
    {
```

```
        cout << "将手臂变成绵羊角的招式 -- 巨羊角" << endl;
    }
};

// 人造恶魔果实· 狮子形态
class LionSmile : public AbstractSmile
{
public:
    void transform() override
    {
        cout << "变成人兽 -- 狮子人形态..." << endl;
    }
    void ability() override
    {
        cout << "火遁· 豪火球之术..." << endl;
    }
};

class BatSmile : public AbstractSmile
{
public:
    void transform() override
```

通过上面的代码，我们实现了一个简单工厂模式，关于里边的细节有以下几点需要说明：

1. 由于人造恶魔果实类有继承关系， 并且实现了多态，所以父类的析构函数也应该是虚函数，这样才能够通过父类指针或引用析构子类的对象。

2. 工厂函数 `createSmile(Type type)` 的返回值修改成了 `AbstractSmile*` 类型，这是人造恶魔果实类的基类，通过这个指针保存的是子类对象的地址，这样就实现了多态，所以在 `main()` 函数中，通过 `obj` 对象调用的实际是子类 `BatSmile` 中的函数，因此打印出的信息应该是这样的：

C++

```
变成人兽 -- 蝙蝠人形态...
声纳引箭之万剑归宗...
```

爱编程的大丙 subingwen.cn

文章作者: 苏丙榅

文章链接: https://subingwen.cn/design-patterns/simple-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

上一篇
工厂模式 - 人造恶魔果实工厂 2

下一篇
设计模式三原则

相关推荐

评论

昵称 邮箱 网址(http://)

来都来了, 说点什么吧...

提交

3 评论

Anonymous Chrome 113.0.0.0 macOS 10.15.7

2023-05-26

回复

不得了，不得了

Anonymous Chrome 109.0.0.0 Windows 11

2023-01-31

回复

1

Anonymous Chrome 109.0.0.0 Windows 11

2023-01-15

回复

类名中的 Smile 是什么意思？shape?

Anonymous Chrome 109.0.0.0 Windows 11

2023-01-15

回复

@Anonymous , 看到了，“所以人造恶魔果实被称为 SMILE”

Anonymous Chrome 112.0.0.0 Windows 11

2023-05-27

回复

OK

Powered By Valine
v1.5.1

©2021 - 2023 By 苏丙榅
冀 ICP 备 2021000342 号 - 1
冀公网安备 13019902000353 号