桥接模式 - 大海贼时代

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

| 字数总计: 3.1k | 阅读时长: 10 分钟 | 阅读量: 544 | 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. 组建海贼团

哥尔·D·罗杰 是罗杰海贼团船长。他最终征服了伟大航路，完成了伟大航路的航行，被人们成为 海贼王 。后来得了绝症，得知自己命不久矣，主动自首并在东海罗格镇被处刑。临死前罗杰的一句话“想要我的宝藏吗？想要的话就全都给你！”

很多人为了梦想，为了罗杰留下的宝藏竞相出海，大海贼时代就此开启。

对于罗杰的行为，最不高兴的肯定是世界政府和海军了，世界政府是海贼中的最高权利机构，海军是世界政府的直属组织，他们以绝对的正义为名在全世界海洋执行维持治安工作。

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

最新文章

现在海军的最高统帅也就是海军元帅是 赤犬 ，假设现在赤犬想要将现在已知的所有的海贼团全部记录在册，那么肯定得写个程序，关于如何能够高效的记录和管理这些信息，隶属海军的程序猿们展开了讨论：

如果只记录海贼船的名字肯定的不完整的，这个海贼团内部还有一个完整的组织结构，每个海贼团有若干船员，每个船员有不同的分工，也就是他们的职责不同。

比如草帽海贼团目前一共十个人，分别是： 船长 、 剑士 、 厨师 、 航海士 、 考古学家 、 船医 、 音乐家 、 船工 、 狙击手 、 舵手 。

方案 1

CMake 保姆级教程（下）
2023-03-15

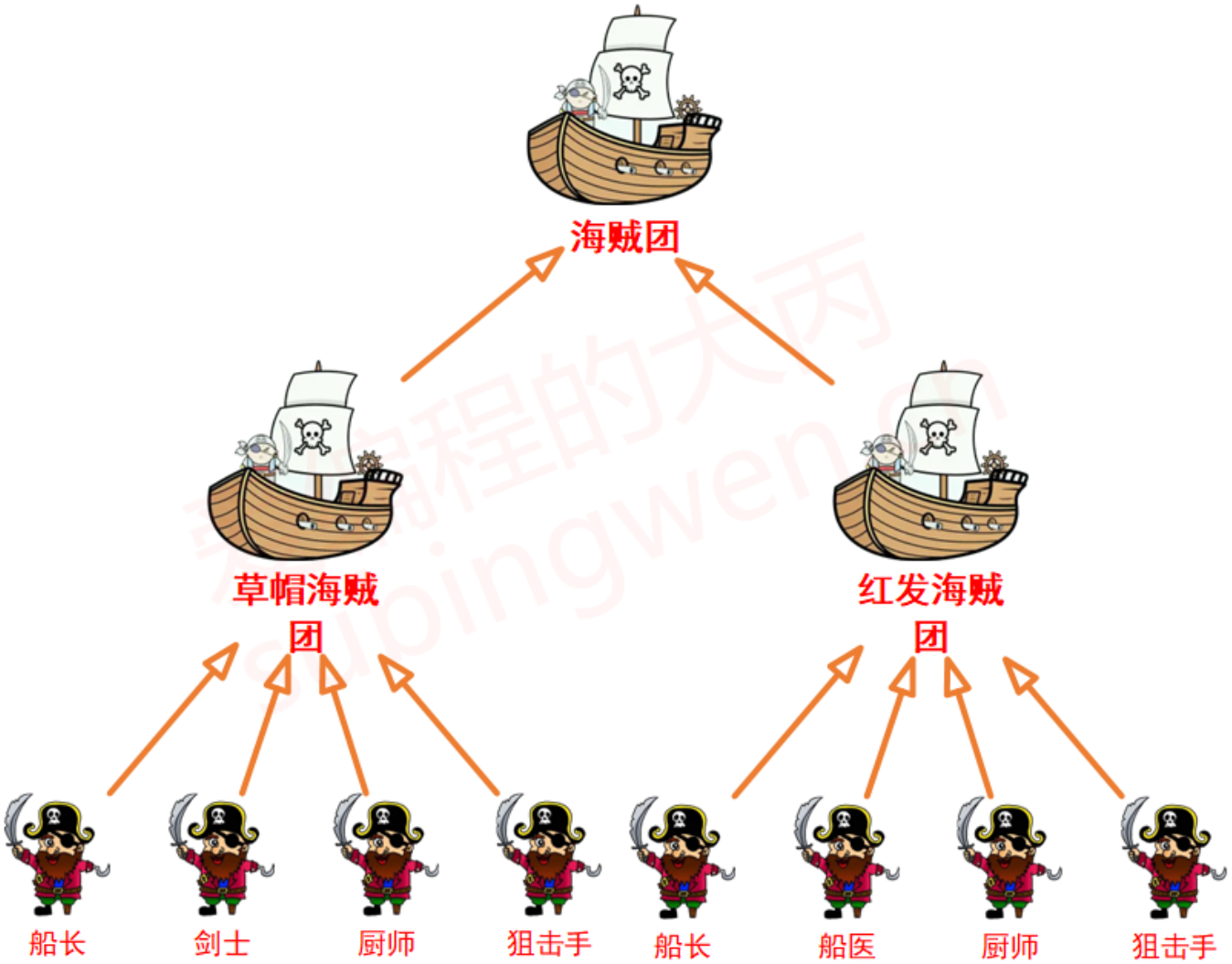
CMake 保姆级教程（上）
2023-03-06

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

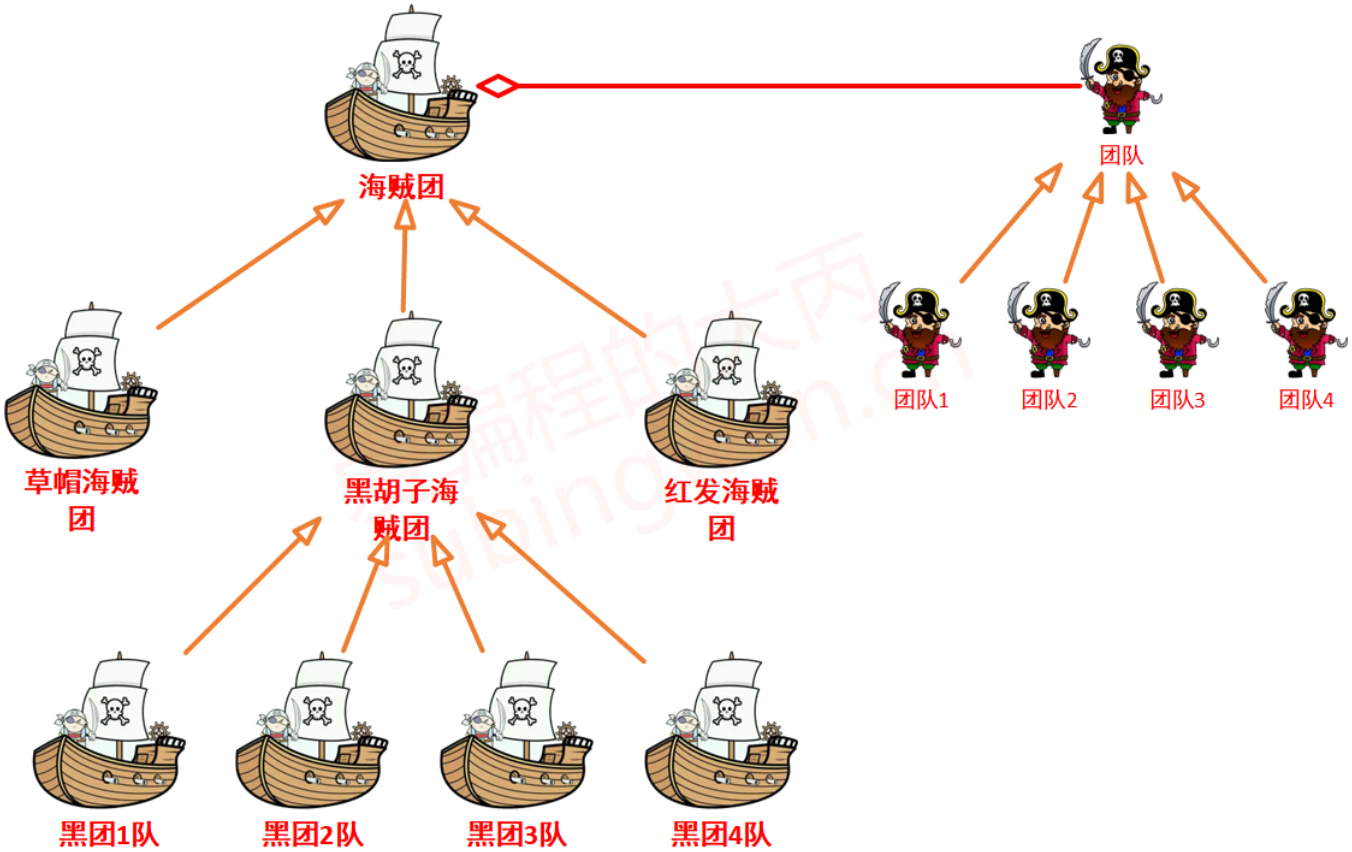
状态模式 - 文斯莫克・山治
2022-09-20

因为每个船员都具有所属海贼团的一些特性，所以可以让每个船员都从对应的海贼团类派生，如下图，它描述的某一个海贼团中类和类之间的关系：



方案 2

将海贼团的船员和海贼团之间的继承关系，改为聚合关系，如下图：



盖棺定论

上面的两种方案你觉得哪个更合理一些呢？ 很明显，是第二种。

- 第一种解决方案

1. 每个船员都是当前海贼团的子类，这样船员就继承了海贼团的属性，合情合理。

2. 如果当前海贼团添加了一个成员就需要给当前海贼团类添加一个子类。拿路飞举个例子，他在德雷斯罗萨王国打败多弗朗明哥之后，组成了草帽大船团，小弟一下子扩充了 5640 人，难道要给草帽团添加五千多个子类吗？如果这样处理，海贼船和船员的耦合度就比较高了。

- 第二种解决方案

1. 海贼团之间是继承关系，但是 此时的海贼团也只是一个抽象 ，因为 组成海贼团的人已经被抽离了 ，船员已经和所属的海贼团没有了继承关系。
2. 关于海贼世界的船员在船上对应不同的职责担任不同的职务，他们是一个团队，所以可以给船员抽象出一个团队类，用于管理船上的成员。
3. 抽象的海贼团只有一个空壳子，所以要赋予其灵魂也就是给它添加船员，此时的海贼团和船员团队可以通过聚合的方式组合成为一个整体。
4. 这种解决方案不仅适用于管理海贼团，用于管理海军的各个舰队也是没有问题的。

通过上述分析，肯定是使用第二种解决方案程序猿的工作量会更少一些，且更容易维护和扩展。第二种方式的原则就是**将抽象部分和它的实现部分分离，使它们可以独立的变化，这种处理模式就是桥接模式。**

关于桥接模式的使用对应的应用场景也有很多，比如：

1. 空调、电视机等和它们对应的遥控器
   - 空调、电视机是抽象，遥控器是实现
2. 手机品牌和手机软件
   - 手机品牌是抽象，手机软件是实现

3. 跨平台的 GUI 在不同平台上运行
    - 程序的 GUI 层是抽象，操作系统的 API 是实现

2. 路飞出海

年幼的路飞误食了红发香克斯的橡胶果实（人人果实·幻兽种·尼卡形态），受到香克斯的影响决定出海，他要做的第一件事儿就是找一艘船并找一些伙伴一起去冒险，下面我们通过桥接模式来记录下草帽团的信息。

2.1 伙伴

不论是哪艘船上的船员肯定都是有一些个人的身份信息，为了将这些信息记录下来，先定一个存储数据的结构体：

C++

```cpp
// 人员信息
struct Person
{
    Person(string name, string job, string ability, string reward, string b
    {
        this->name = name;
        this->job = job;
        this->ability = ability;
        this->reward = reward;
        this->beiZhu = biezhu;
    }
```

```
    ~Person()
    {
        cout << name << "被析构..." << endl;
    }
    string name;    // 名字
    string job;     // 职责
    string ability; // 能力
    string reward;  // 赏金
    string beiZhu;  // 备注
};
```

在上面已经提到了，关于团队的成员组成可以是海贼，也可以是海军，所以可以先定义一个团队的抽象类：

C++

```
// 抽象船员
class AbstractTeam
{
public:
    AbstractTeam(string name) : m_teamName(name){}
    string getTeamName()
    {
        return m_teamName;
    }
    void addMember(Person* p)
    {
        m_infoMap.insert(make_pair(p->name, p));
    }
    void show()
    {
```

```
        cout << m_teamName << ": " << endl;
        for (const auto& item : m_infoMap)
        {
            cout << "【Name: " << item.second→name
                << ", Job: " << item.second→job
                << ", Ability: " << item.second→ability
                << ", MoneyReward: " << item.second→reward
                << ", BeiZhu: " << item.second→beiZhu
                << "】" << endl;
        }
    }
    virtual void executeTask() = 0;    // 执行任务
    virtual ~AbstractTeam()
    {
        for (const auto& item : m_infoMap)
        {
            delete item.second;
        }
    }

protected:
    string m_teamName = string();
    map<string, Person*> m_infoMap;
};
```

在上面的抽象类中可以添加 `addMember(Person* p)` 和显示 `show()` 当前团队成员的信息。不同的团队他们的目标是不一样的，所以需要在子类中重写这个纯虚函数 `executeTask()`。

当路飞一行人到达东海罗格镇的时候，就开始被当时还是海军大佐的斯摩格追捕（现在是中将），接下来基于上面的基类将路飞和斯摩格团队对应的类定义出来：

路飞

C++

```cpp
class CaoMaoTeam : public AbstractTeam
{
public:
    using AbstractTeam::AbstractTeam;
    void executeTask() override
    {
        cout << "在海上冒险，找到 ONE PIECE 成为海贼王！" << endl;
    }
};
```

在子类 `CaoMaoTeam` 中必须重写父类的纯虚函数 `executeTask()`，这样才能创建这个子类的实例对象。

斯摩格

C++

```cpp
class SmokerTeam : public AbstractTeam
{
public:
    using AbstractTeam::AbstractTeam;
    void executeTask() override
    {
        cout << "为了正义，先将草帽一伙一网打尽!!!" << endl;
    }
};
```

在子类 `SmokerTeam` 中必须重写父类的纯虚函数 `executeTask()`，这样才能创建这个子类的实例对象，斯摩格和路飞的立场不同，所以他们通过这个函数干的事情是不一样的。

2.2 海上交通工具

不论是海军还是海贼在大海上航行都需要船，虽然他们驾驶的船只不同，但是有很多属性还是一致的，所以我们可以先定义一个船的抽象类：

C++

```cpp
// 船的抽象类
class AbstractShip
{
public:
    AbstractShip(AbstractTeam* team) : m_team(team) {}
    void showTeam()
    {
        m_team->show();
        m_team->executeTask();
    }
    virtual string getName() = 0;
    virtual void feature() = 0;
    virtual ~AbstractShip() {}
protected:
    AbstractTeam* m_team = nullptr;
};
```

在这个抽象类中提供了一个纯虚函数用来描述船的特点 `feature()`，在不同的子类中都需要重写这个虚函数。

> 对于一个海贼团或者一支海军部队来说，光有船是不完整的，船只是这个团队的抽象，如果想要让它鲜活起来就必要要有由人组成的团队，也就是抽象的具体实现。所以，在这个抽象类中包含了一个团队对象，船和团队二者之间的关系可以看做是聚合关系。

梅利号

路飞来到东海的西罗布村，得到梅利号，这是草帽海贼团的第一艘船，下面把梅利号对应的类定义出来：

C++

```cpp
class Merry : public AbstractShip
{
public:
    using AbstractShip::AbstractShip;
    string getName() override
    {
        return string("前进·梅利号");
    }
    void feature() override
    {
        cout << getName()
             << " -- 船首为羊头，在司法岛化身船精灵舍己救下了草帽一伙！" << endl;
    }
};
```

海军无敌战舰

C++

```cpp
class HaiJunShip : public AbstractShip
{
public:
    using AbstractShip::AbstractShip;
    string getName() override
    {
        return string("无敌海军号");
    }
    void feature() override
    {
        cout << getName() << " -- 船底由海楼石建造，可以穿过无风带的巨大炮舰!" <<
    }
};
```

2.3 你追我跑

在上面的讲解中，我们已经把要出海冒险的抽象部分（海贼船/海军军舰）和实现部分（海贼团团队/海军部队）分别定义出来了，最后需要将它们合二为一，使他们成为一个有灵魂的整体。

C++

```cpp
int main()
{
    // 草帽海贼团
    CaoMaoTeam* caomao = new CaoMaoTeam("草帽海贼团");
    Person* luffy = new Person("路飞", "船长", "橡胶果实能力者", "30亿贝里", "爱
    Person* zoro = new Person("索隆", "剑士", "三刀流", "11亿1100万贝里", "路痴
    Person* sanji = new Person("山治", "厨师", "隐形黑", "10亿3200万贝里", "好
    Person* nami = new Person("娜美", "航海士", "天候棒+宙斯", "3亿6600万贝里",
```

```
    caomao->addMember(luffy);
    caomao->addMember(zoro);
    caomao->addMember(sanji);
    caomao->addMember(nami);
    Merry* sunny = new Merry(caomao);
    sunny->feature();
    sunny->showTeam();

    // 斯摩格的船队
    SmokerTeam* team = new SmokerTeam("斯摩格的海军部队");
    Person* smoker = new Person("斯摩格", "中将", "冒烟果实能力者", "", "爱吃烟
    Person* dasiqi = new Person("达斯琪", "大佐", "一刀流", "", "近视");
    team->addMember(smoker);
    team->addMember(dasiqi);
    HaiJunShip* ship = new HaiJunShip(team);
    ship->feature();
    ship->showTeam();

    delete caomao;
    delete sunny;
    delete team;
    delete ship;

    return 0;
}
```

程序输出的结果为:

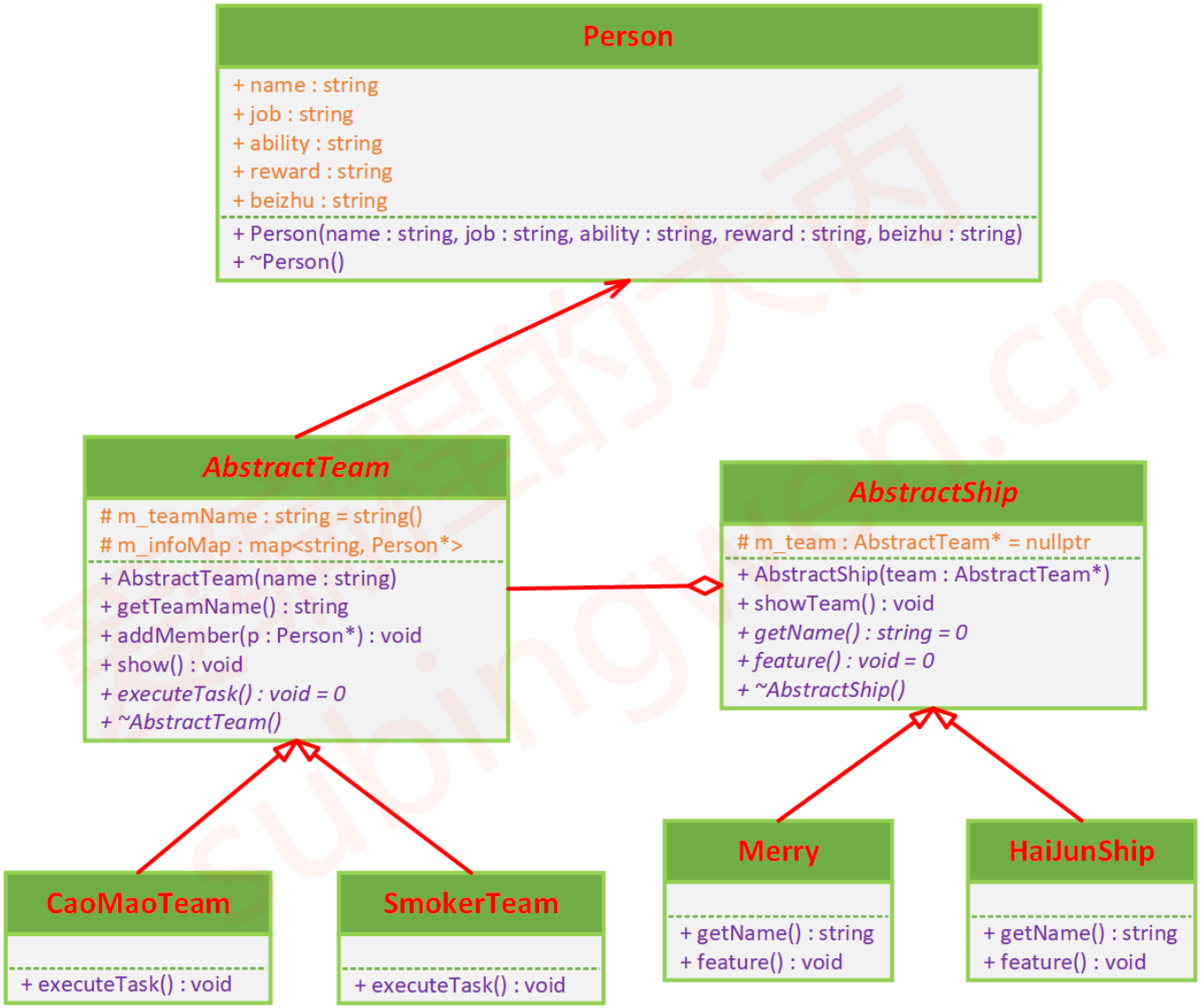
C++

```
前进·梅利号 -- 船首为羊头，在司法岛化身船精灵舍己救下了草帽一伙！
草帽海贼团：
【Name: 路飞, Job: 船长, Ability: 橡胶果实能力者, MoneyReward: 30亿贝里, BeiZhu:
【Name: 娜美, Job: 航海士, Ability: 天候棒+宙斯, MoneyReward: 3亿6600万贝里, Bei
【Name: 山治, Job: 厨师, Ability: 隐形黑, MoneyReward: 10亿3200万贝里, BeiZhu:
【Name: 索隆, Job: 剑士, Ability: 三刀流, MoneyReward: 11亿1100万贝里, BeiZhu:
在海上冒险，找到 ONE PIECE 成为海贼王！
====================================
无敌海军号 -- 船底由海楼石建造，可以穿过无风带的巨大炮舰！
斯摩格的海军部队：
【Name: 达斯琪, Job: 大佐, Ability: 一刀流, MoneyReward: , BeiZhu: 近视】
【Name: 斯摩格, Job: 中将, Ability: 冒烟果实能力者, MoneyReward: , BeiZhu: 爱吃
为了正义，先将草帽一伙一网打尽!!!
================== 资源释放 ==================
路飞被析构...
娜美被析构...
山治被析构...
索隆被析构...
达斯琪被析构...
斯摩格被析构...
```

这样，我们就通过桥接模式完成了对于 海贼 或者 海军 的管理。

3. 结构图

最后根据上面的代码将其对应的桥接模式的 UML 类图画出来（学会桥接模式之后，应该先画类图在写程序。）



上面已经多次提到，桥接模式就是就将抽象和实现分离开来，在上图中描述二者之间的聚合关系的那条线就可以看做是一座桥梁，把两个独立的对象关联到了一起。在某些业务场景下抽象部分或者实现部分可能是没有子类的，这在桥接模式中是被允许的。

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