作业六

概念题

1. 什么是抽象类？抽象类的作用是什么？抽象类的派生类是否一定要给出纯虚函数的实现？

抽象类是一种特殊的类，它不能被实例化，只能被用作其他类的基类。抽象类中可以包含抽象方法（也称为纯虚函数），这些方法在抽象类中没有具体的实现，而是在其派生类中实现。抽象类的作用在于定义一种通用的接口或者行为，强制其派生类去实现特定的方法或者属性。

抽象类的派生类不一定要给出所有纯虚函数的实现，但如果派生类不实现所有纯虚函数，那么它也会成为抽象类，无法实例化。通常情况下，如果一个抽象类有纯虚函数，那么派生类必须实现这些纯虚函数才能具有实例化的能力，否则派生类也会变成抽象类。

2. 使用类聚合的方式和类组合的方式复用代码有什么不同？什么情况下适合使用聚合？什么情况下适合使用组合？在编程时需要注意什么？

类聚合
类聚合表示一种“拥有”关系，其中一个类包含了另一个类的对象作为其成员，并且这个对象可以在外部被访问。
类聚合的对象之间的生命周期可以是独立的，即包含类的对象可以在不影响被聚合类的情况下被创建或销毁。
适合使用聚合的情况是当一个类需要使用另一个类的功能，并且这个功能可以被多个类共享时。

类组合
类组合表示一种更强的“拥有”关系，其中一个类的对象是另一个类对象的一部分，而不是简单地引用。
类组合通常在组合对象的构造函数中创建组合对象，而且组合对象的生命周期与包含类对象的生命周期紧密相关。
适合使用组合的情况是当一个类必须拥有另一个类的对象，并且这个对象是该类的一部分，不能独立存在时。

3. 聚合/组合相比继承的代码复用有哪些优点？能否仅仅通过前两者实现代码复用？为什么？

优点：灵活、低耦合、适用性强
虽然聚合和组合可以实现代码复用，但并不是所有情况都适合使用。有些情况下，继承仍然是更合适的选择。例如，当子类需要完全继承父类的行为并且与父类具有相同的接口时，继承可以更清晰地表示对象之间的“is-a”关系。此外，继承还可以通过多态实现运行时的动态行为，这是聚合和组合无法直接提供的。

4. 多继承解决了什么问题？额外引入了哪些问题？这些问题如何解决？

代码复用：多继承允许一个子类同时继承多个父类的特性，从而实现代码的复用。这意味着子类可以获得多个父类的行为和属性，而不需要重复编写相同的代码。

多重角色：有些类在现实世界中可能同时扮演多个角色，多继承可以更自然地模拟这种情况。例如，一个机器人类可能同时继承自动化设备类和人工智能类。

引入问题：

菱形继承问题：当一个类同时继承自两个不相关的类，并且这两个父类又继承自同一个基类时，就会导致菱形继承问题。这可能导致派生类中包含了两份相同的基类成员，造成资源浪费和逻辑混乱。

方法冲突：如果多个父类中具有相同的方法或属性，子类在继承这些父类时可能会产生方法冲突，不知道应该使用哪个版本的方法。这就需要在子类中显式地解决冲突，或者通过虚拟继承等方式避免。

解决方法：

虚拟继承：使用虚拟继承可以解决菱形继承问题。虚拟继承使得基类的共享部分在派生类中只有一份实例，从而避免了资源浪费和逻辑混乱。

重写或别名：当父类中存在方法冲突时，子类可以选择重写其中一个方法，或者为其中一个方法提供别名，以明确指定使用哪个版本的方法。

使用接口：在某些情况下，可以使用接口（抽象类）来代替多继承。一个类可以实现多个接口，从而达到相似的效果，而不会引入多继承的问题。

编程题

题目描述：

模拟实现下列类并完成相应功能：交通工具（Vehicle）描述了不同交通工具的共有特点。汽车（Car）、船（Boat）、水陆两用汽车（AmphibianCar）分别描述了三种不同的交通工具。具体说明如下：

1. 交通工具有一个共同属性 weight ，表示交通工具的重量（吨）；具有一个共同方法setWeight() 用于修改重量。
2. 交通工具有一个 drive() 方法表示可以进行驶，不同的交通工具的行驶可能不同。如：汽车可以在路上行驶；船可以在水中行驶。
3. 水陆两用汽车可以在路上或在水中行驶。其具有一个状态变量，其 drive() 方法根据状态变量不同决定行驶的条件。
4. 水陆两用汽车有一个 setFlag() 方法来更改状态变量。
5. 交通工具有一个共同属性 capacity ，表示交通工具的核载人数（人）；在构造时确定。
6. 交通工具提供两个方法 getOn() 和 getOff() ，分别表示特定乘客的乘坐和离开。
7. 交通工具提供方法 showPassenger() ，用于显示当前乘坐交通工具的乘客信息。
8. 交通工具提供方法 showMembers() ，用于显示交通工具相关配置。

要求：

1. 请为每一个类设计数据成员和成员函数，完成上述功能。
2. 尝试使用C++提供的继承的机制，使得你的设计更加合理。并在作业报告中对你使用的机制进行讨论。

```
Vehicle 作为基类，定义了所有交通工具都具有的属性和方法
Car Boat AmphibianCar 都是Vehicle的派生类，各自实现覆盖了基类的driver方法。

作业里使用抽象类希望各自的派生类都必须实现drive方法。
但是更好的实现应该是将drive的职责单独抽象出来成为一个DoDriverBehaivor类，根据不同情况做出不同行为，
这样一个派生类可以在运行中动态地使用不同的drive方法，适用于水陆两栖车类。也适用于将来可能增加的不同的车辆，而无需改变太多代码。

```

3. 乘客（Passenger）类可以采用如下实现，请自行编写测试用例，展示 getOn() , getOff() 和 showPassenger() 等相关方法的正常运行

```cpp
#include <iostream>
#include <vector>
#include <string>

class Vehicle {
protected:
    double weight;
    int capacity;
    std::vector<Passenger> passengers;
public:
    Vehicle(double _weight, int _capacity) : weight(_weight), capacity(_capacity) {}
    void setWeight(double w) { weight = w; }
    virtual void drive() = 0;
    void getOn(const Passenger& passenger) {
        if (passengers.size() < capacity) {
            passengers.push_back(passenger);
            std::cout << passenger.getName() << " boarded the vehicle." << std::endl;
        } else {
            std::cout << "The vehicle is full. " << passenger.getName() << " cannot board." << std::endl;
        }
    }
    void getOff(const Passenger& passenger) {
        for (auto it = passengers.begin(); it != passengers.end(); ++it) {
            if (it->getName() == passenger.getName()) {
                passengers.erase(it);
                std::cout << passenger.getName() << " left the vehicle." << std::endl;
                return;
            }
        }
        std::cout << passenger.getName() << " is not on the vehicle." << std::endl;
    }
    void showPassengers() const {
        std::cout << "Passengers on the vehicle: ";
        for (const auto& passenger : passengers) {
            std::cout << passenger.getName() << ", ";
        }
        std::cout << std::endl;
    }
    virtual void showMembers() const = 0;
};

class Car : public Vehicle {
public:
    Car(double _weight, int _capacity) : Vehicle(_weight, _capacity) {}
    void drive() override {
        std::cout << "The car is driving on the road." << std::endl;
    }
    void showMembers() const override {
        std::cout << "Car weight: " << weight << " tons, Capacity: " << capacity << " people" << std::endl;
```

```
    }
};

class Boat : public Vehicle {
public:
    Boat(double _weight, int _capacity) : Vehicle(_weight, _capacity) {}
    void drive() override {
        std::cout << "The boat is sailing on the water." << std::endl;
    }
    void showMembers() const override {
        std::cout << "Boat weight: " << weight << " tons, Capacity: " <<
capacity << " people" << std::endl;
    }
};

class AmphibianCar : public Vehicle {
private:
    bool isOnWater;
public:
    AmphibianCar(double _weight, int _capacity) : Vehicle(_weight,
_capacity), isOnWater(false) {}
    void drive() override {
        if (isOnWater) {
            std::cout << "The amphibian car is driving on the water." <<
std::endl;
        } else {
            std::cout << "The amphibian car is driving on the road." <<
std::endl;
        }
    }
    void setFlag(bool flag) { isOnWater = flag; }
    void showMembers() const override {
        std::cout << "Amphibian car weight: " << weight << " tons, Capacity:
" << capacity << " people" << std::endl;
    }
};


```

参考测试用例：

```
int main() {
    Car car(1.5, 4);
    Boat boat(2.5, 6);
    AmphibianCar amphiCar(2.0, 5);

    Passenger p1("Alice");
    Passenger p2("Bob");
    Passenger p3("Charlie");

    car.getOn(p1);
    car.getOn(p2);
    car.showPassengers();
    car.getOn(p3);
    car.showPassengers();
```

```
    car.getOff(p1);
    car.showPassengers();

    boat.getOn(p1);
    boat.getOn(p2);
    boat.showPassengers();
    boat.getOn(p3);
    boat.showPassengers();
    boat.getOff(p2);
    boat.showPassengers();

    amphiCar.drive();
    amphiCar.setFlag(true);
    amphiCar.drive();

    return 0;
}
```

参考输出：

```
Alice boarded the vehicle.
Bob boarded the vehicle.
Passengers on the vehicle: Alice, Bob,
Charlie boarded the vehicle.
Passengers on the vehicle: Alice, Bob, Charlie,
Alice left the vehicle.
Passengers on the vehicle: Bob, Charlie,
Alice boarded the vehicle.
Bob boarded the vehicle.
Passengers on the vehicle: Alice, Bob,
Charlie boarded the vehicle.
Passengers on the vehicle: Alice, Bob, Charlie,
Bob left the vehicle.
Passengers on the vehicle: Alice, Charlie,
The amphibian car is driving on the road.
The amphibian car is driving on the water.
```

提交注意事项

截止时间：2024-4-16 23:59

文件格式：姓名-学号.pdf

提交方式：南大计科在线实验教学平台

请同学们于截止时间前在南大计科在线实验教学平台上提交，每次作业最终只需要提交一个pdf文件即可，以“姓名-学号.pdf”的方式命名。

注意：

请按要求命名文件，并且只提交一个PDF文件，编程题代码请附在PDF中。任何错误的命名和文件格式将影响你的作业得分。