class design practice. write classes in c++

Create classes for Truck, Car and SUV. Your design must meet the following requirments:

1. Those three wil have shared attibutes: 1. horse power(int). 2. seat number(int). 3. MPG(int). 4. Make(stirng). 5. Model(stirng). 6. Vehicle Type(enum, google how to use enum in c++) These shared attributes must be put in the parent class Vehicle. Make sure you understand the meaning of the word in attribute. If you don't know, you can search google, i.e., what is the MPG of a vehicle?

2. These attributes CANNOT be accessed directly from outside of the class (trying to use the attritbutes will cause a build error). Must use member function GetAttributeName or SetAttributeName.

3. Each child class has it's own attributes:. For truck, it has its own member Towing Capacity (float). For car, it has its own member Car Type (it should be an enum of Coupe and Sedan, if you don't know how to use enum, you can google). For SUV, it has its own member AWD (bool, check the meaning of AWD if you don't now). And their own get and set. These members are also not allowed to be accessed from outside, need to use Get and Set to access.

4. Set the Vehicle Type (enum) in the constructor.

5. If an object of Truck, Car and SUV, is pointed by a Vehicle pointer, you should be able to call the member function "GetVehicleType" of Vehicle to get the type of the object. It should be an enum of Truck, Car or SUV. Here there is NO need to use virtual function.

6. If an object of Truck, Car and SUV, is pointed by a Vehicle pointer, you should be able to call the member function "GetMonthlyCost" of vehicle to get different monthly cost for the truck, car and suv. For truck, the monthly cost is MPG \* Tow Capacity / 100; for car, the montly cost is MPG \* 30 for sedan, MPG \* 35 for coupe; for SUV the monthly cost is MPG \* 40 for not AWD, MPG \* 45 for AWD. This one use virtual function.

7. You can build it without any problems.

8. You can run the following main function and get correct results. And you can get build failure after you uncomment line "std::cout << v->mpg << std::endl; "

int main()

{

Car car1, car2;

Truck truck1, truck2;

SUV suv1, suv2;

Vehicle \*v = nullptr;

car1.SetMPG(30);

car1.SetCarType(CAR\_TYPE\_SEDAN); // this will work after you add the enum

car2.SetMPG(30);

car2.SetCarType(CAR\_TYPE\_COUPE); // this will work after you add the enum

truck1.SetMPG(20);

truck1.SetTowingCapacity(6000);

truck2.SetMPG(20);

truck2.SetTowingCapacity(5000);

suv1.SetMPG(25);

suv1.SetAWD(true);

suv2.SetMPG(25);

suv2.SetAWD(false);

v = &car1;

// this line SHOULD cause a BUILD FAILE.

// After you cause the build fail, you can comment it out.

// if you cannot cause build fail. Your code is wrong.

// std::cout << v->mpg << std::endl;

// This will work after you add the enum

if (v->GetVehicleType() == VEHICLE\_TYPE\_CAR) {

std::cout << "test 1 passed\n";

}

else {

std::cout << "test 1 failed\n";

}

v = &car1;

if (v->GetMonthlyCost() == 900) {

std::cout << "test 2 passed\n";

}

else {

std::cout << "test 2 failed\n";

}

v = &truck2;

if (v->GetMonthlyCost() == 1000) {

std::cout << "test 3 passed\n";

}

else {

std::cout << "test 3 failed\n";

}

}