SD3 Programming in C++

“Memory Management”

In this assignment you will practice with constructors, copy constructors, destructors and overloading the = operator.

We have three classes: wheel, car and truck. The UML class model below shows how they are related. As you can see, we have a composition relation between Car and Wheel.

For your memory management this means:

- Wheel does not have allocated memory
- Car manages objects of type Wheel
- Truck manages memory for storing power

One reason for writing a constructor is to initialise an object with default values, if needed you must also allocate memory for local objects.

The destructor removes allocated memory while destroying the object.

The copy constructor and assignment operator need to allocate new memory to enable deep copying of data.

Writing a destructor, copy constructor and assignment operator are only needed when an object has allocated memory.

Wheel therefore does not need a destructor, copy constructor or assignment operator, as the default implementations (when needed, created by the compiler) handles everything just fine. Car and Truck do need these three functions!

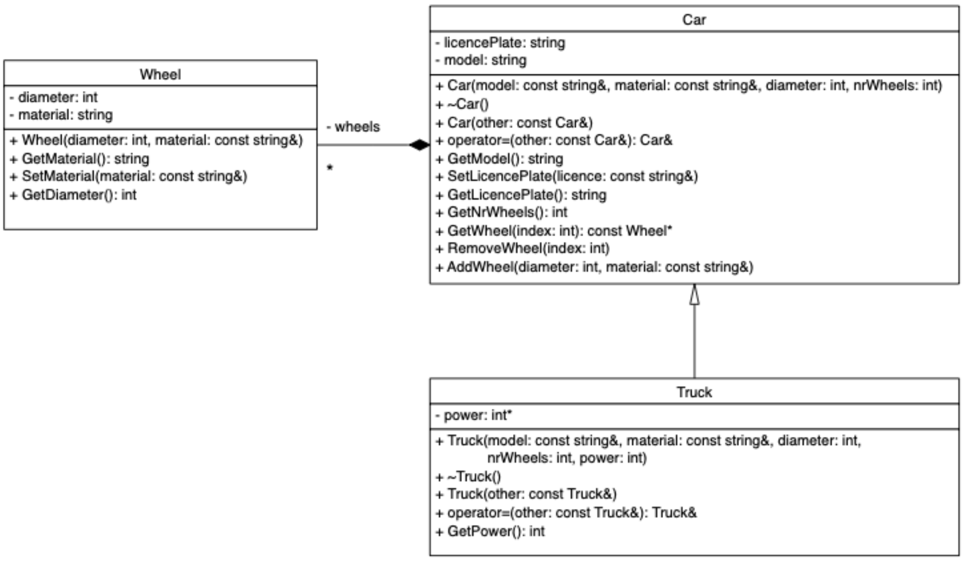
Wheel
- diameter: int
- material: string
+ Wheel(diameter: int, material: const string&)
+ GetMaterial(): string
+ SetMaterial(material: const string&)
+ GetDiameter(): int
- wheels
*
Car
- licencePlate: string
- model: string
+ Car(model: const string&, material: const string&, diameter: int, nrWheels: int)
+ ~Car()
+ Car(other: const Car&)
+ operator=(other: const Car&): Car&
+ GetModel(): string
+ SetLicencePlate(licence: const string&)
+ GetLicencePlate(): string
+ GetNrWheels(): int
+ GetWheel(index: int): const Wheel*
+ RemoveWheel(index: int)
+ AddWheel(diameter: int, material: const string&)
Truck
- power: int*
+ Truck(model: const string&, material: const string&, diameter: int, nrWheels: int, power: int)
+ ~Truck()
+ Truck(other: const Truck&)
+ operator=(other: const Truck&): Truck&
+ GetPower(): int

Figure 1: UML class diagram for Wheel, Car and Truck.

The assignment:

Implement classes Wheel, Car and Truck in a Linux project. The headerfiles are available for download on Sharepoint.
Tip: use inheritance to the max!

Make unit tests using GoogleTest for all three classes (WheelTest, CarTest and TruckTest) in which you prove the functionality of at least:

Correct implementation of:

- Car destructor
- Car copy constructor
- Car assignment operator
- Truck destructor
- Truck copy constructor
- Truck assignment operator

Do show that you have neither wheel nor power leaks in your implementation (i.e.: no memory leaks at all, hint: use valgrind)!