Practical 2 - Inheritance and subclasses. Polymorphism. Operator Overloading.

Instructions

- Download the ***aeda2021_p02.zip*** file from the moodle page and unzip it (contains the ***lib*** folder, the ***Tests*** folder with files ***vehicle.h***, ***vehicle.cpp, fleet,h, fleet.cpp*** and ***tests.cpp***, and the ***CMakeLists*** and ***main.cpp*** files)
- In CLion, open a ***project***, selecting the folder containing the files from the previous point
- Do *"Load CMake Project"* on the *CMakeLists.txt* file
- Run the project (**Run**)
- Note that the ***unit tests for this project are commented***. Remove comments as you implement the tests.
- You must perform the exercise respecting the order of the questions
- Perform the implementation in the carpark.cpp file.

Exercise

To store and manipulate information about a fleet of vehicles, class ***Fleet*** is used:

```
class Fleet {
        vector<Vehicle *> vehicles;
public:
        ...
};
```

Consider also the existence of classes ***Vehicle***, ***MotorVehicle***, ***Car***, ***Truck*** and ***Bicycle***.

```
class  Car {
protected:
  string brand;
  int month, year;
public:
  virtual float calculateTax() const = 0;
  …
};
class MotorVehicle: public Vehicle {
  string fuel; int cylinder;
public:
    …
};
```

```
class Car: public MotorVehicle {
public:
  …
};
class Truck: public MotorVehicle {
  int maximumLoad;
public:
  …
};
class Bicycle: public Vehicle {
 string type;
public:
  …
};
```

a) Implement the constructors of ***Vehicle, MotorVehicle, Car, Truck e Bicycle***, which initialize all data-members of the classes. In the appropriate classes, implement the member functions:

string getFuel() const;

string getBrand() const;

that return the fuel and the brand of the vehicle, respectively.

b) Implement the following member function:

*void Fleet::addVehicle(Vehicle *v1);*

This function adds the vehicle *v1* (car, truck or bicycle) to the vector vehicles.

Implement also the member functions:

int Fleet::numVehicles() const; // return the number of vehicles in vector *vehicles*

int Fleet::lowestYear() const; // return the lowest year of the vehicles in vector *vehicles*
// return 0 if there is no vehicle

c) Implement, for each of the ***Vehicle, MotorVehicle, Car, Truck*** and ***Bicycle*** classes, the member function:

int info() const;

that returns the number of data members and prints their value on the monitor. Should any of these member functions be declared ***virtual***?

d) Implement the function *operador* << in class **Fleet**:

friend ostream & operator<<(ostream & o, const Fleet & f);

This function prints on the monitor the value of the data members of all vehicles present in the vector *vehicles*. Note that this test does not fail. You must validate it by checking the correct writing of the information.

(note: if you did not use virtual methods in c), now reconsider this)

e) Implement the member function:

bool operator < (const Veiculo & v) const;

A vehicle is smaller than another if it is older (check year and month).

f) Implement the following function operator in the **Fleet** class:

*vector<Veiculo *> operator () (int y1) const;*

This function returns a vector of pointers for vehicles whose year of registration is equal to *y1*.

g) Implement, for each of the ***Vehicle, MotorVehicle, Car, Truck*** and ***Bicycle*** classes, the member function:

float calculateTax() const;

This function returns the amount of tax to be paid for the respective vehicle. Should any of these function members be declared virtual?

Only motor vehicles pay tax, and this value is given in the following table:

fuel		*year*	
gas	*other*	*>1995*	*<=1995*
cyl <=1000	cyl <=1500	14,56	8,10
1000< cyl <=1300	1500< cyl <=2000	29,06	14,56
1300< cyl <=1750	2000< cyl <=3000	45,15	22,65
1750< cyl <=2600	cyl >3000	113,98	54,89
2600< cyl <=3500		181,17	87,13
cyl >3500		320,89	148,37

h) Implement the member function:

float Fleet::totalTax() const;

This function returns the sum of the tax amount to pay for vehicles in the vector *vehicles*.

(note: if you did not use virtual methods in g), now reconsider this)

i) Implement the member function:

unsigned Fleet::removeOldVehicles(int y1);

This function removes vehicles whose year of manufacture is less than or equal to *y1* from the vector *vehicles*. Returns the number of vehicles removed.