

2ª aula prática - Herança e subclasses. Polimorfismo. Operadores**Instruções**

- Faça download do ficheiro *aeda1920_p02.zip* da página da disciplina e descomprima-o (contém a pasta *lib*, a pasta *Tests* com os ficheiros *vehicle.h*, *vehicle.cpp*, *fleet.h*, *fleet.cpp* e *tests.cpp*, e os ficheiros *CMakeLists* e *main.cpp*)
- No CLion, abra um *projeto*, selecionando a pasta que contém os ficheiros do ponto anterior.
- Efetuar “Load CMake Project” sobre o ficheiro *CMakeLists.txt*
- Execute o projeto (**Run**)
- Note que os *testes unitários deste projeto estão comentados*. Retire os comentários à medida que vai implementando os testes.
- *Deverá realizar esta ficha respeitando a ordem das alíneas.*

Enunciado

Pretende-se guardar e manipular informação sobre uma frota de veículos, usando a classe *Fleet*:

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

Considere ainda a existência das classes *Vehicle*, *MotorVehicle*, *Car*, *Truck* e *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) Implemente os construtores das classes *Vehicle*, *MotorVehicle*, *Car*, *Truck* e *Bicycle*, que inicializam todos os membros-dado da classe. Implemente, nas classes adequadas, os membros-função:

string getFuel() const

string getBrand() const

que retornam o combustível e a marca, respetivamente, do veículo em questão.

- b) Implemente o membro-função:

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

Esta função adiciona um veículo *v1* (automóvel, camião ou bicicleta) ao vetor *vehicles*.

Implemente ainda os membros-função:

```
int Fleet::numVehicles() const; // retorna o nº de veículos no vector veiculos
```

```
int Fleet::lowestYear() const; // retorna o menor ano dos veículos presentes no vetor veiculos;  
// retorna 0 se não existir nenhum veículo
```

- c) Implemente, para cada uma das classes **Vehicle**, **MotorVehicle**, **Car**, **Truck** e **Bicycle**, o membro-função:

```
int info() const;
```

que retona o número de membros-dado e imprime no monitor o valor destes. Algum(ns) destes membros-função deve ser declarado como virtual?

- d) Implemente a função *operador <<* para a classe **Fleet**:

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

Esta função imprime no monitor o valor dos atributos de todos os veículos presentes no vetor *veiculos*. Note que **este teste não falha**. Deve validar este teste pela verificação da escrita correta da informação relativa aos atributos de cada veículo.

(nota: se não usou métodos virtuais na alínea c) reconsidere agora essa questão)

- e) Implemente o membro-função:

```
bool operator < (const Veiculo & v) const;
```

Um veículo é menor que outro se é mais antigo (verificar ano e mês de fabrico).

- f) Implemente o seguinte operador de função na classe **Fleet**:

```
vector<Veiculo *> operator () (int yearM) const;
```

Esta função retorna um vetor de apontadores para os veículos cujo ano de matrícula é igual a *anoM*.

- g) Implemente, para cada uma das classes **Vehicle**, **MotorVehicle**, **Car**, **Truck** e **Bicycle**, o membro-função:

```
float calculateTax() const;
```

Esta função retorna o valor do imposto a pagar pelo veículo respetivo. Algum(ns) destes membro-função deve ser declarado como virtual?

Só os veículos motorizados pagam imposto, cujo valor é dado pela tabela seguinte:

<i>fuel</i>		<i>year</i>	
<i>gas</i>	<i>other</i>	<i>>1995</i>	<i><=1995</i>
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) Implemente o membro-função:

float Fleet::totalTax() const;

Esta função retorna a soma do valor do imposto a pagar pelos veículos presentes no vetor *veiculos*.

(nota: se não usou métodos virtuais na alínea g) reconsidere agora essa questão)

i) Implemente o membro-função:

unsigned Fleet::removeOldVehicles(int y1);

Esta função remove do vetor *vehicles* os veículos cujo ano de fabrico é menor ou igual a y1. Retorna o numero de veículos removidos.