

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

- Faça download do ficheiro *aeda1920_fp02.zip* da página da disciplina e descomprima-o (contém a pasta *lib*, a pasta *Tests* com os ficheiros *veiculo.h*, *veiculo.cpp*, *frota.h*, *frota.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 **Frota**.

```
class Frota {
    vector<Veiculo *> veiculos;
public:
    ...
};
```

Considere ainda a existência das classes **Veiculo**, **Motorizado**, **Automovel**, **Camiao** e **Bicicleta**.

class Veiculo {	class Automovel: public Motorizado {
protected:	public:
string marca;	...
int mes, ano;	};
public:	class Camiao: public Motorizado {
virtual float calcImposto() const = 0;	int carga_maxima;
...	public:
};	...
class Motorizado: public Veiculo {	};
string combustivel; int cilindrada;	class Bicicleta: public Veiculo {
public:	string tipo;
...	public:
};	...
	};

- a) Implemente os construtores das classes **Veiculo**, **Motorizado**, **Automovel**, **Camiao** e **Bicicleta**, que inicializam todos os membros-dado da classe. Implemente, nas classes adequadas, os membros-função:

string getCombustivel() const

string getMarca() const

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

- b) Implemente o membro-função:

```
void Frota::adicionaVeiculo(Veiculo *v1);
```

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

Implemente ainda os membros-função:

```
int Frota::numVeiculos() const; // retorna o nº de veículos no vector veiculos
```

```
int Frota::menorAno() 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 **Veiculo**, **Motorizado**, **Automovel**, **Camiao** e **Bicicleta**, 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 **Frota**:

```
friend ostream & operator<<(ostream & o, const Frota & 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 **Frota**:

```
vector<Veiculo *> operator () (int anoM) 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 **Veiculo**, **Motorizado**, **Automovel**, **Camiao** e **Bicicleta**, o membro-função:

```
float calcImposto() 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>combustivel</i>		<i>ano</i>	
<i>gasolina</i>	<i>outro</i>	<i>>1995</i>	<i><=1995</i>
<i>cil <=1000</i>	<i>cil <=1500</i>	14,56	8,10
<i>1000<cil<=1300</i>	<i>1500<cil<=2000</i>	29,06	14,56
<i>1300<cil<=1750</i>	<i>2000<cil<=3000</i>	45,15	22,65
<i>1750<cil<=2600</i>	<i>cil>3000</i>	113,98	54,89
<i>2600<cil<=3500</i>		181,17	87,13
<i>cil>3500</i>		320,89	148,37

h) Implemente o membro-função:

float Frota::totalImposto() 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)