2ªaula prática - Herança e subclasses. Polimorfismo. Operadores

Faça download do ficheiro ***aeda1819_fp02.zip*** da página da disciplina e descomprima-o (contém os ficheiros ***Veiculo.h, Veiculo.cpp, Frota.h***, ***Frota.cpp*** e ***Test.cpp***)

- 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*. Poderá executar o projeto como CUTE Test quando quiser saber se a implementação que fez é suficiente para passar no teste correspondente.
- Deve alterar os ficheiros ***Veiculo.h, Veiculo.cpp, Frota.h*** e ***Frota.cpp*** como achar conveniente.

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 {
protected:
  string marca;
  int mes, ano;
public:
  virtual float calcImposto() const = 0;
  …
};
class Motorizado: public Veiculo {
  string combustivel; int cilindrada;
public:
    …
};
```

```
class Automovel: public Motorizado {
public:
  …
};
class Camiao: public Motorizado {
  int carga_maxima;
public:
  …
};
class Bicicleta: public Veiculo {
 string tipo;
public:
  …
};
```

A declaração das classes deve ser efectuada no ficheiro *Frota.h* e a definição dos seus membros-função no ficheiro *Frota.cpp*.

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 *veículos*;
// 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:

combustivel		*ano*	
gasolina	*outro*	*>1995*	*<=1995*
cil <=1000	cil <=1500	14,56	8,10
1000<cil<=1300	1500<cil<=2000	29,06	14,56
1300<cil<=1750	2000<cil<=3000	45,15	22,65
1750<cil<=2600	cil>3000	113,98	54,89
2600<cil<=3500		181,17	87,13
cil>3500		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)