1ªaula prática - Introdução ao CLion e testes unitários. Classes e vetores. (mais)

Instruções

- Faça download do ficheiro ***aed2122_p01_extra.zip*** da página da disciplina e descomprima-o (contém a pasta ***lib,*** a pasta ***Tests*** com os ficheiros ***mail.h***, ***mail.cpp*****, postman.h, postman.cpp, postOffice.h, postOffice.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**)
- Efetue a implementação no ficheiro ***parque.cpp***.

Enunciado

Pretende-se implementar um sistema de gestão da correspondência numa estação de correio. A correspondência (classe **Mail**) é identificada por nome do remetente (*sender*), nome do destinatário (*receiver*) e código postal do destinatário (*zipCode*). Por uma questão de simplificação, a correspondência considerada neste cenário tem peso máximo de 2Kg, só é efetuado o envio para território nacional e existem apenas dois tipos de envio: correio normal e correio verde.

O correio normal é caracterizado por mais um atributo, que é o peso (*weight*) da correspondência e o correio verde pelo tipo de embalagem usada (*envelope*, *bag*, *box*). As classes *RegularMail* e *GreenMail* caracterizam os dois tipos de correspondência enumerados.

A classe **Postman** identifica um carteiro, e inclui um identificador (*id*), o nome do carteiro (*name*) e a correspondência que este está encarregue de distribuir (*myMail*).

A classe **PostOffice** identifica uma estação de correio e inclui informação sobre os carteiros que aí trabalham (*postmen*), gama de códigos postais da localidade que a estação serve ([*firstZipCode*, *lastZipCode*]), correspondência entregue pelos clientes para envio para outras estações de correio (*mailToSend*) e correspondência a ser distribuída na localidade (*mailToDeliver*).

```
class Mail {
   string sender;
   string receiver;
   string zipCode;
public:
   Mail (string send, string rec, string code);
   // …
};

class RegularMail: public Mail {
     unsigned int weight;
public:
     RegularMail (string send, string rec, string code,
                                    unsigned int w);
};

class GreenMail: public Mail {
     string type;        // "envelope", "bag", "box"
public:
     GreenMail (string send, string rec, string code, string t);
};
```

```
class Postman {
      unsigned int id;
      string name;
      vector<Mail *> myMail;
public:
      Postman();
      // …
};

class PostOffice {
      vector<Mail *> mailToSend;
      vector<Mail *> mailToDeliver;
      vector<Postman> postmen;
      string firstZipCode, lastZipCode;
public:
      PostOffice();
      PostOffice(string fzcode, string lzcode);
      // …
};
```

a) Implemente, nas classes que considerar necessário, o membro-função que efetua o cálculo do preço do selo, em cêntimos, para envio de uma correspondência:

unsigned int getPrice () const;

Esta função retorna o preço do envio da correspondência, sabendo que:

- o custo de envio de correspondência por correio normal depende do seu peso (p):
  50 cent, se p<=20g ; 75 cent, se 20<p<=100g ; 140 cent, se 100<p<=500g ; 325 cent, se p>500
- o custo do envio de correspondência por correio verde depende do tipo de embalagem usada:
  80 cent, se envelope (*envelope*) ; 200 cent, se saqueta (*bag*) ; 240 cent, se caixa de cartão (*box*)

b)] Implemente na classe **PostOffice** o membro-função:

*vector<Mail *> removePostman(string name)*

Esta função remove do vetor *postmen* o carteiro de nome *name*, retornando a correspondência que estava a cargo desse carteiro (*myMail*). Se não existe nenhum carteiro de nome *name*, a função retorna um vetor vazio.

c) Implemente na classe **PostOffice** o membro-função:

*vector<Mail *> endOfDay(unsigned int &balance)*

No final do dia, o responsável pela estação de correios deve:

- verificar se o valor em caixa está correto. Esta função calcula o preço total relativo a toda a correspondência entregue na estação de correios (vetor *mailToSend*) e coloca esse valor no argumento *balance*

- tratar a correspondência entregue na estação de correios (esvaziar o vetor *mailToSend)*, sendo: a correspondência a entregar pelos carteiros adicionada ao vetor *mailToDeliver* (se zipCode está na gama dos códigos postais da estação ([*firstZipCode*, *lastZipCode*]) ; a correspondência a enviar para outras estações colocada num vetor que é o retorno da função *endOfDay*. Nota: A classe **PostOffice** já possui o membro-função *void addMailToDeliver(Mail *m)*.

d) Implemente na classe **PostOffice** o membro-função:

*Postman addMailToPostman(Mail *m, string name)*

Esta função adiciona a correspondência *m* ao vetor de correspondências que o carteiro de nome *name* é responsável por distribuir (*myMail*) e retorna esse carteiro. Se não existir um carteiro de nome *name*, a função deve lançar a exceção *NoPostmanException*. Esta classe exceção deve incluir o membro-função *getName()* que retorna o nome do carteiro não existente.