UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Educação e Ciência

p-Fólio

U.C. (21093)

Programação por Objetos

XX de Julho de 2017

-- INSTRUÇÕES --

- O estudante deverá responder à prova na folha de ponto e preencher o cabeçalho e todos os espaços reservados à sua identificação, com letra legível.
- Sempre que não utilize o enunciado da prova para resposta, poderá ficar na posse do mesmo.
- No caso de provas com escolha múltipla, **sem grelha de resposta**, deverá indicar a resposta correcta na folha de ponto, indicando o número da pergunta e a resposta que considera correcta.
- No caso de provas com escolha múltipla, **com grelha de resposta, tabela e/ou espaços para preenchimento**, deverá efectuar as respostas no enunciado, pelo que o mesmo deverá ser entregue ao vigilante, juntamente com a folha de ponto, **não sendo permitido ao estudante levar o enunciado.**
- Verifique no momento da entrega da(s) folha(s) de ponto se todas as páginas estão rubricadas pelo vigilante. Caso necessite de mais do que uma folha de ponto, deverá numerá-las no canto superior direito.
- Em hipótese alguma serão aceites folhas de ponto dobradas ou danificadas.
- Exclui-se, para efeitos de classificação, toda e qualquer resposta apresentada em folhas de rascunho.
- Os telemóveis deverão ser desligados durante toda a prova e os objetos pessoais deixados em local próprio da sala de exame.
- A prova é constituída por **1** página e termina com a palavra **FIM**. Verifique o seu exemplar e, caso encontre alguma anomalia, dirija-se ao professor vigilante nos primeiros 15 minutos da mesma, pois qualquer reclamação sobre defeito(s) de formatação e/ou de impressão que dificultem a leitura não será aceite depois deste período.
- Utilize unicamente tinta azul ou preta.
- Responda às questões de forma clara, sucinta, e apresente todos os cálculos.
- Quando solicitado, apresente ainda uma representação gráfica do resultado final obtido na questão.
- A cotação de cada uma das questões é indicada junto do enunciado da mesma.
- A prova é **SEM CONSULTA**. Todos os elementos necessários à resolução são fornecidos no enunciado.

Duração: 90 minutos

QUESTÃO 1 (12 valores)

A videolocadora "Century" deseja desenvolver uma aplicação que permita a gestão de seus clientes, sócios e filmes (tanto para venda como para aluguel). A situação existente é a seguinte:
Os sócios podem alugar filmes. Os clientes podem apenas comprar filmes. Para os clientes é registado o nome, morada, telefone e o nº do cartão de cidadão. No caso de se tornar sócio, além desses dados, também é atribuído um nº de sócio e um cartão com saldo inicial de 50 €. O cartão também possui um número. Quando um sócio aluga, é feito um débito no saldo de seu cartão. Cada vez que é solicitado um aluguel, é verificado o saldo do cartão, concretizando-se o aluguel se o saldo for positivo. Os filmes podem ser para venda ou compra. Em ambos os casos é registado o título do filme, diretor, ator principal, ano e tipologia (ficção, terror, etc.), preço (diferente entre a venda e a compra) e o *status* (disponível ou não). No caso de aluguel, é registado o número de dias de aluguel e a data de saída. No caso de compra, é emitido um pedido de compra onde se registam os dados do cliente e a data do pedido. A videolocadora deseja também permitir a consulta a sua lista de filmes para alugar ou vender, além de claro, executar as tarefas básicas de gestão (filmes e sócios), que incluem verificar diariamente se há algum filme em falta a ser devolvido.

Esta questão será avaliada da seguinte forma:

a) Declaração da classe (apenas o ficheiro .h) para definir a videolocadora (todos os atributos e métodos necessários, incluindo os *getters* e *setters*) – 3 pontos
b) Declaração da(s) classe(s) (apenas os ficheiros .h) para definir os clientes (todo o tipo) da videolocadora (todos os atributos e métodos necessários, incluindo os *getters* e *setters*) - 3 pontos
c) Definição dos **métodos na classe VideoLocadora** necessários para (ficheiro .cpp):
   - Cancelar um sócio; (2 pontos)
   - Adicionar filmes; (2 pontos)
   - Listar os filmes que tenha para venda ou compra; (2 pontos)

Não esqueça de incluir os *#includes* necessários nos ficheiros .h.

Escreva com letra legível e adicione comentários onde for necessário para maior clareza!

FIM

SOLUÇÃO:

Questão a)

```
#ifndef VIDEOLOCADORA_H_
#define VIDEOLOCADORA_H_

#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include "FilmeCompra.h"
#include "FilmeAluguer.h"
#include "Socio.h"
#include "Pedido.h"

namespace std {

class VideoLocadora {
private:
  string nome;
  string morada;
  vector <FilmeAluguer> vFilme;
  vector <FilmeAluguer>::iterator itFilme;
  vector <FilmeCompra> vFilmeComp;
  vector <FilmeCompra>::iterator itFilmeComp;
  vector <Socio> vSocio;
  vector <Socio>::iterator itSocio;
  vector <Cliente> vCliente;
  vector <Cliente>::iterator itCliente;
public:
  VideoLocadora();
  virtual ~VideoLocadora();
 vector<Cliente> getCliente() const;
 vector<FilmeAluguer> getFilme() const;
 vector<FilmeCompra> getFilmeComp() const;
 string getMorada() const;
 string getNome() const;
 vector<Socio> getSocio() const;
 void setCliente(vector<Cliente> vCliente);
 void setFilme(vector<FilmeAluguer> vFilme);
 void setFilmeComp(vector<FilmeCompra> vFilmeComp);
 void setMorada(string morada);
 void setNome(string nome);
 void setSocio(vector<Socio> vSocio);
 void alugaFilme();
 void devolveFilme();
 void adicionaFilme();
 void listarFilmes();
 void compraFilme();
 void subscreveSocio();
 void cancelaSocio();
 bool verificaStatus();// verifica se há filmes por devolver fora do prazo
};

} /* namespace std */
#endif /* VIDEOLOCADORA_H_ */
```

Questão b)

```
#ifndef CLIENTE_H_
#define CLIENTE_H_
#include <iostream>
#include <string>

namespace std {

class Cliente {
private:
  string nome;
  string morada;
  long int telefone;
  int bi;
public:
  Cliente();
  virtual ~Cliente();
  string getNome() const;
  string getMorada() const;
  int getBI() const;
  long int getTelefone();
  void setNome(string nome);
  void setMorada(string morada);
  void setTelefone(long int telefone);
  void setBI(int bi);
};




} /* namespace std */
#endif /* CLIENTE_H_ */

#ifndef SOCIO_H_
#define SOCIO_H_
#include "Cliente.h"
#include "Cartao.h"

namespace std {

class Socio:public std::Cliente {
private:
  int nSocio;
  Cartao cartao;
public:
  Socio();
  virtual ~Socio();
  int getnSocio() const;
  void setnSocio(int nSocio);
 Cartao getCartao() const;
 void setCartao(Cartao cartao);
 virtual void atualizaCredito(int op, float valor);
};

} /* namespace std */
#endif /* SOCIO_H_ */
```

Questão c)

```
// Retira sócio da lista da video-locadora
    void VideoLocadora::cancelaSocio()
    {
     int num, cont=0;
     cout << "Informe o nº de sócio a eliminar:" << endl;
     cin >> num;
     for(itSocio = vSocio.begin(); itSocio != vSocio.end(); itSocio++)
     {
      ++cont;
      if (num == itSocio->getnSocio()) vSocio.erase(vSocio.begin()+ cont);
     }
    }

// Adiciona a lista de filmes
    void VideoLocadora::adicionaFilme()
    {
     bool resp = true;
     int op;
     string r;
     while(resp)
     {
      cout << "Este filme é: 1-aluguer, 2-venda" << endl;
      cin >> op;
      switch(op)
      {
       case 1: vFilme.push_back(FilmeAluguer());
               break;
       case 2: vFilmeComp.push_back(FilmeCompra());
              break;
      }
      cout << "Vai inserir mais filmes?  (s/n)"<< endl;
      cin >> r;
      if (r.compare("s")!=0) resp = false;
     }
    }


   // Lista os filmes para venda ou aluguel
    void VideoLocadora::listarFilmes()
    {
     int op;
     cout << "Deseja consultar a lista de filmes para alugar ou comprar? (1 ou 2)"
<< endl;
     cin >> op;
     switch(op)
     {
      case 1: for(itFilme = vFilme.begin();itFilme != vFilme.end(); ++itFilme)
      {
          (*itFilme).imprime();
          cout << "Preço:" << (*itFilme).getPreco() << " euros" << endl;
      }
        break;
      case 2: for(itFilmeComp = vFilmeComp.begin();itFilmeComp !=
     vFilmeComp.end(); ++ itFilme)
         {
             (*itFilmeComp).imprime();
             cout << "Preço:" << (*itFilmeComp).getPreco() << " euros" << endl;
         }
         break;
     }
    }
```