```
 1: #ifndef PERSISTENCIA_H_
 2: #define PERSISTENCIA_H_
 3: #include<string>
 4: #include<iostream>
 5: #include<fstream>
 6: #include"Cliente.h"
 7:
 8: //*************************************************************************
 9: //          CLASSE MAPEADOR DE DADOS DA PERSISTENCIA
10: //*************************************************************************
11:
12: class MapeadorDeDados
13: {
14:     //Metodos
15:         public:
16:            MapeadorDeDados() {};
17:            virtual ~MapeadorDeDados(){};
18:             virtual bool gravarDados(ItemPersistencia *item);
19:             virtual bool recuperarDados(ItemPersistencia *item);
20:             virtual bool removerDados(ItemPersistencia *item);
21:             //  criar
22:         protected:
23:             virtual void gravarNoArmazenamento(ItemPersistencia *item)=0;
24:             virtual bool buscarNoArmazenamento(unsigned int id,ItemPersistencia *it
25:             virtual void removerNoArmazenamento(ItemPersistencia *item)=0;
26: };
27:
28: bool MapeadorDeDados::gravarDados(ItemPersistencia *item)
29: {
30:     if(!buscarNoArmazenamento(item->obterIdentificador(),item)) {
31:   //obterIdentificador retorna o codigo identificador do cliente
32:   //busca retorna os dados do "procuradO"
33:         gravarNoArmazenamento(item);
34:         //com o metodo de "saida" do metodo busca e com a negaçao do mesmo
35:         //eh enviado os dados para serem gravados com o metodo gravarNoArmazenament
36:         return true;//gravado com sucesso
37:     }return false;//nao eh possivel gravar
38: }
39:
40: bool MapeadorDeDados::recuperarDados(ItemPersistencia *item)
41: {
42:     if(buscarNoArmazenamento(item->obterIdentificador(),item)) return true;
43:         else return false;
44: }
45:
46: bool MapeadorDeDados::removerDados(ItemPersistencia *item)//chamdo da classe persi
47: //remove um nome
48: {
49:     if(buscarNoArmazenamento(item->obterIdentificador(),item)){
50:         //recebe o item(objeto da classe Cliente) como parametro da busca
51:         removerNoArmazenamento(item);
52:         //chama o metodo remover da classe MapeadorDeCliente
53:         //arquivo removido e retornado true
54:         return true;
55:     }
56:         else return false;
57: }
58:
59: //*************************************************************************
60: //          CLASSE MAPEADOR DE DADOS DO CLIETNE NA PERSISTENCIA
61: //*************************************************************************
62: //esta classe "cuida" dos cadastro de clientes no disco
```

```
63:
64: class MapeadorDeCliente:  public MapeadorDeDados
65: {
66:         private:
67:             const std::string nomeDoArquivoNoDisco;
68:         public:
69:             MapeadorDeCliente();
70:             void gravarNoArmazenamento(ItemPersistencia *item);
71:             bool buscarNoArmazenamento(unsigned int id,ItemPersistencia *item);
72:             //tem como parametro de entrada um numero de identificacao e de saida
73:             //os dados do respectvo number de identificao procurado
74:             void removerNoArmazenamento(ItemPersistencia *item);
75:             //remove um nome do arquivo
76: };
77:
78: MapeadorDeCliente::MapeadorDeCliente():
79:                     nomeDoArquivoNoDisco("ArquivoDeCliente.txt")
80: {
81:     std::fstream arquivo;
82:     arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::in);
83:     if(!arquivo.is_open()) arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::out
84:     arquivo.close();
85: }
86:
87: bool MapeadorDeCliente::buscarNoArmazenamento(unsigned int id, ItemPersistencia *it
88: //tem como parametro de entrada o nro identificador de um cliente
89: //
90: {
91:     std::fstream arquivo;
92:     //abre um arquivo para leitura e escrita
93:     std::string linha;
94:     Cliente pessoaAux;
95:     Cliente *pessoa;
96:     pessoa=static_cast<Cliente *>(item);
97:     //convertendo objeto da classe ItemPersistencia para um objeto da classe Cl
98:     arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::in);
99:     //abrindo arquivo para leitura
100:     getline(arquivo,linha);
101:     while(!arquivo.eof()){
102:  //este laço busca se existe o nro de identificacao no arquivo
103:         pessoaAux.materializar(linha);
104:         //materializar: tem como parametro de entrada uma string e arqmazena os da
105:         if(id==pessoaAux.obterIdentificador()){
106:         //compara o nro identificador q eh um atributo de entrada com o nro identi
107:         //de cada linha do arquivo
108:             arquivo.close();
109:             //se encontrado fechara o arquivo
110:             std::string nome,email;
111:             unsigned int id;
112:             long telefone;
113:             pessoaAux.obter(id,nome,telefone,email);
114:             //recebera os dados do obj temporario e armazenara eles em um novo obj
115:             //este novo objeto e o paramentro de saida q foi reservado um espaco na
116:             //parametro e na declaracao do obje pessoa da classe Cliente
117:             pessoa->atribuir(id,nome,telefone,email);
118:             //apos atribuido este objeto retornara como parametro e o metodo retor
119:             return true;
120:         }
121:         getline(arquivo,linha);
122:     }
123:     arquivo.close();
124:     //caso n encontre fechara o arquivo e retornara false
```

```
125:     return false;
126: }
127:
128: void MapeadorDeCliente::gravarNoArmazenamento(ItemPersistencia *item)
129: {
130:         // Convertendo da variavel polimorfica do tipo ItemPersistencia para uma do
131:             Cliente *cliente;
132:             cliente=static_cast<Cliente *>(item);
133:             //convertendo o objeto para armazena-los no arquivo
134:
135:         //  Gravar no arquivo
136:             std::ofstream arquivo;
137:             arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::app);
138:
139:             std::string linha = cliente->desmaterializar();
140:             //o comando desmaterializar da classe cliente retorna uma string
141:             arquivo<<linha<<std::endl;
142:             //gravando a string dentro do arquivo
143:             arquivo.close();
144: }
145:
146:
147: //   ESTOU AQUI
148:
149: void MapeadorDeCliente::removerNoArmazenamento(ItemPersistencia *item)
150: {
151:         // Convertendo da variavel polimorfica do tipo ItemPersistencia para uma do
152:         //  Cliente *cliente;
153:         //  cliente=static_cast<Cliente *>(item);
154:
155:         // Criando Arquivo Auxiliar
156:         std::fstream arquivoAux;
157:         arquivoAux.open("ArquivoTemporario.txt",std::ios::out|std::ios::trunc);
158:         //abrindo arquivo para escrita  e limpando o mesmo
159:
160:         //  abrindo arquivo Original no inicio
161:         std::fstream arquivo;
162:         arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::in);
163:         //abrindo o arquivo original para leitura
164:
165:         std::string linha;
166:         Cliente pessoaAux;
167:         unsigned int id = item->obterIdentificador();
168:         //obtendo o numero identificador do cliente q vai ser apagado
169:         getline(arquivo,linha);
170:         //lendo a primeira linha do arquivo original
171:         while(!arquivo.eof()){
172:             pessoaAux.materializar(linha);
173:             if(id!=pessoaAux.obterIdentificador()) arquivoAux<<linha<<std::endl;
174:             //if: copiando para o arquivoAux os dados dos clientes q n contem akel
175:             getline(arquivo,linha);
176:             //lendo a proima linha do arquivo, te o final do mesmo
177:         }
178:         arquivo.close();
179:         arquivoAux.close();
180:         //fechando os dois arquivos
181:
182:         //apagando o arquivo original e renomeando o temporario
183:         std::remove("ArquivoDeCliente.txt");
184:         std::rename("ArquivoTemporario.txt","ArquivoDeCliente.txt");
185:         // Apagando arquivo Original
186: //      std::string comando = "rm "+nomeDoArquivoNoDisco;
```

```
187: //      system(comando.c_str());
188:
189:         // Renomeia arquivo Auxiliar
190: //      comando =
191:
192: }
193:
194:
195: //*************************************************************************
196: //          CLASSE PERSISTENCIA
197: //*************************************************************************
198:
199: class Persistencia
200: {
201:     //Atributos
202:         private:
203:             MapeadorDeDados  *baseDeDados[3];
204:     //Metodos
205:         public:
206:             Persistencia();
207:             ~Persistencia();
208:             bool gravar(ItemPersistencia *item);
209:             bool recuperar(ItemPersistencia *item);
210:             bool remover(ItemPersistencia *item);
211:
212: };
213:
214: Persistencia::Persistencia()
215: {
216:     baseDeDados[0] = new MapeadorDeCliente();
217:                 //baseDeDados[1] = new MapeadorFilme();
218:                 //baseDeDados[2] = new MapeadorLocacao();
219: }
220:
221: Persistencia::~Persistencia()
222: {
223:                 delete baseDeDados[0];
224: }
225:
226: bool Persistencia::gravar(ItemPersistencia *item)//recebendo um objeto como parame
227: {
228:                  return baseDeDados[item->obterTipoDaClasse()]->gravarDados(item);
229:                  //baseDeDados: um objeto da classe Mapeador de Dados
230:                  //Mapeador de Dados
231:                  //item->obterTipoClasse() : chamada do metodo da clsse ItemPersis
232:                  //TIPO DE CLASSE SERIA SE ELA EH UMA PESSOA, FITA OU OUTRO
233:                  //ENTAO BASE DE DADOS E UM VETOR COM APENAS 3 POSICOES PARA IDENT
234:                  //QUE BASE DE DADOS BUSCAR...CLIENTE, FILME OU PESSOA
235:
236: }
237:
238: bool Persistencia::recuperar(ItemPersistencia *item)
239: {
240:                  return baseDeDados[item->obterTipoDaClasse()]->recuperarDados(ite
241: }
242:
243: bool Persistencia::remover(ItemPersistencia *item)
244: {
245:                  return baseDeDados[item->obterTipoDaClasse()]->removerDados(item)
246:                  //apos feito todo o processo de remocao sera retornado o "status"
247: }
248:
```

```
249: #endif /*PERSISTENCIA_H_*/
250:
```