Classe intemPersistencia.h

```
#ifndef CLASSEITEMPERSISTENCIA_H_
#define CLASSEITEMPERSISTENCIA_H_
#include<string>
class ItemPersistencia
{
	protected:
		unsigned int identificador;
		unsigned int tipoDaClasse;
		//essa var serve para identificar o tipo de classe
		// 0 - CLIENTE
		// 1 - FILME
		// 2 - LOCACAO
	public:
	ItemPersistencia(unsigned int
identificador):identificador(identificador){};
//tem como parametro um numero digitado pelo usuario, aonde sera o
identificador
//no construtor da classe Cliete tem a inicializacao do atributo tipoClasse
	virtual ~ItemPersistencia(){};
	unsigned int obterIdentificador()const{return identificador;};
	unsigned int obterTipoDaClasse()const{return tipoDaClasse;};
	virtual const std::string desmaterializar() = 0;
	virtual void materializar(const std::string s) = 0 ;
};
#endif /*CLASSEITEMPERSISTENCIA_H_*/
```

****

classe Cliente.h

```
#ifndef CLASSECLIENTE_H_
#define CLASSECLIENTE_H_
#include"ItemPersistencia.h"
#include<string>
//TIPO DE CLASSE 0 = CLIENTE
class Cliente: public ItemPersistencia
{
  private:
	std::string nome;
	long telefone;
	std::string email;
  public:
	Cliente():ItemPersistencia(0){tipoDaClasse=0;};
		Cliente( unsigned int identificador, const std::string
&nome, long telefone, const std::string &email);
	~Cliente(){};
	void obter( unsigned int &identificador,std::string &nome, long
&telefone,std::string &email)const;
	void atribuir(unsigned int identificador, const std::string &nome,
long telefone, const std::string &email);
	const std::string desmaterializar();
//transforma atributos em uma string, esta obtendo os dados dos atributos
para string
	void materializar(const std::string s);
//recebe uma string como parametro e armazena nos atributos os respectivos
valores
};//fim da classe

Cliente::Cliente( unsigned int identificador, const std::string &nome,
	long telefone, const std::string &email):
	ItemPersistencia(identificador),
 //recebe um numero(unsigned int) digitado "no programa principal"
	nome(nome),
	telefone(telefone),
	email(email)
{
	tipoDaClasse=0;
//este tipo seria para identificacao q eh uma pessoa para a classe
//itemPersistencia
}

void Cliente::obter(    unsigned int &identificador,std::string &nome,
	long &telefone,std::string &email)const
{
	identificador = this->identificador;
	nome = this->nome;
	telefone = this->telefone;
	email = this->email;
}

void Cliente::atribuir( unsigned int identificador,const std::string &nome, long
telefone,const std::string &email)
{
	this->identificador=identificador;
	this->nome=nome;
	this->telefone=telefone;
	this->email=email;
}

const std::string Cliente::desmaterializar()
//transformando atributos em string's, linha
//mesma funcao do metodo obter
{
	std::string linha;
	std::string idString;
	for(unsigned int aux=identificador;aux;){
		char letra= (aux%10)+48;
		idString=letra+idString;
		aux=aux/10;
	}
	linha=idString;
	linha +=";";
	linha+=nome;
	linha+=";";
	std::string foneString;
	for(unsigned int aux=telefone;aux;){
		char letra= (aux%10)+48;
		foneString=letra+foneString;
		aux=aux/10;
	}
	linha+=foneString;
	linha+=";";
	linha+=email;
	linha+=";";
	return linha;
}

void Cliente::materializar(const std::string s)
//transformando string em atributos
//mesma funcao do metodo atribuir
{
	unsigned int aux=0;
	unsigned int pos=0;
	for(;s[pos]!=';';pos++) aux=aux*10+(s[pos]-48);
	identificador=aux;
	std::string strAux="";
	for(pos++;s[pos]!=';';pos++) strAux+=s[pos];
	nome=strAux;
	long aux2=0;
	for(pos++;s[pos]!=';';pos++) aux2=aux2*10+(s[pos]-48);
	telefone=aux2;
	strAux="";
	for(pos++;s[pos]!=';';pos++) strAux+=s[pos];
	email=strAux;
}

#endif /*CLASSECLIENTE_H_*/
```

Classe Filme.h

```
#ifndef FILME_H_
#define FILME_H_
#include"classeItemPersistencia.h"
#include<string>

//TIPO DE CLASSE 1 = FILME
//classe para guardar dados do filme

class Filme: public ItemPersistencia
//esta classe guarda um objeto da classe itemPersistencia para o nro
//identificador do filme
{
	//Atributos
	private:
		std::string titulo;
		int precoDeCompra;
		int precoDeLocacao;
		bool situacao;
	//Metodos
	public:
		Filme( unsigned int identificador, const std::string
&titulo, int precoDeCompra, int precoDeLocacao, bool situacao);
		void obter(unsigned int &identificador, std::string
&titulo, int &precoDeCompra, int &precoDeLocacao, bool &situacao)const;
		void atribuir(unsigned int identificador, const std::string
&titulo, int precoDeCompra, int precoDeLocacao, bool situacao);
		int obterPrecoDeCompra()const{
		return precoDeCompra;};
		int obterPrecoDeLocacao()const{
		return precoDeLocacao;};
		bool obterSituacao()const{
		return situacao;};
		void atribuirSituacao(bool situacao){
		this->situacao=situacao;};

};

Filme::Filme(unsigned int identificador, const std::string &titulo, int
precoDeCompra, int precoDeLocacao, bool situacao):
					ItemPersistencia(identificador),
	titulo(titulo),
	precoDeCompra(precoDeCompra),
	precoDeLocacao(precoDeLocacao),
	situacao(situacao)

{
	tipoDaClasse=1;
//TIPO DE CLASSE PARA IDENTIFICACAO Q EH UM FILME ( 1 )
}

void Filme::obter(unsigned int &identificador, std::string &titulo, int
&precoDeCompra, int &precoDeLocacao, bool &situacao)const
{
	identificador= this->identificador;
	titulo=this->titulo;
	precoDeCompra=this->precoDeCompra;
	precoDeLocacao=this->precoDeLocacao;
	situacao=this->situacao;
}

void Filme::atribuir(   unsigned int identificador, const std::string &titulo, int
precoDeCompra,  int precoDeLocacao, bool situacao)
{
	this->identificador=identificador;
	this->titulo=titulo;
	this->precoDeCompra=precoDeCompra;
	this->precoDeLocacao=precoDeLocacao;
	this->situacao=situacao;
}

#endif /*FILME_H_*/
```

****

Classe Locação.h

```
#ifndef LOCACAO_H_
#define LOCACAO_H_
#include"classeItemPersistencia.h"
// TIPO DE CLASSE 2 = LOCACAO
class Locacao: public ItemPersistencia
{
	private:
	unsigned int idCliente;
//e o numero de identificacao do cliente, um codigo para identificalo
	unsigned int idFilme;  //e o codigo do filme
	int precoDeLocacao;
	bool situacao;  //situação da fita, locado ou nao
	public:
	Locacao(   unsigned int identificador, unsigned int idCliente,
unsigned int idFilme,  int precoDeLocacao, bool situacao);
	void obter(unsigned int &identificador, unsigned int &idCliente,
unsigned int &idFilme, int &precoDeLocacao, bool &situacao)const;
	bool obterSituacao()const{return situacao;};
	void atribuirSituacao(bool situacao){
	this->situacao=situacao;};
};
Locacao::Locacao(unsigned int identificador, unsigned int idCliente, unsigned
int idFilme,		int precoDeLocacao, bool situacao):
	ItemPersistencia(identificador),
	idCliente(idCliente),
	idFilme(idFilme),
	precoDeLocacao(precoDeLocacao),
	situacao(situacao)
{
	tipoDaClasse=2;
//TIPO DE IDENTIFICACAO PARA DIZER Q EH  UMA LOCACAO
}
void Locacao::obter(   unsigned int &identificador, unsigned int &idCliente,
unsigned int &idFilme,  int &precoDeLocacao, bool &situacao)const
{
	identificador=this->identificador;
	idCliente=this->idCliente;
	idFilme=this->idFilme;
	precoDeLocacao=this->precoDeLocacao;
	situacao=this->situacao;
}
#endif /*LOCACAO_H_*/
```

Classe Persistencia.h

```
#ifndef PERSISTENCIA_H_
#define PERSISTENCIA_H_
#include<string>
#include<iostream>
#include<fstream>
#include"Cliente.h"

class MapeadorDeDados
{
		//Metodos
		public:
		  MapeadorDeDados() {};
		  virtual ~MapeadorDeDados(){};
		  virtual bool gravarDados(ItemPersistencia *item);
		  virtual bool recuperarDados(ItemPersistencia *item);
		  virtual bool removerDados(ItemPersistencia *item);
		//  criar
		protected:
		  virtual void gravarNoArmazenamento(ItemPersistencia *item)=0;
		virtual bool buscarNoArmazenamento(unsigned int
id,ItemPersistencia *item) =0;
		virtual void removerNoArmazenamento(ItemPersistencia
*item)=0;
};
bool MapeadorDeDados::gravarDados(ItemPersistencia *item)
{
if(!buscarNoArmazenamento(item->obterIdentificador(),item)) {
 //obterIdentificador retorna o codigo identificador do cliente
 //busca retorna os dados do "procuradO"
		gravarNoArmazenamento(item);
//com o metodo de "saida" do metodo busca e com a negaçao do //mesmo
//eh enviado os dados para serem gravados com o metodo
gravarNoArmazenamento
		return true;//gravado com sucesso
}return false;
//nao eh possivel gravar
}

bool MapeadorDeDados::recuperarDados(ItemPersistencia *item)
{
if(buscarNoArmazenamento(item->obterIdentificador(),item))
		return true;
else return false;
}
bool MapeadorDeDados::removerDados(ItemPersistencia *item)//chamdo da
classe persistencia
//remove um nome
{
		if(buscarNoArmazenamento(item->obterIdentificador(),item)){
			//recebe o item(objeto da classe Cliente) como
parametro da busca
	removerNoArmazenamento(item);
	//chama o metodo remover da classe MapeadorDeCliente
			//arquivo removido e retornado true
			return true;
		}
			else return false;
}
//*******************************************************CLASSE
MAPEADOR DE DADOS DO CLIETNE NA PERSISTENCIA
//*******************************************************//esta
classe "cuida" dos cadastro de clientes no disco

class MapeadorDeCliente:  public MapeadorDeDados
{
private:
		const std::string nomeDoArquivoNoDisco;
public:
		MapeadorDeCliente();
		void gravarNoArmazenamento(ItemPersistencia *item);
		bool buscarNoArmazenamento(unsigned int id, ItemPersistencia
*item);
//tem como parametro de entrada um numero de identificacao e de //saida
 //os dados do respectvo number de identificao procurado
		void removerNoArmazenamento(ItemPersistencia *item);
//remove um nome do arquivo
};

MapeadorDeCliente::MapeadorDeCliente():
		nomeDoArquivoNoDisco("ArquivoDeCliente.txt")
{
		std::fstream arquivo;
		arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::in);
		if(!arquivo.is_open())
arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::out| std::ios::trunc);
		arquivo.close();
}

bool MapeadorDeCliente::buscarNoArmazenamento(unsigned int id,
ItemPersistencia *item)
//tem como parametro de entrada o nro identificador de um cliente
{
		std::fstream arquivo;
		//abre um arquivo para leitura e escrita
		std::string linha;
		Cliente pessoaAux;
		Cliente *pessoa;
		pessoa=static_cast<Cliente *>(item);
//convertendo objeto da classe ItemPersistencia para um objeto da //classe
Cliente
		arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::in);
 //abrindo arquivo para leitura
		getline(arquivo,linha);
		while(!arquivo.eof()){
 //este laço busca se existe o nro de identificacao no arquivo
			pessoaAux.materializar(linha);
//materializar: tem como parametro de entrada uma string e //arqmazena os
dados em um atributo
			if(id==pessoaAux.obterIdentificador()){
 //compara o nro identificador q eh um atributo de entrada com o //nro
identificador
 //de cada linha do arquivo
				arquivo.close();
		//se encontrado fechara o arquivo
				std::string nome,email;
				unsigned int id;
				long telefone;
				pessoaAux.obter(id,nome,telefone,email);
//recebera os dados do obj temporario e armazenara eles em um //novo
objeto
//este novo objeto e o paramentro de saida q foi reservado um //espaco na
memoria na declaracao do
//parametro e na declaracao do obje pessoa da classe Cliente
				pessoa->atribuir(id,nome,telefone,email);
//apos atribuido este objeto retornara como parametro e o metodo
//retornara true
				return true;
			 }//fim do if
				getline(arquivo,linha);
		}//fim do while
		arquivo.close();
//caso n encontre fechara o arquivo e retornara false
		return false;
}
void MapeadorDeCliente::gravarNoArmazenamento(ItemPersistencia *item)
{
// Convertendo da variavel polimorfica do tipo ItemPersistencia para //uma
do tipo Cliente
		Cliente *cliente;
		cliente=static_cast<Cliente *>(item);
//convertendo o objeto para armazena-los no arquivo
//  Gravar no arquivo
```

```
        std::ofstream arquivo;
        arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::app);

        std::string linha = cliente->desmaterializar();
//o comando desmaterializar da classe cliente retorna uma string
        arquivo<<linha<<std::endl;
//gravando a string dentro do arquivo
        arquivo.close();
}

//   ESTOU AQUI

void MapeadorDeCliente::removerNoArmazenamento(ItemPersistencia
*item)
{
// Convertendo da variavel polimorfica do tipo ItemPersistencia para uma do
//tipo Cliente
        //      Cliente *cliente;
        //      cliente=static_cast<Cliente *>(item);
        // Criando Arquivo Auxiliar
        std::fstream arquivoAux;
        arquivoAux.open("ArquivoTemporario.txt",std::ios::out|
std::ios::trunc);
//abrindo arquivo para escrita  e limpando o mesmo
//  abrindo arquivo Original no inicio
        std::fstream arquivo;
        arquivo.open(nomeDoArquivoNoDisco.c_str(),std::ios::in);
//abrindo o arquivo original para leitura
        std::string linha;
        Cliente pessoaAux;
        unsigned int id = item->obterIdentificador();
//obtendo o numero identificador do cliente q vai ser apagado
        getline(arquivo,linha);
//lendo a primeira linha do arquivo original
        while(!arquivo.eof()){
                pessoaAux.materializar(linha);
if(id!=pessoaAux.obterIdentificador()) arquivoAux<<linha<<std::endl;
//if: copiando para o arquivoAux os dados dos clientes q n contem akele id no
arquivo original
                getline(arquivo,linha);
//lendo a proima linha do arquivo, te o final do mesmo
        }//fim do while
        arquivo.close();
        arquivoAux.close();
//fechando os dois arquivos
//apagando o arquivo original e renomeando o temporario
        std::remove("ArquivoDeCliente.txt");
        std::rename("ArquivoTemporario.txt","ArquivoDeCliente.txt");
// Apagando arquivo Original
//              std::string comando = "rm "+nomeDoArquivoNoDisco;
//              system(comando.c_str());
                // Renomeia arquivo Auxiliar
//              comando =

}


//*************************************************************
//                              CLASSE PERSISTENCIA
//*************************************************************
class Persistencia
{
        //Atributos
        private:
                MapeadorDeDados  *baseDeDados[3];
        //Metodos
        public:
                Persistencia();
                ~Persistencia();
                bool gravar(ItemPersistencia *item);
                bool recuperar(ItemPersistencia *item);
                bool remover(ItemPersistencia *item);
};

Persistencia::Persistencia()
{
        baseDeDados[0] = new MapeadorDeCliente();
                        //baseDeDados[1] = new MapeadorFilme();
                        //baseDeDados[2] = new
MapeadorLocacao();
}

Persistencia::~Persistencia()
{
                                        delete baseDeDados[0];
}

bool Persistencia::gravar(ItemPersistencia *item)
//recebendo um objeto como parametro de uma chamada
{
 return baseDeDados[item->obterTipoDaClasse()]->gravarDados(item);
 //baseDeDados: um objeto da classe Mapeador de Dados
 //Mapeador de Dados
 //item->obterTipoClasse() : chamada do metodo da clsse ItemPersistencia q
//retorna o atributo tipoDaClasse
 //TIPO DE CLASSE SERIA SE ELA EH UMA PESSOA, FITA OU OUTRO
 //ENTAO BASE DE DADOS E UM VETOR COM APENAS 3 POSICOES PARA
//IDENTIFICAR EM
 //QUE BASE DE DADOS BUSCAR...CLIENTE, FILME OU PESSOA
}

bool Persistencia::recuperar(ItemPersistencia *item)
{
 return baseDeDados[item->obterTipoDaClasse()]->recuperarDados(item);
}

bool Persistencia::remover(ItemPersistencia *item)
{
 return baseDeDados[item->obterTipoDaClasse()]->removerDados(item);
  //apos feito todo o processo de remocao sera retornado o "status" da
//operacao
}
#endif /*PERSISTENCIA_H_*/
```

ProgramaPrincipal.cpp

```
#include<iostream>
#include<string>
#include"Interface.h"
#include"Cliente.h"
#include"persistencia.h"
using namespace std;
int main(void)
{
	Interface::menuPrincipal();
	Persistencia per;
///*
//TESTE DE FUNCIONAMENTO DA PERSISTENCIA  - ITEM GRAVACAO

	Cliente teste(10,"Carlos da Silva",2339988,"carlos@teste.dat");

//criando um objeto da classe Cliente,
//PARAMETROS: cliente(identificador, nome, telefone, email)
  if(per.gravar(&teste))std::cout<<"gravado 10 ";

  //enviando um objeto da classe cliente para a persistencia
  //gravar tem como parametro um objeto da classe ItemPersistencia
  //que o mesmo tem dois atributos
	else std::cout<<"nao gravado 10";
	std::cout<<endl<<endl;
teste.atribuir(11,"Antonio dos Santos",5442377,"antonio@teste.dat");
	if(per.gravar(&teste))std::cout<<"gravado 11 ";
		else std::cout<<"nao gravado 11";
	std::cout<<endl<<endl;

	teste.atribuir(12,"Jose Matheus",3278855,"jose@teste.dat");
	if(per.gravar(&teste))std::cout<<"gravado 12 ";
		else std::cout<<"nao gravado 12";
	std::cout<<endl<<endl;

	teste.atribuir(13,"Maria do Carmo",999999,"maria@teste.dat");
	if(per.gravar(&teste))std::cout<<"gravado 13 ";
		else std::cout<<"nao gravado 13";
	std::cout<<endl<<endl;

teste.atribuir(11,"Antonio dos Santos",5442377,"antonio@teste.dat");
	if(per.gravar(&teste))std::cout<<"gravado 11 ";
		else std::cout<<"nao gravado 11";
	std::cout<<endl<<endl;

teste.atribuir(19,"MARCO TULIO RODRIGUES
BRAGA",12345678,"brancotulio@brancotulio.com");
	if(per.gravar(&teste))std::cout<<"gravado 19 ";
		else std::cout<<"nao gravado 19";
	std::cout<<endl<<endl;
//*/

///*
//TESTE DE FUNCIONAMENTO DA PERSISTENCIA  - ITEM RECUPERACAO
	std::string nome,email;
	unsigned int id;
	long telefone;
	Cliente teste2(12,"",0,"");
	if(per.recuperar(&teste2)){
		teste2.obter(id,nome,telefone,email);
		cout<<endl<<"Id: "<<id<<"   Nome: "<<nome;
		cout<<endl<<"Telefone: "<<telefone<<"  Email:
"<<email<<endl<<endl;
	}
	else std::cout<<"nao Recuperado 11"<<endl;

	teste2.atribuir(10,"",0,"");
	if(per.recuperar(&teste2)){
		teste2.obter(id,nome,telefone,email);
		cout<<endl<<"Id: "<<id<<"   Nome: "<<nome;
		cout<<endl<<"Telefone: "<<telefone<<"  Email:
"<<email<<endl<<endl;
	}
		else std::cout<<"nao Recuperado 11"<<endl;

	teste2.atribuir(99,"",0,"");
	if(per.recuperar(&teste2)){
		teste2.obter(id,nome,telefone,email);
		cout<<endl<<"Id: "<<id<<"   Nome: "<<nome;
		cout<<endl<<"Telefone: "<<telefone<<"  Email:
"<<email<<endl<<endl;
	}
		else std::cout<<"nao Recuperado 99"<<endl;

//*/
	//TESTE DE FUNCIONAMENTO DA PERSISTENCIA  - ITEM REMOVER

	std::string nome2,email2;
	unsigned int id2;
	long telefone2;
	Cliente teste22(12,"",0,"");
	if(per.recuperar(&teste22)){
		teste22.obter(id2,nome2,telefone2,email2);
		cout<<endl<<"Id: "<<id2<<"   Nome: "<<nome2;
		cout<<endl<<"Telefone: "<<telefone2<<"  Email:
"<<email2<<endl<<endl;
		if(per.remover(&teste22))std::cout<<"Removido 12";
		else std::cout<<"Removido 12";
	}
	else std::cout<<"nao Recuperado 11"<<endl;

	//Interface::menuPrincipal();
}
```