Trabalho TP1

AUTORES:
Lucas de Moura Quadros 14/0150668
João Paulo Vaz Mendes 17/0002934
Versão 1.0
Sexta, 27 de Setembro de 2019

Sumário

Table of contents

README

Trabalho_TP1

Índice Hierárquico

Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

Índice dos Componentes

Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Índice dos Arquivos

Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Classes

Referência da Classe CartaoCredito

`#include <entidades.h>`

Membros Públicos

- void **defineNumCartao** (const **NumCartao** &num)
  Função para definir número do cartão

- **NumCartao pegaNumCartao** () const
  Função para retornar número do cartão

- void **defineCodVer** (const **CodigoVerificacao** &cod)
  Função para definir código de verificação do cartão

- **CodigoVerificacao pegaCodVer** () const
  Função para retornar codigo de verificação do cartão

- void **defineValidade** (const **Validade** &val)
  Função para definir validade do cartão

- **Validade pegaValidade** () const
  Função para retornar validade do cartão

- void **imprimeIngresso** ()
  Função para imprimir informações de cartão de crédito.

Descrição detalhada

Definição de entidade Cartao de Crédito

Funções membros

void CartaoCredito::defineCodVer (const CodigoVerificacao & *cod*) [inline]

Função para definir código de verificação do cartão

```
137                                                     {
138             this->codigo = cod;
139         }
```

void CartaoCredito::defineNumCartao (const NumCartao & *num*) [inline]

Função para definir número do cartão

```
129                                                     {
```

```
130             this->numero = num;
131         }
```

void CartaoCredito::defineValidade (const Validade & *val*) [inline]

Função para definir validade do cartão

```
145                                                 {
146             this->validade = val;
147         }
```

void CartaoCredito::imprimeIngresso () [inline]

Função para imprimir informações de cartão de crédito.

```
153                                 {
154             std::cout <<"\t== Cartão de crédito Definido ==" << std::endl;
155             std::cout << "Nuúmero do Cartão: " <<
this->pegaNumCartao().pegaNumCartao() << std::endl;
156             std::cout << "Código de verificação do Cartão: " <<
this->pegaCodVer().pegaCodigo() << std::endl;
157             std::cout << "Validade do Cartão: " <<
this->pegaValidade().pegaValidade() << std::endl;
158         }
```

CodigoVerificacao CartaoCredito::pegaCodVer () const [inline]

Função para retornar codigo de verificação do cartão

```
141                                                 {
142             return codigo;
143         }
```

NumCartao CartaoCredito::pegaNumCartao () const [inline]

Função para retornar número do cartão

```
133                                             {
134             return numero;
135         }
```

Validade CartaoCredito::pegaValidade () const [inline]

Função para retornar validade do cartão

```
149                                             {
150             return validade;
151         }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

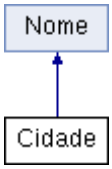
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**entidades.h**

Referência da Classe Cidade

Classe do domínio **Cidade** que herda da classe **Nome**.

```
#include <dominios.h>
```

Diagrama de hierarquia para Cidade:



Membros Públicos

- virtual **~Cidade** ()
  Destrutor virtual para excluir objeto da classe sem problema no polimorphismo.

Outros membros herdados

Descrição detalhada

Classe do domínio **Cidade** que herda da classe **Nome**.

Construtores e Destrutores

virtual Cidade::~Cidade () `[inline], [virtual]`

Destrutor virtual para excluir objeto da classe sem problema no polimorphismo.

```
211 {}
```

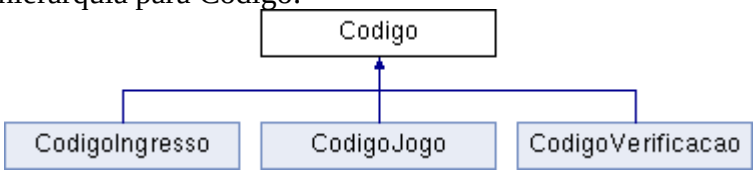
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Codigo

`#include <dominios.h>`

Diagrama de hierarquia para Codigo:



Membros Públicos

- void **defineCodigo** (std::string cod)
  Função para definir Código.

- std::string **pegaCodigo** ()
  Função para retornar Código.

Membros Protegidos

- void **validaCodigo** (std::string &cod) throw (std::invalid_argument)
  Função para validar Código.

Atributos Protegidos

- std::string **codigo**

Funções membros

void Codigo::defineCodigo (std::string *cod*)

Função para definir Código.

Definição de Funções da classe Código

```
228                                         {
229     validaCodigo(cod);
230     this->codigo = cod;
231 }
```

std::string Codigo::pegaCodigo () [inline]

Função para retornar Código.

```
75 {return codigo;}
```

void Codigo::validaCodigo (std::string & *cod*) throw (std::invalid_argument) [protected]

Função para validar Código.

lança exceção se algum dígito não for um número

```
  233                                                                 {
  234     int i,comp = cod.length();
  236     for (i=0;i<comp;i++) {
  237         if(isdigit(cod[i]) == false)
  238             throw invalid_argument("Argumento Inválido: o Código deve ter
exclusivamente números");
  239     }
  240 }
```

Atributos

std::string Codigo::codigo `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

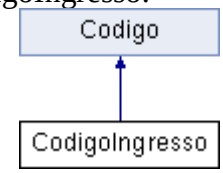
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe CodigoIngresso

Classe do código do **Ingresso**.

`#include <dominios.h>`

Diagrama de hierarquia para CodigoIngresso:



Membros Públicos

- void **defineCodIng** (std::string cod)
  *Função para validar o Código do **Ingresso**.*

Outros membros herdados

Descrição detalhada

Classe do código do **Ingresso**.

Funções membros

void CodigoIngresso::defineCodIng (std::string *cod*)

Função para validar o Código do **Ingresso**.

Função para definir Código de **Ingresso**.

```
358                                                 {
359     validaCodIng(cod);
360     this->codigo = cod;
361 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

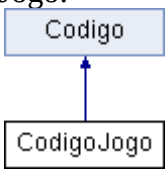
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe CodigoJogo

Classe do código do **Jogo**.

`#include <dominios.h>`

Diagrama de hierarquia para CodigoJogo:



Membros Públicos

- void **defineCodJogo** (std::string cod)
  *Função para definir o Código do **Jogo**.*

Outros membros herdados

Descrição detalhada

Classe do código do **Jogo**.

Funções membros

void CodigoJogo::defineCodJogo (std::string *cod*)

Função para definir o Código do **Jogo**.

Função para definir Código de **Jogo**.

```
342                                             {
343     validaCodJogo(cod);
344     this->codigo = cod;
345 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe CodigoVerificacao

Classe do Código de verificação do cartão CVC.

`#include <dominios.h>`

Diagrama de hierarquia para CodigoVerificacao:

Membros Públicos

- void **defineCodVer** (std::string cod)
  Função para definir CVC.

Outros membros herdados

Descrição detalhada

Classe do Código de verificação do cartão CVC.

Funções membros

void CodigoVerificacao::defineCodVer (std::string *cod*)

Função para definir CVC.

```
326                                                 {
327     validaCodVer(cod);
328     this->codigo = cod;
329 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Cpf

`#include <dominios.h>`

Membros Públicos

- void **defineCpf** (std::string cod)
  Função para definir o número do cpf.

- std::string **pegaCpf** ()
  função para retornar o CPF

Funções membros

void Cpf::defineCpf (std::string *cod*)

Função para definir o número do cpf.

Definição de funções para a classe **Cpf**

```
 97                                {
 98     validaCpf(cod);
 99     this->codigo = cod;
100 }
```

std::string Cpf::pegaCpf () `[inline]`

função para retornar o CPF

```
42 {return codigo;}
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Data

`#include <dominios.h>`

Membros Públicos

- **Data** (int d=1, int m=1, int a=2000)
  *construtor da classe **Data***

- std::string **viraString** ()
  função para retornar data como string

- void **defineData** (int d, int m, int a)
  função para definir os valores de data

- void **defineData** (std::string dt)
  função para definir data a partir de string

- std::string **trataNum** (int num)
  Função para tratar números de dia e mês

- void **validaData** (int &d, int &m, int &a) throw (std::invalid_argument)
  Função para validar data a partir dos atributos int.

- void **validaData** (std::string &dt) throw (std::invalid_argument)
  Função para validar data a partir do atributo string.

Descrição detalhada

Trabalho desenvolvido por: João Paulo Vaz Mendes - Matrícula: 170002934 Lucas de Moura Quadros - Matrícula: 140150668

Construtores e Destrutores

Data::Data (int *d* = 1, int *m* = 1, int *a* = 2000)

construtor da classe **Data**

construtor de data

Definição de Funções da classe **Data**

```
17                              {
18     defineData(d,m,a);
19 }
```

Funções membros

void Data::defineData (int *d*, int *m*, int *a*)

função para definir os valores de data

```
   21                                          {
   22     validaData(d,m,a);
   23     this->dia = d;
   24     this->mes = m;
   25     this->ano = a;
   26     this->data = this->viraString();
   27 }
```

void Data::defineData (std::string *dt*)

função para definir data a partir de string

string Data::trataNum (int *num*)

Função para tratar números de dia e mês

```
   58                                  {
   59     ostringstream aux;
   60     if(num < 10)
   61         aux << "0" << num;
   62     else
   63         aux << num;
   64     return aux.str();
   65 }
```

void Data::validaData (int & *d*, int & *m*, int & *a*) throw (std::invalid_argument)

Função para validar data a partir dos atributos int.

lança exceções no caso de números que superem o escopo de data

```
   67                                                                          {
   69     if(d < 1 || d > 31)
   70        throw invalid_argument("Argumento Inválido: o dia deve ser um número de
1 a 31");
   71     if(m < 1 || m > 12)
   72         throw invalid_argument("Argumento Inválido: o mes deve ser um número de
1 a 12");
   73     if(a < 1900 || a > 3000)
   74         throw invalid_argument("Argumento Inválido: o ano deve ser um número de
1900 a 3000");
   75 }
```

void Data::validaData (std::string & *dt*) throw (std::invalid_argument)

Função para validar data a partir do atributo string.

string Data::viraString ()

função para retornar data como string

```
   51                          {
   52     ostringstream aux;
   53     aux << trataNum(dia) << "/" << trataNum(mes) << "/" << ano;
   54     return aux.str();
   55 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Disponibilidade

`#include <dominios.h>`

Membros Públicos

- void **defineDisp** (int dsp)
  *Função para definir **Disponibilidade**.*

- void **validaDisp** (int &dsp) throw (std::invalid_argument)
  *Função para validar **Disponibilidade**.*

- int **pegaDisp** ()
  *Função para retornar **Disponibilidade**.*

Funções membros

void Disponibilidade::defineDisp (int *dsp*)

Função para definir **Disponibilidade**.

Definição de funções para a classe **Disponibilidade**

```
260                                          {
261     validaDisp(dsp);
262     this->disponibilidade = dsp;
263 }
```

int Disponibilidade::pegaDisp () `[inline]`

Função para retornar **Disponibilidade**.

```
 99 {return disponibilidade;}
```

void Disponibilidade::validaDisp (int & *dsp*) throw (std::invalid_argument)

Função para validar **Disponibilidade**.

```
265                                                                    {
266     if (dsp < 0 || dsp > 300)
267         throw invalid_argument("Argumento Inválido: a disponibilidade deve ser
um valor de 0 a 300");
268 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Estado

classe do **Estado**
`#include <dominios.h>`

Membros Públicos

- void **defineEstado** (std::string est)
  Função para definir o estado.

- std::string **pegaEstado** ()
  Função para retornar estado.

Descrição detalhada

classe do **Estado**

Funções membros

void Estado::defineEstado (std::string *est*)

Função para definir o estado.

```
373                                         {
374     validaEstado(est);
375     this->estado = est;
376 }
```

std::string Estado::pegaEstado () `[inline]`

Função para retornar estado.

```
154 {return estado;}
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Horario

`#include <dominios.h>`

Membros Públicos

- **Horario** (int h=12, int m=0)
  Construtor da classe.

- void **defineHorario** (int h, int m)
  Função para definir os valores do objeto.

- void **defineHorario** (std::string hrr)
  Função para defninir horário a partir de string.

- std::string **viraString** ()
  Função para retornar o Horário como string.

- std::string **trataNum** (int &num)
  Função para tratar números de hora e minuto.

- void **validaHorario** (int &h, int &m) throw (std::invalid_argument)
  função para validar horário dentro dos limites necessários

- void **validaHorario** (std::string &hrr) throw (std::invalid_argument)
  função para validar a string horário para q seja válida

Construtores e Destrutores

Horario::Horario (int *h* = 12, int *m* = 0)

Construtor da classe.

Definição de funções para a classe **Horario**

```
158                                {
159     defineHorario(h,m);
160 }
```

Funções membros

void Horario::defineHorario (int *h*, int *m*)

Função para definir os valores do objeto.

```
162                                            {
163     validaHorario(h,m);
164     this->hora = h;
165     this->minuto = m;
166     this->horario = this->viraString();
```

```
167 }
```

void Horario::defineHorario (std::string *hrr*)

Função para defninir horário a partir de string.

string Horario::trataNum (int & *num*)

Função para tratar números de hora e minuto.

```
193                                     {
194     ostringstream aux;
195     if(num < 10)
196         aux << "0" << num;
197     else
198         aux << num;
199     return aux.str();
200 }
```

void Horario::validaHorario (int & *h*, int & *m*) throw (std::invalid_argument)

função para validar horário dentro dos limites necessários

função para validar horário dentro dos limites necessários dos atributos int

lança exceções no caso de números que superem o escopo de horário

```
202                                                                         {
204     if(h < 0 || h > 23)
205        throw invalid_argument("Argumento Inválido: as horas devem ser um número
de 0 a 23");
206     if(m < 0 || m > 60)
207         throw invalid_argument("Argumento Inválido: os minutos devem ser um
número de 0 a 60");
208 }
```

void Horario::validaHorario (std::string & *hrr*) throw (std::invalid_argument)

função para validar a string horário para q seja válida

string Horario::viraString ()

Função para retornar o Horário como string.

```
187                             {
188     ostringstream aux;
189     aux << trataNum(hora) << ":" << trataNum(minuto);
190     return aux.str();
191 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Ingresso

`#include <entidades.h>`

Membros Públicos

- void **defineCodIng** (const **CodigoIngresso** &cod)
  Função para definir codigo do ingresso.

- **CodigoIngresso pegaCodIng** () const
  Função para retornar codigo do ingresso.

- void **imprimeIngresso** ()
  *Função para imprimir informações de **Ingresso**.*

Descrição detalhada

Definição de entidade **Ingresso**

Funções membros

void Ingresso::defineCodIng (const CodigoIngresso & *cod*) [inline]

Função para definir codigo do ingresso.

```
  108                                                         {
  109             this->codigo = cod;
  110         }
```

void Ingresso::imprimeIngresso () [inline]

Função para imprimir informações de **Ingresso**.

```
  116                                 {
  117             std::cout <<"\t== Ingresso Definido ==" << std::endl;
  118             std::cout << "Código do Ingresso: " <<
this->pegaCodIng().pegaCodigo() << std::endl;
  119         }
```

CodigoIngresso Ingresso::pegaCodIng () const [inline]

Função para retornar codigo do ingresso.

```
  112                                             {
  113             return codigo;
  114         }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**entidades.h**

Referência da Classe Jogo

`#include <entidades.h>`

Membros Públicos

- void **defineCodJogo** (const **CodigoJogo** &cod)
  *Função para definir Código do **Jogo**.*

- **CodigoJogo pegaCodJogo** () const
  *Função para retornar Código do **Jogo**.*

- void **defineNomeJogo** (const **NomeJogo** &nom)
  *Função para definir **Nome** do **Jogo**.*

- **NomeJogo pegaNomeJogo** () const
  *Função para retornar **Nome** do **Jogo**.*

- void **defineCidade** (const **Cidade** &cid)
  *Função para definir **Cidade** do **Jogo**.*

- **Cidade pegaCidade** () const
  *Função para retornar **Cidade** do **Jogo**.*

- void **defineEstado** (const **Estado** &est)
  *Função para definir **Estado** do **Jogo**.*

- **Estado pegaEstado** () const
  *Função para retornar **Estado** do **Jogo**.*

- void **defineTipo** (const **Tipo** &tip)
  *Função para definir **Tipo** do **Jogo**.*

- **Tipo pegaTipo** () const
  *Função para retornar **Tipo** do **Jogo**.*

- void **imprimeJogo** ()
  *Função para imprimir informações de **Jogo**.*

Descrição detalhada

Definição de entidade **Jogo**

Funções membros

void Jogo::defineCidade (const Cidade & *cid*) [inline]

Função para definir **Cidade** do **Jogo**.

```
186                                             {
187         this->cidade = cid;
188     }
```

void Jogo::defineCodJogo (const CodigoJogo & *cod*) [inline]

Função para definir Código do **Jogo**.

```
170                                                 {
171         this->codigo = cod;
172     }
```

void Jogo::defineEstado (const Estado & *est*) [inline]

Função para definir **Estado** do **Jogo**.

```
194                                             {
195         this->estado = est;
196     }
```

void Jogo::defineNomeJogo (const NomeJogo & *nom*) [inline]

Função para definir **Nome** do **Jogo**.

```
178                                                 {
179         this->nome = nom;
180     }
```

void Jogo::defineTipo (const Tipo & *tip*) [inline]

Função para definir **Tipo** do **Jogo**.

```
202                                           {
203         this->tipo = tip;
204     }
```

void Jogo::imprimeJogo () [inline]

Função para imprimir informações de **Jogo**.

```
210                           {
211             std::cout <<"\t== Jogo Definido ==" << std::endl;
212             std::cout << "Código do Jogo: " << this->pegaCodJogo().pegaCodigo()
<< std::endl;
213             std::cout << "Nome do Jogo: " << this->pegaNomeJogo().pegaNome() <<
std::endl;
214             std::cout << "Cidade do Jogo: " << this->pegaCidade().pegaNome() <<
std::endl;
215             std::cout << "Estado do Jogo: " << this->pegaEstado().pegaEstado()
<< std::endl;
216             std::cout << "Tipo do Jogo: " << this->pegaTipo().stringTipo() <<
std::endl;
217     }
```

Cidade Jogo::pegaCidade () const [inline]

Função para retornar **Cidade** do **Jogo**.

```
190                                 {
```

```
191         return cidade;
192     }
```

CodigoJogo Jogo::pegaCodJogo () const [inline]

Função para retornar Código do **Jogo**.

```
174                                     {
175         return codigo;
176     }
```

Estado Jogo::pegaEstado () const [inline]

Função para retornar **Estado** do **Jogo**.

```
198                                 {
199         return estado;
200     }
```

NomeJogo Jogo::pegaNomeJogo () const [inline]

Função para retornar **Nome** do **Jogo**.

```
182                                     {
183         return nome;
184     }
```

Tipo Jogo::pegaTipo () const [inline]

Função para retornar **Tipo** do **Jogo**.

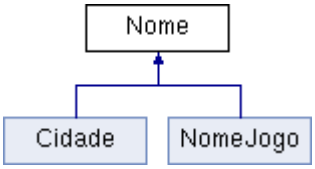
```
206                             {
207         return tipo;
208     }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**entidades.h**

Referência da Classe Nome

Classe Genérica **Nome**.
`#include <dominios.h>`
Diagrama de hierarquia para Nome:



Membros Públicos

- void **defineNome** (std::string n)
  *Função para definir **Nome**.*

- virtual void **validaNome** (std::string)=0 throw (std::invalid_argument)
  Definição genérica de validação do nome.

- std::string **pegaNome** ()
  *Função para retornar **Nome**.*

Atributos Protegidos

- std::string **nome**

Descrição detalhada

Classe Genérica **Nome**.

Funções membros

void Nome::defineNome (std::string *n*) [inline]

Função para definir **Nome**.

```
184                                         {
185             validaNome(n);
186             this->nome=n;
187         }
```

std::string Nome::pegaNome () [inline]

Função para retornar **Nome**.

```
191 {return nome;}
```

virtual void Nome::validaNome (std::string) throw (std::invalid_argument) [pure virtual]

Definição genérica de validação do nome.

Atributos

std::string Nome::nome `[protected]`

A documentação para essa classe foi gerada a partir do seguinte arquivo:

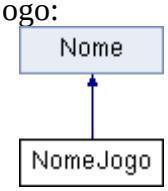
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**

Referência da Classe NomeJogo

Classe do domínio **Nome** de **Jogo** que herda da classe **Nome**.

```
#include <dominios.h>
```

Diagrama de hierarquia para NomeJogo:



Membros Públicos

- virtual **~NomeJogo** ()
  Destrutor virtual para excluir objeto da classe sem problema no polimorphismo.

Outros membros herdados

Descrição detalhada

Classe do domínio **Nome** de **Jogo** que herda da classe **Nome**.

Construtores e Destrutores

virtual NomeJogo::~NomeJogo () `[inline]`, `[virtual]`

Destrutor virtual para excluir objeto da classe sem problema no polimorphismo.

```
202 {}
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe NumCartao

Classe do Domínio Número de Cartão de Crédito.
`#include <dominios.h>`

Membros Públicos

- void **defineNumCartao** (std::string num)
  Função para definir número de cartão de crédito.

- std::string **pegaNumCartao** ()
  Função para retornar número do cartão de crédito.

Descrição detalhada

Classe do Domínio Número de Cartão de Crédito.

Funções membros

void NumCartao::defineNumCartao (std::string *num*)

Função para definir número de cartão de crédito.

Definição de funções para a classe **NumCartao**

```
406                                             {
407     validaNumCartao(num);
408     this->numero = num;
409 }
```

std::string NumCartao::pegaNumCartao () `[inline]`

Função para retornar número do cartão de crédito.

```
166 {return numero;}
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Partida

`#include <entidades.h>`

Membros Públicos

- void **defineCodigo** (const **Codigo** &cod)
  *Função para definir código da **Partida**.*

- **Codigo pegaCodigo** () const
  *Função para retornar código da **Partida**.*

- void **defineData** (const **Data** &dt)
  *Função para definir **Data** da **Partida**.*

- **Data pegaData** () const
  *Função para retornar código da **Partida**.*

- void **defineHorario** (const **Horario** &hr)
  *Função para definir Horário da **Partida**.*

- **Horario pegaHorario** () const
  *Função para retornar Horário da **Partida**.*

- void **definePreco** (const **Preco** &pr)
  *Função para definir Preço da **Partida**.*

- **Preco pegaPreco** () const
  *Função para retornar Preço da **Partida**.*

- void **defineDisp** (const **Disponibilidade** &dp)
  *Função para definir **Disponibilidade** da **Partida**.*

- **Disponibilidade pegaDisp** () const
  *Função para retornar **Disponibilidade** da **Partida**.*

- void **imprimePartida** ()
  *Função para imprimir informações de **Partida**.*

Descrição detalhada

Funções da entidade **Partida**

Funções membros

void Partida::defineCodigo (const Codigo & *cod*) [inline]

Função para definir código da **Partida**.

```
   53                                              {
   54         this->codigo = cod;
   55     }
```

void Partida::defineData (const Data & *dt*) [inline]

Função para definir **Data** da **Partida**.

```
   61                                            {
   62         this->data = dt;
   63     }
```

void Partida::defineDisp (const Disponibilidade & *dp*) [inline]

Função para definir **Disponibilidade** da **Partida**.

```
   85                                                    {
   86         this->dispo = dp;
   87     }
```

void Partida::defineHorario (const Horario & *hr*) [inline]

Função para definir Horário da **Partida**.

```
   69                                                {
   70         this->hora = hr;
   71     }
```

void Partida::definePreco (const Preco & *pr*) [inline]

Função para definir Preço da **Partida**.

```
   77                                              {
   78         this->preco = pr;
   79     }
```

void Partida::imprimePartida () [inline]

Função para imprimir informações de **Partida**.

```
   93                                  {
   94         std::cout <<"\t==Partida Definida==" << std::endl;
   95         std::cout << "Data: " << this->pegaData().viraString() << std::endl;
   96         std::cout << "Horário: " << this->pegaHorario().viraString() <<
std::endl;
   97         std::cout << "Código: " << this->pegaCodigo().pegaCodigo() << std::endl;
   98         std::cout << "Preço: " << this->pegaPreco().pegaPreco() << std::endl;
   99         std::cout << "Disponibilidade: " << this->pegaDisp().pegaDisp() <<
std::endl;
  100     }
```

Codigo Partida::pegaCodigo () const [inline]

Função para retornar código da **Partida**.

```
   57                                    {
   58         return codigo;
   59     }
```

Data Partida::pegaData () const [inline]

Função para retornar código da **Partida**.

```
65                              {
66          return data;
67      }
```

Disponibilidade Partida::pegaDisp () const [inline]

Função para retornar **Disponibilidade** da **Partida**.

```
89                                      {
90          return dispo;
91      }
```

Horario Partida::pegaHorario () const [inline]

Função para retornar Horário da **Partida**.

```
73                                  {
74          return hora;
75      }
```

Preco Partida::pegaPreco () const [inline]

Função para retornar Preço da **Partida**.

```
81                              {
82          return preco;
83      }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**entidades.h**

Referência da Classe Preco

`#include <dominios.h>`

Membros Públicos

- void **definePreco** (double prc)
  Função para definir Preço

- double **pegaPreco** ()
  Função para retornar Preço

- void **validaPreco** (double &prc) throw (std::invalid_argument)
  Função para validar Preço

Funções membros

void Preco::definePreco (double *prc*)

Função para definir Preço

Definição de funções para a classe Preço

```
246                                     {
247     validaPreco(prc);
248     this->preco = prc;
249 }
```

double Preco::pegaPreco () [inline]

Função para retornar Preço

```
 85 {return preco;}
```

void Preco::validaPreco (double & *prc*) throw (std::invalid_argument)

Função para validar Preço

```
251                                                         {
252     if (prc < 0.00 || prc > 3000.00)
253         throw invalid_argument("Argumento Inválido: os valores de preço
excederam os limites que são de 0.00 a 3000.00");
254 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Senha

`#include <dominios.h>`

Membros Públicos

- void **defineSenha** (std::string sn)
  *Função para definir **Senha**.*

- std::string **pegaSenha** ()
  *Função para retornar **Senha**.*

Funções membros

void Senha::defineSenha (std::string *sn*)

Função para definir **Senha**.

```
310                                 {
311     validaSenha(sn);
312     this->senha = sn;
313 }
```

std::string Senha::pegaSenha () `[inline]`

Função para retornar **Senha**.

```
111 {return senha;}
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe Tipo

Classe do domínio **Tipo**.
`#include <dominios.h>`

Membros Públicos

- void **defineTipo** (int d)
  Função de definição do tipo.

- **Id pegaTipo** ()
  Função para retornar tipo.

- std::string **stringTipo** ()
  Função para imprimir tipo.

Descrição detalhada

Classe do domínio **Tipo**.

Funções membros

void Tipo::defineTipo (int *d*)

Função de definição do tipo.

```
554                             {
555     validaDigito(d);
556     this->digito=d;
557     this->tp=Id(d);
558 }
```

Id Tipo::pegaTipo () `[inline]`

Função para retornar tipo.

```
221 {return tp;}
```

string Tipo::stringTipo ()

Função para imprimir tipo.

Função para imprimir o tipo.

```
560                             {
561     string tipo;
562     switch(this->pegaTipo()) {
563             case 1: tipo="LOCAL";
564                     break;
565             case 2: tipo="ESTADUAL";
566                     break;
567             case 3: tipo="NACIONAL";
568                     break;
569             case 4: tipo="INTERNACIONAL";
570                     break;
571     }
```

```
572     return tipo;
573 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Referência da Classe TUCidade

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  *Definições de constantes para reportar resultado do **TUCidade**.*

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **Cidade**

Funções membros

int TUCidade::executar ()

Método para executar o teste.

```
618                         {
619     montar();
620     testarCenarioSucesso();
621     testarCenarioFalha();
622     destruir();
623     return estado;
624 }
```

Atributos

const int TUCidade::FALHA = -1 [static]

const int TUCidade::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do **TUCidade**.

Definições de constantes de TUCidade*/.

Definições de implementação do Teste de Unidade do Domínio **Cidade**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUCodigo

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  Definições de constantes para reportar resultado do TU.

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe Código

Funções membros

int TUCodigo::executar ()

Método que para executar o teste.

```
165                         {
166     montar();
167     testarCenarioSucesso();
168     testarCenarioFalha();
169     destruir();
170     return estado;
171 }
```

Atributos

const int TUCodigo::FALHA = -1 [static]

const int TUCodigo::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do TU.

Definições de constantes de TUCodigo*/.

Definições de implementação do Teste de Unidade do Domínio Código

---

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUCodIng

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  *Definições de constantes para reportar resultado do **TUCodIng**.*

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **CodigoIngresso**

Funções membros

int TUCodIng::executar ()

Método que para executar o teste.

```
412                         {
413     montar();
414     testarCenarioSucesso();
415     testarCenarioFalha();
416     destruir();
417     return estado;
418 }
```

Atributos

const int TUCodIng::FALHA = -1 [static]

const int TUCodIng::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do **TUCodIng**.

Definições de constantes de TUCodIng*/.

Definições de implementação do Teste de Unidade do Domínio **CodigoIngresso**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUCodJogo

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  *Definições de constantes para reportar resultado do **TUCodJogo**.*

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **CodigoJogo**

Funções membros

int TUCodJogo::executar ()

Método que para executar o teste.

```
371                         {
372     montar();
373     testarCenarioSucesso();
374     testarCenarioFalha();
375     destruir();
376     return estado;
377 }
```

Atributos

const int TUCodJogo::FALHA = -1 [static]

const int TUCodJogo::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do **TUCodJogo**.

Definições de constantes de TUCodJogo*/.

Definições de implementação do Teste de Unidade do Domínio **CodigoJogo**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUCodVer

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  *Definições de constantes para reportar resultado do **TUCodVer**.*

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **CodigoVerificacao**

Funções membros

int TUCodVer::executar ()

Método que para executar o teste.

```
330                         {
331     montar();
332     testarCenarioSucesso();
333     testarCenarioFalha();
334     destruir();
335     return estado;
336 }
```

Atributos

const int TUCodVer::FALHA = -1 [static]

const int TUCodVer::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do **TUCodVer**.

Definições de constantes de TUCodVer*/.

Definições de implementação do Teste de Unidade do Domínio **CodigoVerificacao**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUCpf

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  Definições de constantes para reportar resultado do TU.

- const static int **FALHA** = -1

Descrição detalhada

Testede Unidade da Classe do Domínio **Cpf**

Funções membros

int TUCpf::executar ()

Método que para executar o teste.

```
83                      {
84     montar();
85     testarCenarioSucesso();
86     testarCenarioFalha();
87     destruir();
88     return estado;
89 }
```

Atributos

const int TUCpf::FALHA = -1 [static]

const int TUCpf::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do TU.

Definições de constantes de TUCpf*/.

Definições de implementação do Teste de Unidade do Domínio **Cpf**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUData

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  Definições de constantes para reportar resultado do TU.

- const static int **FALHA** = -1

Descrição detalhada

Trabalho desenvolvido por: João Paulo Vaz Mendes - Matrícula: 170002934 Lucas de Moura Quadros - Matrícula: 140150668 Teste de Unidade da Classe do Domínio **Data**

Funções membros

int TUData::executar ()

Método que para executar o teste.

```
42                      {
43     montar();
44     testarCenarioSucesso();
45     testarCenarioFalha();
46     destruir();
47     return estado;
48 }
```

Atributos

const int TUData::FALHA = -1 [static]

const int TUData::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do TU.

Definições de constantes de TUData*/.

Definições de implementação do Teste de Unidade do Domínio **Data**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUDisp

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  Definições de constantes para reportar resultado do TU.

- const static int **FALHA** = -1

Descrição detalhada

Testede Unidade da Classe do Domínio Disponobilidade

Funções membros

int TUDisp::executar ()

Método que para executar o teste.

```
248                     {
249     montar();
250     testarCenarioSucesso();
251     testarCenarioFalha();
252     destruir();
253     return estado;
254 }
```

Atributos

const int TUDisp::FALHA = -1 [static]

const int TUDisp::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do TU.

Definições de constantes de TUDisp*/.

Definições de implementação do Teste de Unidade do Domínio de **Disponibilidade**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUEstado

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  *Definições de constantes para reportar resultado do **TUEstado**.*

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **Estado**

Funções membros

int TUEstado::executar ()

Método para executar o teste.

```
453                         {
454     montar();
455     testarCenarioSucesso();
456     testarCenarioFalha();
457     destruir();
458     return estado;
459 }
```

Atributos

const int TUEstado::FALHA = -1 [static]

const int TUEstado::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do **TUEstado**.

Definições de constantes de TUEstado*/.

Definições de implementação do Teste de Unidade do Domínio **Estado**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUHorario

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  Definições de constantes para reportar resultado do TU.

- const static int **FALHA** = -1

Descrição detalhada

Testede Unidade da Classe do Domínio Horário

Funções membros

int TUHorario::executar ()

Método que para executar o teste.

```
124                         {
125     montar();
126     testarCenarioSucesso();
127     testarCenarioFalha();
128     destruir();
129     return estado;
130 }
```

Atributos

const int TUHorario::FALHA = -1 [static]

const int TUHorario::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do TU.

Definições de constantes de TUHorario*/.

Definições de implementação do Teste de Unidade do Domínio Horário

---

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUNomeJogo

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  *Definições de constantes para reportar resultado do **TUNomeJogo**.*

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **NomeJogo**

Funções membros

int TUNomeJogo::executar ()

Método para executar o teste.

```
577                         {
578     montar();
579     testarCenarioSucesso();
580     testarCenarioFalha();
581     destruir();
582     return estado;
583 }
```

Atributos

const int TUNomeJogo::FALHA = -1 [static]

const int TUNomeJogo::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do **TUNomeJogo**.

Definições de constantes de TUNomeJogo*/.

Definições de implementação do Teste de Unidade do Domínio **Nome** do **Jogo**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUNumCartao

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  *Definições de constantes para reportar resultado do **TUNumCartao**.*

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **NumCartao**

Funções membros

int TUNumCartao::executar ()

Método para executar o teste.

```
494                          {
495     montar();
496     testarCenarioSucesso();
497     testarCenarioFalha();
498     destruir();
499     return estado;
500 }
```

Atributos

const int TUNumCartao::FALHA = -1 [static]

const int TUNumCartao::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do **TUNumCartao**.

Definições de constantes de TUNumCartao*/.

Definições de implementação do Teste de Unidade do Domínio Número do Cartão de Crédito

---

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUPreco

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  Definições de constantes para reportar resultado do TU.

- const static int **FALHA** = -1

Descrição detalhada

Testede Unidade da Classe do Domínio **Preco**

Funções membros

int TUPreco::executar ()

Método que para executar o teste.

```
206                     {
207     montar();
208     testarCenarioSucesso();
209     testarCenarioFalha();
210     destruir();
211     return estado;
212 }
```

Atributos

const int TUPreco::FALHA = -1 [static]

const int TUPreco::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do TU.

Definições de constantes de TUPreco*/.

Definições de implementação do Teste de Unidade do Domínio Preço

---

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUSenha

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  Definições de constantes para reportar resultado do TU.

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **Senha**

Funções membros

int TUSenha::executar ()

Método que para executar o teste.

```
289                     {
290     montar();
291     testarCenarioSucesso();
292     testarCenarioFalha();
293     destruir();
294     return estado;
295 }
```

Atributos

const int TUSenha::FALHA = -1 [static]

const int TUSenha::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do TU.

Definições de constantes de TUSenha*/.

Definições de implementação do Teste de Unidade do Domínio **Senha**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUTipo

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método que para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  *Definições de constantes para reportar resultado do **TUTipo**.*

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **Tipo**

Funções membros

int TUTipo::executar ()

Método que para executar o teste.

```
660                     {
661     montar();
662     testarCenarioSucesso();
663     testarCenarioFalha();
664     destruir();
665     return estado;
666 }
```

Atributos

const int TUTipo::FALHA = -1 [static]

const int TUTipo::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do **TUTipo**.

Definições de constantes de TUTipo*/.

Definições de implementação do Teste de Unidade do Domínio de **Tipo**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe TUValidade

`#include <testes.h>`

Membros Públicos

- int **executar** ()
  Método para executar o teste.

Atributos Estáticos Públicos

- const static int **SUCESSO** = 0
  *Definições de constantes para reportar resultado do **TUValidade**.*

- const static int **FALHA** = -1

Descrição detalhada

Testes de Unidade da Classe **Validade**

Funções membros

int TUValidade::executar ()

Método para executar o teste.

```
536                         {
537     montar();
538     testarCenarioSucesso();
539     testarCenarioFalha();
540     destruir();
541     return estado;
542 }
```

Atributos

const int TUValidade::FALHA = -1 [static]

const int TUValidade::SUCESSO = 0 [static]

Definições de constantes para reportar resultado do **TUValidade**.

Definições de constantes de TUValidade*/.

Definições de implementação do Teste de Unidade do Domínio **Data** de **Validade**

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**testes.cpp**

Referência da Classe Usuario

`#include <entidades.h>`

Membros Públicos

- void **defineCpf** (const **Cpf** &c)
  Função para definir CPF de Usuário.

- **Cpf pegaCpf** () const
  Função para retornar CPF do Usuário.

- void **defineSenha** (const **Senha** &sn)
  *Função para definir **Senha** do Usuário.*

- **Senha pegaSenha** () const
  *Função para retornar **Senha** do Usuário.*

- void **imprimeUsuario** ()
  Função para imprimir informações de Usuário.

Descrição detalhada

Trabalho desenvolvido por: João Paulo Vaz Mendes - Matrícula: 170002934 Lucas de Moura Quadros - Matrícula: 140150668 Funções da entidade **Usuario**

Funções membros

void Usuario::defineCpf (const Cpf & *c*) [inline]

Função para definir CPF de Usuário.

```
19                                      {
20          this->cpf = c;
21      }
```

void Usuario::defineSenha (const Senha & *sn*) [inline]

Função para definir **Senha** do Usuário.

```
27                                              {
28          this->senha = sn;
29      }
```

void Usuario::imprimeUsuario () [inline]

Função para imprimir informações de Usuário.

```
35                              {
36          std::cout <<"\t== Usuário Definido ==" << std::endl;
37          std::cout << "Cpf: " << this->pegaCpf().pegaCpf() << std::endl;
38          std::cout << "Senha: " << this->pegaSenha().pegaSenha() << std::endl;
```

```
39
40      }
```

Cpf Usuario::pegaCpf () const [inline]

Função para retornar CPF do Usuário.

```
23                              {
24          return cpf;
25      }
```

Senha Usuario::pegaSenha () const [inline]

Função para retornar **Senha** do Usuário.

```
31                                  {
32          return senha;
33      }
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**entidades.h**

Referência da Classe usuario

`#include <usuario.h>`

Membros Públicos

- **usuario** ()
- virtual **~usuario** ()
- unsigned int **Getm_Counter** ()
- void **Setm_Counter** (unsigned int val)

Construtores e Destrutores

usuario::usuario ()

```
 4 {
 5     //ctor
 6 }
```

usuario::~usuario () `[virtual]`

```
 9 {
10     //dtor
11 }
```

Funções membros

unsigned int usuario::Getm_Counter () `[inline]`

```
11 { return m_Counter; }
```

void usuario::Setm_Counter (unsigned int *val*) `[inline]`

```
12 { m_Counter = val; }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**usuario.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**usuario.cpp**

Referência da Classe Validade

Classe do Domínio **Data** de **Validade**.
`#include <dominios.h>`

Membros Públicos

- void **defineValidade** (std::string val)
  Função para definir a data de validade.

- std::string **pegaValidade** ()
  Função para retornar a data de validade.

Descrição detalhada

Classe do Domínio **Data** de **Validade**.

Funções membros

void Validade::defineValidade (std::string *val*)

Função para definir a data de validade.

```
477                                             {
478     validaValidade(val);
479     this->validade = val;
480 }
```

std::string Validade::pegaValidade () `[inline]`

Função para retornar a data de validade.

```
178 {return validade;}
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.h**
- C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/**dominios.cpp**

Arquivos

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/dominios.cpp

```
#include "dominios.h"
#include <sstream>
#include <stdlib.h>
#include <iostream>
```

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/dominios.h

```
#include <string>
#include <stdexcept>
```

Componentes

- class **Data**
- class **Cpf**
- class **Horario**
- class **Codigo**
- class **Preco**
- class **Disponibilidade**
- class **Senha**
- class **CodigoVerificacao**
  Classe do Código de verificação do cartão CVC.
- class **CodigoJogo**
  *Classe do código do **Jogo**.*
- class **CodigoIngresso**
  *Classe do código do **Ingresso**.*
- class **Estado**
  *classe do **Estado***
- class **NumCartao**
  Classe do Domínio Número de Cartão de Crédito.
- class **Validade**
  *Classe do Domínio **Data** de **Validade**.*
- class **Nome**
  *Classe Genérica **Nome**.*
- class **NomeJogo**
  *Classe do domínio **Nome** de **Jogo** que herda da classe **Nome**.*
- class **Cidade**
  *Classe do domínio **Cidade** que herda da classe **Nome**.*
- class **Tipo**
  *Classe do domínio **Tipo**.*

Enumerações

- enum **Id** { **LOCAL** =1, **ESTADUAL**, **NACIONAL**, **INTERNACIONAL** }
  *Definição de tipos enum dos tipos da classe **Tipo**.*

Enumerações

enum Id

Definição de tipos enum dos tipos da classe **Tipo**.

Enumeradores:

LOCAL	
ESTADUAL	
NACIONAL	

INTERNACIONAL	

```
214 {LOCAL=1, ESTADUAL, NACIONAL, INTERNACIONAL};
```

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/entidades.cpp

```
#include "entidades.h"
```

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/entidades.h

```
#include "dominios.h"
#include <iostream>
```

Componentes

- class **Usuario**
- class **Partida**
- class **Ingresso**
- class **CartaoCredito**
- class **Jogo**

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/main.cpp

```
#include <iostream>
#include "entidades.h"
#include "dominios.h"
#include "testes.h"
#include <string>
#include <locale.h>
```

Funções

- int **main** ()

Funções

int main ()

permitir a impressão de caracteres em português

Declaração dos Testes de Unidade dos Domínios

Execução dos Testes de Unidade dos Domínios

Declaração de objetos dos Dompinios

Declaração de objetos das Entidades

definições dos domínios

definições de uma partida

definições de um usuário

Definição de **Jogo**

Definição de **Ingresso**

Definição de Cartão de Crédito

Impressão de domínios

Impressão de **Partida**

Impressão de Usuário

Impressão de **Jogo**

impressão de **Ingresso**

impressão de Cartão de Crédito

```
15 {
16     setlocale(LC_ALL,"portuguese");
17 
18 
20     TUData testeData;
21     TUCpf testeCpf;
22     TUHorario testeHorario;
23     TUCodigo testeCodigo;
24     TUPreco testePreco;
25     TUDisp testeDisp;
26     TUSenha testeSenha;
27     TUCodVer testeCodVer;
28     TUCodJogo testeCodJogo;
29     TUCodIng testeCodIng;
30     TUEstado testeEstado;
31     TUNumCartao testeNumCartao;
32     TUValidade testeValidade;
```

```
   33     TUNomeJogo testeNomeJogo;
   34     TUCidade testeCidade;
   35     TUTipo testeTipo;
   36
   37     cout << endl << endl;
   39     switch(testeData.executar()){
   40         case TUData::SUCESSO: cout << "==  TESTE DE UNIDADE DE DATA: SUCESSO ==" 
<< endl << endl;
   41                                   break;
   42         case TUData::FALHA  : cout << "==  TESTE DE UNIDADE DE DATA: FALHA ==" 
<< endl << endl;
   43                                   break;
   44     }
   45     switch(testeCpf.executar()){
   46         case TUCpf::SUCESSO: cout << "==  TESTE DE UNIDADE DE CPF: SUCESSO ==" 
<< endl << endl;
   47                                   break;
   48         case TUCpf::FALHA  : cout << "==  TESTE DE UNIDADE DE CPF: FALHA ==" << 
endl << endl;
   49                                   break;
   50     }
   51     switch(testeHorario.executar()){
   52         case TUHorario::SUCESSO: cout << "==  TESTE DE UNIDADE DE HORÁRIO: 
SUCESSO ==" << endl << endl;
   53                                   break;
   54         case TUHorario::FALHA  : cout << "==  TESTE DE UNIDADE DE HORÁRIO: FALHA 
==" << endl << endl;
   55                                   break;
   56     }
   57     switch(testeCodigo.executar()){
   58         case TUCodigo::SUCESSO: cout << "==  TESTE DE UNIDADE DE CÓDIGO: SUCESSO 
==" << endl << endl;
   59                                   break;
   60         case TUCodigo::FALHA  : cout << "==  TESTE DE UNIDADE DE CÓDIGO: FALHA 
==" << endl << endl;
   61                                   break;
   62     }
   63     switch(testePreco.executar()){
   64         case TUPreco::SUCESSO: cout << "==  TESTE DE UNIDADE DE PREÇO: SUCESSO 
==" << endl << endl;
   65                                   break;
   66         case TUPreco::FALHA  : cout << "==  TESTE DE UNIDADE DE PREÇO: FALHA ==" 
<< endl << endl;
   67                                   break;
   68     }
   69     switch(testeDisp.executar()){
   70         case TUDisp::SUCESSO: cout << "==  TESTE DE UNIDADE DE DISPONIBILIDADE: 
SUCESSO ==" << endl << endl;
   71                                   break;
   72         case TUDisp::FALHA  : cout << "==  TESTE DE UNIDADE DE DISPONIBILIDADE: 
FALHA ==" << endl << endl;
   73                                   break;
   74     }
   75     switch(testeSenha.executar()){
   76         case TUSenha::SUCESSO: cout << "==  TESTE DE UNIDADE DE SENHA: SUCESSO 
==" << endl << endl;
   77                                   break;
   78         case TUSenha::FALHA  : cout << "==  TESTE DE UNIDADE DE SENHA: FALHA ==" 
<< endl << endl;
   79                                   break;
   80     }
   81     switch(testeCodVer.executar()){
   82         case TUCodVer::SUCESSO: cout << "==  TESTE DE UNIDADE DE CÓDIGO DE 
VERIFICAÇÃO: SUCESSO ==" << endl << endl;
   83                                   break;
   84         case TUCodVer::FALHA  : cout << "==  TESTE DE UNIDADE DE CÓDIGO DE 
VERIFICAÇÃO: FALHA ==" << endl << endl;
   85                                   break;
   86     }
   87     switch(testeCodJogo.executar()){
   88         case TUCodJogo::SUCESSO: cout << "==  TESTE DE UNIDADE DE CÓDIGO DE JOGO: 
SUCESSO ==" << endl << endl;
   89                                   break;
   90         case TUCodJogo::FALHA  : cout << "==  TESTE DE UNIDADE DE CÓDIGO DE JOGO: 
FALHA ==" << endl << endl;
   91                                   break;
   92     }
```

```
   93     switch(testeCodIng.executar()){
   94         case TUCodIng::SUCESSO: cout << "==  TESTE DE UNIDADE DE CÓDIGO DE
INGRESSO: SUCESSO ==" << endl << endl;
   95                                 break;
   96         case TUCodIng::FALHA  : cout << "==  TESTE DE UNIDADE DE CÓDIGO DE
INGRESSO: FALHA ==" << endl << endl;
   97                                 break;
   98     }
   99     switch(testeEstado.executar()){
  100         case TUEstado::SUCESSO: cout << "==  TESTE DE UNIDADE DE ESTADO: SUCESSO
==" << endl << endl;
  101                                 break;
  102         case TUEstado::FALHA  : cout << "==  TESTE DE UNIDADE DE ESTADO: FALHA
==" << endl << endl;
  103                                 break;
  104     }
  105     switch(testeNumCartao.executar()){
  106         case TUNumCartao::SUCESSO: cout << "==  TESTE DE UNIDADE DE NÚMERO DO
CARTÃO DE CRÉDITO: SUCESSO ==" << endl << endl;
  107                                 break;
  108         case TUNumCartao::FALHA  : cout << "==  TESTE DE UNIDADE DE NÚMERO DO
CARTÃO DE CRÉDITO: FALHA ==" << endl << endl;
  109                                 break;
  110     }
  111     switch(testeValidade.executar()){
  112         case TUValidade::SUCESSO: cout << "==  TESTE DE UNIDADE DA DATA DE
VALIDADE: SUCESSO ==" << endl << endl;
  113                                 break;
  114         case TUValidade::FALHA  : cout << "==  TESTE DE UNIDADE DA DATA DE
VALIDADE: FALHA ==" << endl << endl;
  115                                 break;
  116     }
  117     switch(testeNomeJogo.executar()){
  118         case TUNomeJogo::SUCESSO: cout << "==  TESTE DE UNIDADE DO NOME DO JOGO:
SUCESSO ==" << endl << endl;
  119                                 break;
  120         case TUNomeJogo::FALHA  : cout << "==  TESTE DE UNIDADE DO NOME DO JOGO:
FALHA ==" << endl << endl;
  121                                 break;
  122     }
  123     switch(testeCidade.executar()){
  124         case TUCidade::SUCESSO: cout << "==  TESTE DE UNIDADE DA CIDADE: SUCESSO
==" << endl << endl;
  125                                 break;
  126         case TUCidade::FALHA  : cout << "==  TESTE DE UNIDADE DA CIDADE: FALHA
==" << endl << endl;
  127                                 break;
  128     }
  129     switch(testeTipo.executar()){
  130         case TUTipo::SUCESSO: cout << "==  TESTE DE UNIDADE DO TIPO: SUCESSO =="
<< endl << endl;
  131                                 break;
  132         case TUTipo::FALHA  : cout << "==  TESTE DE UNIDADE DO TIPO: FALHA =="
<< endl << endl;
  133                                 break;
  134     }
  135
  137     Data d1;
  138     Cpf c1;
  139     Horario h1;
  140     Codigo cd1;
  141     Preco p1;
  142     Disponibilidade dp1;
  143     Senha s1;
  144     CodigoVerificacao cv1;
  145     CodigoJogo cj1;
  146     CodigoIngresso ci1;
  147     Estado e1;
  148     NumCartao nc1;
  149     Validade v1;
  150     NomeJogo nj1;
  151     Cidade cdd1;
  152     Tipo t1;
  153
  155     Partida part;
  156     Usuario usr;
  157     Jogo jog;
```

```
158     Ingresso ing;
159     CartaoCredito cart;
160 
161     string cpf = "23261322110";
162     string cod = "01030";
163     string dt = "23/10/1917";
164     string hrr = "21:47";
165     string sn = "11aA1";
166     string cvc = "123";
167     string cjg = "460";
168     string cdi = "00539";
169     string est = "AP";
170     string num = "4545419514929815";
171     string val = "01/23";
172     string njo = "Corinthians x Botafogo";
173     string cid = "Brasilia";
174     int tip = 1;
175     double prc=30.05;
176     int dsp=30;
177 
179     c1.defineCpf(cpf);
180     cd1.defineCodigo(cod);
181     p1.definePreco(prc);
182     dp1.defineDisp(dsp);
183     d1.defineData(dt);
184     h1.defineHorario(hrr);
185     s1.defineSenha(sn);
186     cv1.defineCodVer(cvc);
187     cj1.defineCodJogo(cjg);
188     ci1.defineCodIng(cdi);
189     e1.defineEstado(est);
190     nc1.defineNumCartao(num);
191     v1.defineValidade(val);
192     nj1.defineNome(njo);
193     cdd1.defineNome(cid);
194     t1.defineTipo(tip);
195 
197     part.defineCodigo(cd1);
198     part.defineData(d1);
199     part.defineDisp(dp1);
200     part.defineHorario(h1);
201     part.definePreco(p1);
202 
204     usr.defineCpf(c1);
205     usr.defineSenha(s1);
206 
208     jog.defineCodJogo(cj1);
209     jog.defineNomeJogo(nj1);
210     jog.defineCidade(cdd1);
211     jog.defineEstado(e1);
212     jog.defineTipo(t1);
213 
215     ing.defineCodIng(ci1);
216 
218     cart.defineNumCartao(nc1);
219     cart.defineCodVer(cv1);
220     cart.defineValidade(v1);
221 
223     cout << "CPF: " << c1.pegaCpf() << endl;
224     cout << "Data: " << d1.viraString() << endl;
225     cout << "Horário: " << h1.viraString() << endl;
226     cout << "Código: " << cd1.pegaCodigo() << endl;
227     cout << "Preço: " << p1.pegaPreco() << endl;
228     cout << "Disponibilidade: " << dp1.pegaDisp() << endl;
229     cout << "Senha: " << s1.pegaSenha() << endl;
230     cout << "Código de Verificação do Cartão: " << cv1.pegaCodigo() << endl;
231     cout << "Código do Jogo: " << cj1.pegaCodigo() << endl;
232     cout << "Código do Ingresso: " << ci1.pegaCodigo() << endl;
233     cout << "Estado: " << e1.pegaEstado() << endl;
234     cout << "Número do Cartão de Crédito: " << nc1.pegaNumCartao() << endl;
235     cout << "Validade: " << v1.pegaValidade() << endl;
236     cout << "Nome do Jogo: " << nj1.pegaNome() << endl;
237     cout << "Cidade: " << cdd1.pegaNome() << endl;
238     cout << "Tipo: " << t1.stringTipo() << endl;
239 
240     cout << endl << endl;
241 
```

```
  243     part.imprimePartida();
  244
  246     usr.imprimeUsuario();
  247
  249     jog.imprimeJogo();
  250
  252     ing.imprimeIngresso();
  253
  255     cart.imprimeIngresso();
  256
  257     //LEMBRETES DE CÓDIGO
  258     //cout << endl << endl << "==  FAZER VALIDAÇÃO DE DATA E HORÁRIO  ==" << endl
<< endl;
  259
  260     return 0;
  261 }
```

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/README.md

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/testes.cpp

```
#include "testes.h"
```

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/testes.h

```
#include "dominios.h"
#include <string>
```

Componentes

- class **TUData**
- class **TUCpf**
- class **TUHorario**
- class **TUCodigo**
- class **TUPreco**
- class **TUDisp**
- class **TUSenha**
- class **TUCodVer**
- class **TUCodJogo**
- class **TUCodIng**
- class **TUEstado**
- class **TUNumCartao**
- class **TUValidade**
- class **TUNomeJogo**
- class **TUCidade**
- class **TUTipo**

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/usuario.cpp

```
#include "usuario.h"
```

Referência do Arquivo C:/Users/Joao Paulo/Documents/TP1/Trabalho_TP1/usuario.h

Componentes

- class **usuario**

Sumário

INDEX