PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Bacharelado em Sistemas de Informação

Andrew de Oliveira Duchini (537983)

Cristóvão Olegário de Castro (536584)

Pedro Lucas Duarte Faria (536589)

Samara Luiza da Silva (536592)

**RELATÓRIO LAB4**

Belo Horizonte

2015

Andrew de Oliveira Duchini (537983)

Cristóvão Olegário de Castro (536584)

Pedro Lucas Duarte Faria (536589)

Samara Luiza da Silva (536592)

**RELATÓRIO LAB4**

Relatório apresentado à disciplina Programação Orientado a Objetos do curso de Graduação em Sistemas de Informação da Pontifícia Universidade Católica de Minas Gerais. Orientador: Professor Paulo Cesar do Amaral

Belo Horizonte

2015

**SUMÁRIO**

[1. EXERCÍCIO 3.1 3](#_Toc435375958)

[2. EXERCÍCIO 3.2 6](#_Toc435375959)

[3. EXERCÍCIO 3.3 10](#_Toc435375960)

[4. EXERCÍCIO 5.1 13](#_Toc435375961)

[5. EXERCÍCIO 5.2 18](#_Toc435375962)

[6. EXERCÍCIO 5.3 20](#_Toc435375963)

[7. EXERCÍCIO 5.4 25](#_Toc435375964)

[8. EXERCÍCIO 5.5 31](#_Toc435375965)

[9. EXERCÍCIO 6.1 35](#_Toc435375966)

[10. EXERCÍCIO 6.2 41](#_Toc435375967)

[11. EXERCÍCIO 6.3 51](#_Toc435375968)

[12. EXERCÍCIO 6.4 61](#_Toc435375969)

[13. EXERCÍCO 6.5 63](#_Toc435375970)

# EXERCÍCIO 3.1

Converter temperaturas de Celsius para Fahrenheit e de Fahrenheit para Celsius. Apresentar inicialmente na tela um menu com as seguintes opções:

* Converter de Celsius para Fahrenheit
* Converter de Fahrenheit para Celsius

//

// NOME DO PROGRAMA: Conversor

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 29/10/2015

// DESCRIÇÃO: Utilizando os conceitos de formulários, foi implementada um conversor

// que converte Graus C° para F°

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

txtCelsius.Enabled = false;

txtFahr.Enabled = false;

btnConverter.Enabled = false;

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

txtCelsius.Enabled = true;

txtFahr.Enabled = false;

btnConverter.Enabled = true;

txtFahr.Clear();

}

private void rbFahr\_CheckedChanged(object sender, EventArgs e)

{

txtFahr.Enabled = true;

txtCelsius.Enabled = false;

btnConverter.Enabled = true;

txtCelsius.Clear();

}

private void btnConverter\_Click(object sender, EventArgs e)

{

double temp;

if (rbCelsius.Checked == true)

{

if (txtCelsius.Text != "")

{

temp = double.Parse(txtCelsius.Text);

string msg = "Conversão de Celsius para Fahrenheit!\n" + temp + "°C = " + Math.Round(CelsiusToFahrenheit(temp), 2) + "°F";

MessageBox.Show(null, msg, "Resultado da Conversão", MessageBoxButtons.OK);

limpar();

}

else

{

MessageBox.Show("Favor preencher o campo!", "Aviso!");

}

}

else if(rbFahr.Checked == true)

{

if (txtFahr.Text != "")

{

temp = double.Parse(txtFahr.Text);

string msg = "Conversão de Fahrenheit para Celsius!\n" + temp + "°F = " + Math.Round(FahrenheitToCelcius(temp),2) + "°C";

MessageBox.Show(null, msg, "Resultado da Conversão", MessageBoxButtons.OK);

limpar();

}

else

{

MessageBox.Show("Favor preencher o campo!","Aviso!");

}

}

}

private void limpar()

{

txtCelsius.Clear();

rbCelsius.Checked = false;

txtCelsius.Enabled = false;

txtFahr.Clear();

rbFahr.Checked = false;

txtFahr.Enabled = false;

btnConverter.Enabled = false;

}

private double CelsiusToFahrenheit(double temperatura)

{

double conversao;

conversao = (temperatura \* 1.8) + 32;

return conversao;

}

private double FahrenheitToCelcius(double temperatura)

{

double conversao;

conversao = (temperatura - 32) / 1.8;

return conversao;

}

}

# EXERCÍCIO 3.2

Implementar uma calculadora que efetue as operações aritméticas básicas (+, -, x e /) com números reais.

//

// NOME DO PROGRAMA: Calculadora

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 29/10/2015

// DESCRIÇÃO: Utilizando os conceitos de formulários, foi implementada uma calculadora

// com as quatros operações básicas

class Calcula

{

public static string Acheoperacao(double a, double b, char op)

{

switch (op)

{

case '+': return (Convert.ToString(a + b));

case '-': return (Convert.ToString(a - b));

case '\*': return (Convert.ToString(a \* b));

case '/':

if (b==0)

{

return ("Erro!!");

}

else

{

return (Convert.ToString(a / b));

}

default: return (" ");

}

}

}

public partial class Form1 : Form

{

double a, b;

string bla;

char op = 'f';

public Form1()

{

InitializeComponent();

}

private void button15\_Click(object sender, EventArgs e)

{

a = double.Parse(lb\_displaynumbers.Text);

lb\_displayresult.Text = (lb\_displaynumbers.Text + "-");

lb\_displaynumbers.Text = "";

op = '-';

}

private void btn\_1\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "1");

}

private void btn\_2\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "2");

}

private void btn\_3\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "3");

}

private void btn\_4\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "4");

}

private void btn\_5\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "5");

}

private void btn\_6\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "6");

}

private void btn\_7\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "7");

}

private void btn\_8\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "8");

}

private void btn\_9\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "9");

}

private void btn\_virgula\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + ",");

}

private void btn\_0\_Click(object sender, EventArgs e)

{

lb\_displaynumbers.Text = (lb\_displaynumbers.Text + "0");

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void btn\_soma\_Click(object sender, EventArgs e)

{

a = double.Parse(lb\_displaynumbers.Text);

lb\_displayresult.Text = (lb\_displaynumbers.Text + "+");

lb\_displaynumbers.Text = "";

op = '+';

}

private void btn\_multi\_Click(object sender, EventArgs e)

{

a = double.Parse(lb\_displaynumbers.Text);

lb\_displayresult.Text = (lb\_displaynumbers.Text + "\*");

lb\_displaynumbers.Text = "";

op = '\*';

}

private void btn\_divisao\_Click(object sender, EventArgs e)

{

a = double.Parse(lb\_displaynumbers.Text);

lb\_displayresult.Text = (lb\_displaynumbers.Text + "/");

lb\_displaynumbers.Text = "";

op = '/';

}

private void btn\_equal\_Click(object sender, EventArgs e)

{

b = double.Parse(lb\_displaynumbers.Text);

lb\_displayresult.Text = (lb\_displayresult.Text + b);

bla = lb\_displayresult.Text;

lb\_displaynumbers.Text = "";

lb\_displayresult.Text = (bla + " = " + Calcula.Acheoperacao(a, b, op));

}

}

# EXERCÍCIO 3.3

Implementar a classe Conta com as mesmas especificações do exercício da seção 3.9 do LAB1.

//

// NOME DO PROGRAMA: Conta

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 29/10/2015

// DESCRIÇÃO: O programa implementa um formulário, nele pode-se fazer depositos e saques

public partial class Deposito : Form

{

public Deposito()

{

InitializeComponent();

}

private void btnDepositar\_Click(object sender, EventArgs e)

{

if (txtDeposito.Text == "")

{

MessageBox.Show("Digite um valor antes de depositar!", "OPA!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

else

{

double valor = double.Parse(txtDeposito.Text);

Form1.saldo += valor;

MessageBox.Show("Deposito realizado com sucesso!", "Sucesso!", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

}

}

public partial class Form1 : Form

{

public static double saldo = 0;

public Form1()

{

InitializeComponent();

}

private void btnDepositar\_Click(object sender, EventArgs e)

{

Deposito a = new Deposito();

a.Show();

}

private void btnSaldo\_Click(object sender, EventArgs e)

{

MessageBox.Show("Saldo da Conta: " + saldo);

}

private void btnSacar\_Click(object sender, EventArgs e)

{

FormSaque a = new FormSaque();

a.Show();

}

}

public partial class FormSaque : Form

{

public FormSaque()

{

InitializeComponent();

}

private void btnSacar\_Click(object sender, EventArgs e)

{

if (txtSaque.Text == "")

{

MessageBox.Show("Digite um valor antes de sacar!", "OPA!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

else

{

double valor = double.Parse(txtSaque.Text);

if (Form1.saldo >= valor)

{

Form1.saldo -= valor;

MessageBox.Show("Saque realizado com sucesso!", "Sucesso!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Saldo indisponível!", "OPA!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

this.Close();

}

}

}

# EXERCÍCIO 5.1

Implemente uma classe Veículo. O veículo é composto por várias partes: um motor, um tanque de combustível e 4 pneus. O funcionamento do veículo depende das suas partes, da seguinte forma:

Motor

• Possui uma potência (em hp), uma taxa fixa de consumo (em km/litro) e um tanque de combustível (ver abaixo).

• Possui um comando para avançar uma determinada quantidade de km. Se a pressão de mais de um pneu (ver abaixo) estiver abaixo de 20 lb, o consumo do veículo aumenta em 30%. Se não houver combustível suficiente, o veículo avança até o tanque esvaziar.

Tanque de Combustível

• Possui uma determinada capacidade e quantidade atual de combustível (ambos em litros).

• Pode ser abastecido com uma certa quantidade de combustível, limitado à sua capacidade máxima.

Pneu

• Cada pneu possui uma determina pressão (em lb).

• Pode ser calibrado com determinada pressão informada (positiva ou negativa, sendo somada à atual).

Implemente as classes acima, usando composição quando necessário. Modularize o código. Lembre-se de implementar gets e sets necessários, bem como construtores adequados. No programa principal, faça as seguintes operações:

• Instancie um veículo cujo motor tem 71 hp, consumo de 12 km/litro, tanque com capacidade para 50 litros, pneus dianteiros com 27 lb e traseiros com 23 lb.

• Abasteça o tanque com 30 litros.

• Exiba na tela as informações sobre cada componente do veículo.

• Avance 300 km.

• Reduza a pressão do pneu traseiro esquerdo para 17 lb.

• Avance 100 km.

• Reduza a pressão do pneu dianteiro direito para 18 lb

. • Abasteça mais 10 litros.

• Avance 200 km. A cada operação de movimento, mostre na tela a distância percorrida e a quantidade de combustível restante.

//

// NOME DO PROGRAMA: Veiculo

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 29/10/2015

// DESCRIÇÃO: Utilizando os conceitos de composição e agregação, foi implementado

// o programa que contém uma class Veiculo que é composta por motor, tanque e pneus

namespace Exercicio5\_1

{

class Veiculo

{

public Motor M = new Motor();

public Tanque T = new Tanque();

public Pneu PneuDianteiroEsq;

public Pneu PneuDianteiroDir;

public Pneu PneuTraseiroEsq;

public Pneu PneuTraseiroDir;

public Veiculo(Motor M, Tanque T, Pneu PneuDianteiroEsq, Pneu PneuDianteiroDir, Pneu PneuTraseiroEsq, Pneu PneuTraseiroDir)

{

this.M = M;

this.T = T;

this.PneuDianteiroEsq = PneuDianteiroEsq;

this.PneuDianteiroDir = PneuDianteiroDir;

this.PneuTraseiroEsq = PneuTraseiroEsq;

this.PneuTraseiroDir = PneuTraseiroDir;

}

}

class Motor

{

float potencia;

float consumo;

Tanque Combustivel;

Pneu PneuDianteiroEsq;

Pneu PneuDianteiroDir;

Pneu PneuTraseiroEsq;

Pneu PneuTraseiroDir;

public Motor(float potencia, float consumo, Tanque Combustivel)

{

this.potencia = potencia;

this.consumo = consumo;

this.Combustivel = Combustivel;

}

public void avancaKilometragem(float kilometro)

{

float resultado = 0;

if( (PneuDianteiroDir.pressao > 20 && PneuDianteiroEsq.pressao > 20) ||

(PneuTraseiroDir.pressao > 20 && PneuTraseiroEsq.pressao > 20) ||

(PneuDianteiroEsq.pressao > 20 && PneuTraseiroEsq.pressao > 20) ||

(PneuDianteiroDir.pressao > 20 && PneuTraseiroDir.pressao > 20)||

(PneuDianteiroDir.pressao > 20 && PneuTraseiroEsq.pressao > 20)||

(PneuDianteiroEsq.pressao > 20 && PneuTraseiroDir.pressao > 20))

{

consumo = consumo \* 1.3f;

}

resultado = Combustivel.QtdAtual \* consumo;

if((resultado = resultado - kilometro) < 0)

{

Combustivel.QtdAtual = 0;

}

else

{

resultado = kilometro / consumo;

Combustivel.QtdAtual = resultado;

}

}

}

class Tanque

{

float capacidade;

float qtdAtual;

float capacidadeMax;

public Tanque(float capacidade, float qtdAtual, float capacidadeMax)

{

this.capacidade = capacidade;

this.qtdAtual = qtdAtual;

this.capacidadeMax = capacidadeMax;

}

public void abastece(float qtd)

{

if((qtdAtual += qtd) > capacidadeMax)

{

qtdAtual = capacidadeMax;

}

else

{

qtdAtual += qtd;

}

}

public float QtdAtual

{

get{return qtdAtual;}

set{qtdAtual = value;}

}

}

class Pneu

{

public float pressao;

public Pneu(float pressao)

{

this.pressao = pressao;

}

public float Calibra

{

get{return pressao;}

set{pressao += value;}

}

}

class Program

{

static void Main()

{

float potencia = 71;

float consumo = 12;

float capacidadeMax = 50;

float capacidade = capacidadeMax;

/\*

Console.WriteLine("Digite a potencia do motor: ");

potencia = float.parse(Console.ReadLine());

Console.WriteLine("Digite o consumo do motor: ");

consumo = float.parse(Console.ReadLine());

Console.WriteLine("Digite a capacidade maxima do tanque: ");

capacidadeMax = float.parse(Console.ReadLine());

\*/

Tanque T = new Tanque(capacidade, 0, capacidadeMax);

Motor M = new Motor(potencia, consumo, T);

Pneu P1 = new Pneu(27);

Pneu P2 = new Pneu(27);

Pneu P3 = new Pneu(23);

Pneu P4 = new Pneu(23);

Veiculo V = new Veiculo(M, T, P1, P2, P3, P4);

V.T.abastece(50);

V.M.avancaKilometragem(300);

V.PneuTraseiroEsq.Calibra = -17;

V.M.avancaKilometragem(100);

V.PneuDianteiroEsq.Calibra = -18;

V.T.abastece(10);

try {

V.M.avancaKilometragem(200);

}

catch {}

Console.ReadKey();

}

}

}

# EXERCÍCIO 5.2

Crie a classe ContaDePoupanca. Use a variável static taxaDeJurosAnual para armazenar a taxa de juros de todos os correntistas. Cada objeto da classe contém uma variável de instância private saldoPoupanca, indicando a quantidade que o poupador possui atualmente depositada. Forneça o método CalcularJuroMensal para calcular os juros mensais, multiplicando saldoPoupanca por taxaDeJurosAnual, dividindo por 12; esses juros 9 devem ser acrescidos a saldoPoupanca. Forneça um método static AlterarTaxaDeJuros que configure taxaDeJurosAnual para um novo valor. Escreva um programa driver para testar a classe ContaDePoupanca. Instancie dois objetos ContaDePoupanca, poupador1 e poupador2, com saldos de R$2000,00 e R$3000,00, respectivamente. Configure taxaDeJurosAnual como 4% e, em seguida, calcule os juros mensais e imprima os novos saldos para cada um dos poupadores. Depois, configure taxaDeJurosAnual como 5%, calcule os juros do próximo mês e imprima os novos saldos para cada um dos poupadores.

//

// NOME DO PROGRAMA: ContaDePoupança

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 29/10/2015

// DESCRIÇÃO: O programa implementa uma class com métodos de instacia e métodos estaticos

// Utiliza um programa teste para testar sua funcionalidade

namespace ConsoleApplication1

{

class Program

{

class ContaDePoupança

{

// Atributos

static double taxadeJurosAnual = 0.04;

private double saldoPoupanca;

//Construtores

public ContaDePoupança() {}

public ContaDePoupança (double taxa, double saldo)

{

taxadeJurosAnual = taxa;

saldoPoupanca = saldo;

}

// Propiedades

public double SaldoPoupaca

{

get { return saldoPoupanca;}

set { saldoPoupanca = value;}

}

public double Taxa

{

get { return taxadeJurosAnual; }

set { taxadeJurosAnual = value; }

}

// Metodos

public void CalcularJuroMensal ()

{

saldoPoupanca = saldoPoupanca + ((saldoPoupanca \* taxadeJurosAnual) / 12.0);

}

public static void AlterarTaxaDeJuros(double novo)

{

taxadeJurosAnual = novo;

}

}

static void Main(string[] args)

{

Console.Write("\tProgramadores: ");

Console.WriteLine("\tAndrew de Oliveira Duchini (537983)");

Console.WriteLine("\tCristóvão Olegário de Castro (536584)");

Console.WriteLine("\tPedro Lucas Duarte Faria(536589)");

Console.WriteLine("\tSamara Luiza da Silva(536592)");

ContaDePoupança poudador1 = new ContaDePoupança(0.04, 2000);

ContaDePoupança poudador2 = new ContaDePoupança(0.04, 3000);

poudador1.CalcularJuroMensal();

poudador2.CalcularJuroMensal();

Console.WriteLine("Taxa de 4%");

Console.WriteLine("Novo saldo do poupador 1: " + poudador1.SaldoPoupaca);

Console.WriteLine("Novo saldo do poupador 2: " + poudador2.SaldoPoupaca);

ContaDePoupança.AlterarTaxaDeJuros(0.05);

poudador1.CalcularJuroMensal();

poudador2.CalcularJuroMensal();

Console.WriteLine("Taxa de 5%");

Console.WriteLine("Novo saldo do poupador 1: " + poudador1.SaldoPoupaca);

Console.WriteLine("Novo saldo do poupador 2: " + poudador2.SaldoPoupaca);

Console.ReadKey();

}

# EXERCÍCIO 5.3

No sistema de um banco algumas informações dos clientes como nome e código precisam ser armazenadas. Crie uma classe para modelar os objetos que representarão os clientes.

Os bancos oferecem aos clientes a possibilidade de obter um cartão de crédito que pode ser utilizados para fazer compras. Um cartão de crédito possui um número e uma data de validade. Crie uma classe para modelar os objetos que representarão os cartões de crédito

Defina um vínculo entre a classe Cliente e a classe CartaoDeCredito. Para isso deve-se alterar a classe CartaoDeCredito.

Teste o relacionamento entre clientes e cartões de crédito. Escreva o código a seguir no método main()

// Criar alguns objetos

Cliente c = new Cliente();

CartaoDeCredito cdc = new CartaoDeCredito();

// Carregar alguns dados

c.nome = "José da Silva";

cdc.numero = 111111;

// Ligar os objetos (liga o objeto CartãodeCredito cdc ao objeto Cliente c)

cdc.cliente = c;

As agências bancárias possuem um número. Crie uma classe para modelar as agências.

As contas do banco possuem saldo e estão vinculadas a uma agência. Crie uma classe para modelar as contas e estabeleça um vínculo com a classe AGENCIA.

Teste o relacionamento entre contas e agências. Escreva o código abaixo no método main().

// Criar alguns objetos

Agencia a = new Agencia();

Conta c = new Conta();

// Carregar alguns dados

a.numero = 178;

c.saldo = 1000.0;

// Ligar os objetos (liga o objeto Conta c ao objeto Agencia a)

c.agencia = a;

Acrescente alguns métodos na classe CONTA para definir as lógicas de depositar, sacar e imprimir extrato. Testar os métodos da classe CONTA. Criar uma classe para modelar as faturas dos cartões de crédito. Teste a classe FATURA. Alterar a classe CartaodeCredito para incluir as faturas de um cartão. Acrescentar um método à classe CONTA para transferir um valor para uma conta de destino.

//

// NOME DO PROGRAMA: SistemaBancario

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 29/10/2015

// DESCRIÇÃO: Utilizado os conceitos de relacionamento entre classes, foi implementado o programa

// que simula o sistema bancario com classes como conta, fatura, cartão\_de\_credito, cliente, etc.

// A classes testes

namespace SistemaBancario

{

class Program

{

static void Main(string[] args)

{

Console.Write("\tProgramadores: ");

Console.WriteLine("\tAndrew de Oliveira Duchini (537983)");

Console.WriteLine("\tCristóvão Olegário de Castro (536584)");

Console.WriteLine("\tPedro Lucas Duarte Faria(536589)");

Console.WriteLine("\tSamara Luiza da Silva(536592)");

// Teste relacionamento entre clientes e cartão de crédito

Cliente c = new Cliente();

CartaoDeCredito cdc = new CartaoDeCredito();

c.Nome = "José da Silva";

cdc.Numero = 111111;

cdc.cliente = c;

//Teste relacionamento entre contas e agencias

Agencia a = new Agencia();

Conta conta = new Conta();

a.numero = 178;

conta.saldo = 1000.0;

conta.agencia = a;

//Teste dos métodos da classe Conta

conta.deposita(1000);

conta.imprimeExtrato();

conta.saca(100);

conta.imprimeExtrato();

//Teste da classe FATURA

Fatura f = new Fatura();

f.adiciona(100);

f.adiciona(200);

f.imprimeDados();

double multa = f.calculaMulta();

Console.ReadKey();

}

}

}

class Fatura

{

public double total;

// Adiciona o valor ao total da fatura

public void adiciona(double valor)

{

total += valor;

}

// Calcula multa de 2% sobre o total da fatura

public double calculaMulta()

{

double res;

res = 0.02 \* total;

return res;

}

// Imprime o total da fatura

public void imprimeDados()

{

Console.WriteLine("Total da Fatura: "+total);

}

}

class Cliente

{

private string nome;

private int codigo;

public string Nome

{

get { return nome; }

set { nome = value; }

}

public int Codigo

{

get { return codigo; }

set { codigo = value; }

}

}

class Conta

{

public double saldo;

public Agencia agencia;

public void deposita(double valor)

{

saldo += valor;

}

public void saca(double valor)

{

if(valor <= saldo)

{

saldo -= valor;

}

}

public void imprimeExtrato()

{

Console.WriteLine("Saldo: " + saldo);

}

public void transfere(Conta destino, double valor)

{

if(valor <= saldo)

{

destino.saldo = valor;

}

else

{

Console.WriteLine("Saldo insulficiente!!");

}

}

}

class CartaoDeCredito

{

private int numero;

private string dataDeValidade;

public Cliente cliente;

//Incluindo faturas do cartão de credito

Fatura[] fatura;

public string DataDeValidade

{

get { return dataDeValidade; }

set { dataDeValidade = value; }

}

public int Numero

{

get { return numero; }

set { numero = value; }

}

}

class Agencia

{

public int numero;

}

# EXERCÍCIO 5.4

Fazer um programa em C#, orientado por objetos que utilize obrigatoriamente os conceitos de herança e composição, para implementar uma agenda para armazenar os dados de contatos de pessoas, sejam elas físicas ou jurídicas.

Os dados de pessoas físicas e jurídicas são armazenados com os seguintes itens especificados a seguir:

Pessoas Físicas

● Nome

● Endereço

● Data de nascimento

● Sexo

● CPF

Pessoas Jurídicas

● Nome

● Endereço

● Data de estabelecimento

● CNPJ

A agenda tem vários contatos: AGREGAÇÃO / COMPOSIÇÃO Um contato TEM uma pessoa: associação contato → pessoa

//

// NOME DO PROGRAMA: Agenda

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 29/10/2015

// DESCRIÇÃO: Seguindo os conceitos de relacionamentos entre classes o programa

// Cria uma classe Agenda que é agregado a contatos que por sua vez é agregada a pessoas.

// Permite que o usuário entre com contatos, consulte um contato e imprima a relação de contatos

namespace Agenda

{

class Agenda

{

// Atributos

public static int contador = 0;

public static List<Contato> contatos = new List<Contato>();

// Métodos

static public void Addcontato (Contato cont)

{

contador++;

Console.WriteLine("\nContato nº" + contador);

contatos.Add(cont);

}

}

struct Date

{

public int dia;

public int mes;

public int ano;

}

class Contato

{

Pessoa contato;

public Contato(Pessoa a)

{

this.contato = a;

}

public void Imprime()

{

contato.ImprimeDados();

}

}

class Pessoa

{

protected string nome;

protected string endereço;

protected Date data = new Date();

public virtual void ImprimeDados()

{

Console.WriteLine("NOME: "+nome);

Console.WriteLine("ENDEREÇO: "+endereço);

Console.WriteLine("DATA: "+data.dia+"/"+data.mes+"/"+data.ano);

}

}

class PessoaFisica : Pessoa

{

private char sexo;

private string cpf;

public PessoaFisica (string name, string end, Date data, char sexo, string CPF)

{

this.nome = name;

this.endereço = end;

this.data = data;

this.cpf = CPF;

this.sexo = sexo;

}

public override void ImprimeDados()

{

base.ImprimeDados();

Console.WriteLine("CPF: " + cpf);

Console.WriteLine("SEXO: " + sexo);

Console.WriteLine();

}

}

class PessoaJuridica : Pessoa

{

private string CNPJ;

public PessoaJuridica (string name, string end, Date data, string CNPJ)

{

this.nome = name;

this.endereço = end;

this.data = data;

this.CNPJ = CNPJ;

}

public override void ImprimeDados()

{

base.ImprimeDados();

Console.WriteLine("CNPJ: " + CNPJ);

Console.WriteLine();

}

}

class Program

{

static void Cadastro()

{

int op;

string nome;

string endereço;

Date data = new Date();

char sexo;

string cpf;

Console.WriteLine("1 - Pessoa Fisica");

Console.WriteLine("2 - Pessoa Juridica");

op = int.Parse(Console.ReadLine());

if (op==1)

{

Console.Write("NOME: ");

nome = Console.ReadLine();

Console.Write("ENDEREÇO: ");

endereço = Console.ReadLine();

Console.Write("DIA: ");

data.dia = int.Parse(Console.ReadLine());

Console.Write("MÊS: ");

data.mes = int.Parse(Console.ReadLine());

Console.Write("ANO: ");

data.ano = int.Parse(Console.ReadLine());

Console.Write("SEXO: ");

sexo = char.Parse(Console.ReadLine());

Console.Write("CPF: ");

cpf = Console.ReadLine();

PessoaFisica a = new PessoaFisica(nome, endereço, data, sexo, cpf);

Contato b = new Contato(a);

Agenda.Addcontato(b);

}

else if (op==2)

{

Console.Write("NOME: ");

nome = Console.ReadLine();

Console.Write("ENDEREÇO: ");

endereço = Console.ReadLine();

Console.Write("DIA: ");

data.dia = int.Parse(Console.ReadLine());

Console.Write("MÊS: ");

data.mes = int.Parse(Console.ReadLine());

Console.Write("ANO: ");

data.ano = int.Parse(Console.ReadLine());

Console.Write("CNPJ: ");

cpf = Console.ReadLine();

PessoaJuridica a = new PessoaJuridica(nome, endereço, data, cpf);

Contato b = new Contato(a);

Agenda.Addcontato(b);

}

else

{

Console.WriteLine("Digite uma opção válida!");

}

}

static void Consulta()

{

int aux = 0;

Console.Write("Digite o número do contato: ");

aux = int.Parse(Console.ReadLine());

Agenda.contatos[aux - 1].Imprime();

}

static void Main(string[] args)

{

Console.Write("\tProgramadores: ");

Console.WriteLine("\tAndrew de Oliveira Duchini (537983)");

Console.WriteLine("\tCristóvão Olegário de Castro (536584)");

Console.WriteLine("\tPedro Lucas Duarte Faria(536589)");

Console.WriteLine("\tSamara Luiza da Silva(536592)");

Console.WriteLine();

int op = 0;

do

{

Console.WriteLine("1 - Adcionar contato");

Console.WriteLine("2 - Consultar contato");

Console.WriteLine("3 - Imprimir contatos");

Console.WriteLine("4 - Sair");

Console.WriteLine();

op = int.Parse(Console.ReadLine());

switch (op)

{

case 1: Cadastro(); Console.WriteLine(); break;

case 2: Consulta(); Console.WriteLine(); break;

case 3: for (int i = 0; i < Agenda.contador ; i++)

{

Agenda.contatos[i].Imprime();

} break;

case 4: Console.WriteLine("Saindo...."); Console.WriteLine("Pressione uma tecla!"); break;

default: Console.WriteLine("ERRO! Opção inválida"); break;

}

} while (op != 4);

Console.ReadKey();

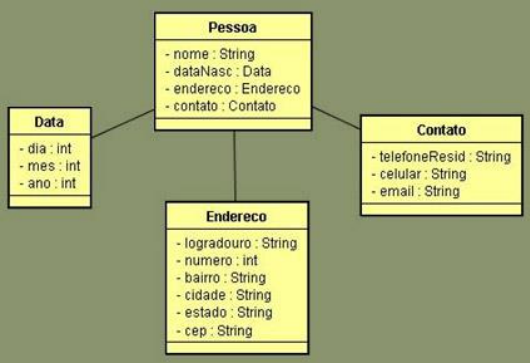
}

}

}

# EXERCÍCIO 5.5

Crie as classes apresentadas no diagrama abaixo e aplique a Composição para a classe Pessoa, que além de possuir um atributo Nome será composta pelas classes Data, Endereço e Contato para os atributosdataNasc, endereço e contato, respectivamente



//

// NOME DO PROGRAMA: Ex5\_5

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 29/10/2015

// DESCRIÇÃO: O programa implementa a class Pessoa que possui por composição

// enderenço, data e contato. No programa de teste, o usuário entra com seu dados

// que são impressos logo depois

using System;

namespace Ex5\_5

{

class Pessoa

{

Endereco E;

Data D;

Contato C;

string nome;

public string Nome{get{return nome;}}

public Endereco e{get{return E;}}

public Data d{get{return D;}}

public Contato c{get{return C;}}

public Pessoa(string nome)

{

E = new Endereco();

D = new Data();

C = new Contato();

Console.WriteLine("Digite seu logradouro: ");

E.Logradouro = Console.ReadLine();

Console.WriteLine("Digite o numero: ");

E.Numero = int.Parse(Console.ReadLine());

Console.WriteLine("Digite seu bairro: ");

E.Bairro = Console.ReadLine();

Console.WriteLine("Digite seu cidade: ");

E.Cidade = Console.ReadLine();

Console.WriteLine("Digite seu estado: ");

E.Estado = Console.ReadLine();

Console.WriteLine("Digite seu cep: ");

E.Cep = Console.ReadLine();

Console.WriteLine("Digite seu telefoneResidencial: ");

C.TelefoneResidencial = Console.ReadLine();

Console.WriteLine("Digite seu celular: ");

C.Celular = Console.ReadLine();

Console.WriteLine("Digite seu email: ");

C.Email = Console.ReadLine();

Console.WriteLine("Digite o ano: ");

D.Ano = int.Parse(Console.ReadLine());

Console.WriteLine("Digite o mes: ");

D.Mes = int.Parse(Console.ReadLine());

Console.WriteLine("Digite o dia: ");

D.Dia = int.Parse(Console.ReadLine());

this.nome = nome;

}

}

class Endereco

{

string logradouro;

int numero;

string bairro;

string cidade;

string estado;

string cep;

public string Logradouro{get{return logradouro;}set{logradouro = value;}}

public int Numero{get{return numero;}set{numero = value;}}

public string Bairro{get{return bairro;}set{bairro = value;}}

public string Cidade{get{return cidade;}set{cidade = value;}}

public string Estado{get{return estado;}set{estado = value;}}

public string Cep{get{return cep;}set{cep = value;}}

}

class Data

{

int dia;

int mes;

int ano;

public int Dia{get{return dia;}set{dia = value;}}

public int Mes{get{return mes;}set{mes = value;}}

public int Ano{get{return ano;}set{ano = value;}}

}

class Contato

{

string telefoneResidencial;

string celular;

string email;

public string TelefoneResidencial{get{return telefoneResidencial;}set{telefoneResidencial = value;}}

public string Celular{get{return celular;}set{celular = value;}}

public string Email{get{return email;}set{email = value;}}

}

class Program

{

public static void Main(string[] args)

{

Console.Write("\tProgramadores: ");

Console.WriteLine("\tAndrew de Oliveira Duchini (537983)");

Console.WriteLine("\tCristóvão Olegário de Castro (536584)");

Console.WriteLine("\tPedro Lucas Duarte Faria(536589)");

Console.WriteLine("\tSamara Luiza da Silva(536592)");

string nome;

Console.WriteLine("Digite seu nome: ");

nome = Console.ReadLine();

Pessoa P = new Pessoa(nome);

Console.WriteLine("Nome: "+ P.Nome);

Console.WriteLine();

Console.WriteLine("Celular: "+ P.c.Celular);

Console.WriteLine("Email: "+ P.c.Email);

Console.WriteLine("Telefone Residencial: "+ P.c.TelefoneResidencial);

Console.WriteLine();

Console.Write("{0}/", P.d.Dia);

Console.Write("{0}/", P.d.Mes);

Console.Write("{0}", P.d.Ano);

Console.WriteLine();

Console.WriteLine("CEP: "+ P.e.Cep);

Console.WriteLine("Logradouro: "+ P.e.Logradouro);

Console.WriteLine("Bairro: "+ P.e.Estado);

Console.WriteLine("Cidade: "+ P.e.Cidade);

Console.WriteLine("Bairro: "+ P.e.Bairro);

Console.WriteLine("Numero: "+ P.e.Numero);

Console.WriteLine();

Console.WriteLine();

Console.ReadKey();

}

}

}

# EXERCÍCIO 6.1

Usar técnicas de tratamento de exceções (C# e Java), interface gráfica de usuário (GUI) (C#) e eventos e delegados (C#) para aperfeiçoar os programas do exercício 3.3 dos Labs 1 e 4 (Formulários):

3.1 Converter C-F/F-C

//

// NOME DO PROGRAMA: Conversor

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 30/10/2015

// DESCRIÇÃO: Programa Conversor com tratamento de erros

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

txtCelsius.Enabled = false;

txtFahr.Enabled = false;

btnConverter.Enabled = false;

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

txtCelsius.Enabled = true;

txtFahr.Enabled = false;

btnConverter.Enabled = true;

txtFahr.Clear();

}

private void rbFahr\_CheckedChanged(object sender, EventArgs e)

{

txtFahr.Enabled = true;

txtCelsius.Enabled = false;

btnConverter.Enabled = true;

txtCelsius.Clear();

}

private void btnConverter\_Click(object sender, EventArgs e)

{

try

{

double temp;

if (rbCelsius.Checked == true)

{

if (txtCelsius.Text != "")

{

temp = double.Parse(txtCelsius.Text);

string msg = "Conversão de Celsius para Fahrenheit!\n" + temp + "°C = " + Math.Round(CelsiusToFahrenheit(temp), 2) + "°F";

MessageBox.Show(null, msg, "Resultado da Conversão", MessageBoxButtons.OK);

limpar();

}

else

{

MessageBox.Show("Favor preencher o campo!", "Aviso!");

}

}

else if (rbFahr.Checked == true)

{

if (txtFahr.Text != "")

{

temp = double.Parse(txtFahr.Text);

string msg = "Conversão de Fahrenheit para Celsius!\n" + temp + "°F = " + Math.Round(FahrenheitToCelcius(temp), 2) + "°C";

MessageBox.Show(null, msg, "Resultado da Conversão", MessageBoxButtons.OK);

limpar();

}

else

{

MessageBox.Show("Favor preencher o campo!", "Aviso!");

}

}

}

catch

{

MessageBox.Show("Valor inválido", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void limpar()

{

txtCelsius.Clear();

rbCelsius.Checked = false;

txtCelsius.Enabled = false;

txtFahr.Clear();

rbFahr.Checked = false;

txtFahr.Enabled = false;

btnConverter.Enabled = false;

}

private double CelsiusToFahrenheit(double temperatura)

{

double conversao;

conversao = (temperatura \* 1.8) + 32;

return conversao;

}

private double FahrenheitToCelcius(double temperatura)

{

double conversao;

conversao = (temperatura - 32) / 1.8;

return conversao;

}

}

3.2 Calculadora

public static string Acheoperacao(double a, double b, char op)

{

switch (op)

{

case '+': return (Convert.ToString(a + b));

case '-': return (Convert.ToString(a - b));

case '\*': return (Convert.ToString(a \* b));

case '/':

if (b==0)

{

return ("Erro!!");

}

else

{

return (Convert.ToString(a / b));

}

default: return (" ");

}

}

3.3 Conta

//

// NOME DO PROGRAMA: Conta\_39\_LAB1

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 30/10/2015

// DESCRIÇÃO: O programa conta com tratamento de erro

public partial class Sacar : Form

{

double a;

public Sacar()

{

InitializeComponent();

a = 0;

}

private void bt\_sacar\_Click(object sender, EventArgs e)

{

try

{

a = double.Parse(tBox\_valor.Text);

if(a<0)

{

MessageBox.Show("Não retiramos valores negativos", "AVISO!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

if (a <= Conta.Saldo)

{

Conta.Saldo = Conta.Saldo - a;

}

else

{

MessageBox.Show("Saldo insuficiente", "AVISO!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

catch

{

MessageBox.Show("Valor inválido", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

public partial class Conta : Form

{

public int agencia;

public int numero; // número da Conta Bancária

public string nome; // nome do titular da conta public

public static double Saldo;

public Conta()

{

InitializeComponent();

agencia = 1010;

numero = 015256;

nome = "Samara Luiza da Silva";

Saldo = 0;

lb\_nome.Text = nome;

lb\_conta.Text = ""+ numero;

lb\_agencia.Text = "" + agencia;

}

private void bt\_depositar\_Click(object sender, EventArgs e)

{

Depositar a = new Depositar();

a.Show();

}

private void bt\_sacar\_Click(object sender, EventArgs e)

{

Sacar b = new Sacar();

b.Show();

}

private void bt\_sair\_Click(object sender, EventArgs e)

{

this.Close();

}

private void bt\_imprimir\_Click(object sender, EventArgs e)

{

MessageBox.Show(("Saldo: " + Saldo), "Saldo", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

public partial class Depositar : Form

{

public Depositar()

{

InitializeComponent();

}

private void bt\_depositar\_Click(object sender, EventArgs e)

{

try

{

double a;

a = double.Parse(tBox\_valor.Text);

if (a < 0)

{

MessageBox.Show("Não depositamos valores negativos", "AVISO!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

Conta.Saldo = Conta.Saldo+double.Parse(tBox\_valor.Text);

}

this.Close();

}

catch

{

MessageBox.Show("Valor inválido", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

# EXERCÍCIO 6.2

Usar técnicas de tratamento de exceções, interface gráfica de usuário (GUI), eventos e delegados para aperfeiçoar os programas codificados em C# do exercício 7 do lab2 (Herança e arquivos):

//

// NOME DO PROGRAMA: Conta(6\_1)

//

// PROGRAMADORES:

// Pedro Lucas Duarte Faria(536589)

// Samara Luiza da Silva(536592)

// Andrew de Oliveira Duchini (537983)

// Cristóvão Olegário de Castro (536584)

// DATA: 08/11/2015

// DESCRIÇÃO: Utilizando os conceitos de formularios, tratamentos de erros e eventos foi implementada

// o programa Conta(6\_1) um aplicativo melhorado do exercício 7.2 do lab2

class Conta

{

protected DateTime abertura; // data de abertura da conta

protected String nome; // nome do titular da conta

protected int agencia; // número da agência da conta

protected double saldo; // saldo atual da conta

protected int numConta; // número da conta

protected int tipoConta; // indica o tipo da conta: corrente, poupança ou investimento

public static int cont = 0; //contador de contas existentes

public bool ativa = true;

public void depositar(double credito)

{

if ((ativa))

{

saldo += credito;

MessageBox.Show("Valor depositado", "Depósito", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Conta não existente", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void sacar(double valor)

{

if ((ativa))

{

saldo -= valor;

MessageBox.Show("Valor retirado!!", "Sacar", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Conta não existente", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void imprimeSaldo()

{

}

public string getNome()

{

return this.nome;

}

public int getAgencia()

{

return this.agencia;

}

public double Saldo

{

get { return this.saldo; }

set { this.saldo = value; }

}

public int getNum()

{

return this.numConta;

}

public int getTipo\_Conta()

{

return this.tipoConta;

}

public int mesesInvestidos(DateTime atual)

{

TimeSpan dif = atual.Subtract(abertura);

int meses = dif.Days / 30;

return meses;

}

public DateTime Abertura

{

set { abertura = value; }

get { return abertura; }

}

}

class Investimento : Conta

{

public static double rendimento = 0.65 / 100;

public static double imposto = 0.15;

public Investimento(String nome, int ag, double saldoInicial, DateTime r)// Contrutor

{

base.nome = nome;

agencia = ag;

saldo = saldoInicial;

numConta = cont + 1;

tipoConta = 2;

cont++;

abertura = r.Date;

}

public Investimento(String nome, int ag, double saldoInicial, int num, DateTime a)// Contrutor para os dados que vem do aquivo

{

base.nome = nome;

base.agencia = ag;

base.saldo = saldoInicial;

base.numConta = num;

base.tipoConta = 2;

cont++;

abertura = a;

}

public double obterSaldo()

{

return base.saldo;

}

}

class Poupanca : Conta

{

public static double taxa\_juros = 0.5/100;

public Poupanca()

{ }

public Poupanca(String nome, int ag, double saldoInicial, DateTime r)// Contrutor

{

base.nome = nome;

base.agencia = ag;

base.saldo = saldoInicial;

base.numConta = cont + 1;

base.tipoConta = 1;

cont++;

abertura = r.Date;

}

public Poupanca(String nome, int ag, double saldoInicial, int num, DateTime a)// Contrutor para os dados que vem do aquivo

{

base.nome = nome;

base.agencia = ag;

base.saldo = saldoInicial;

base.numConta = num;

base.tipoConta = 1;

cont++;

abertura = a;

}

public double obterSaldo()

{

return base.saldo;

}

}

public partial class Menu : Form

{

public Menu()

{

InitializeComponent();

Contole.Ler();

}

private void btn\_novaconta\_Click(object sender, EventArgs e)

{

NovaConta a = new NovaConta();

a.Show();

}

private void btn\_excluir\_Click(object sender, EventArgs e)

{

ExcluirConta a = new ExcluirConta();

a.Show();

}

private void btn\_depositar\_Click(object sender, EventArgs e)

{

Depositar a = new Depositar();

a.Show();

}

private void btn\_sacar\_Click(object sender, EventArgs e)

{

Sacar a = new Sacar();

a.Show();

}

private void btn\_saldo\_Click(object sender, EventArgs e)

{

Saldo a = new Saldo();

a.Show();

}

private void btn\_contas\_Click(object sender, EventArgs e)

{

Relacao a = new Relacao();

a.Show();

}

private void btn\_sair\_Click(object sender, EventArgs e)

{

Contole.Gravar();

this.Close();

}

private void pedroLucasDuarteFaria536589ToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

}

public partial class NovaConta : Form

{

DateTime r = new DateTime();

public NovaConta()

{

InitializeComponent();

lb\_data.Text = (r.Day + "/" + r.Month + "/" + r.Year);

lb\_numConta.Text = Convert.ToString(Conta.cont+1);

}

private void btn\_criar\_Click(object sender, EventArgs e)

{

string nome = "";

int ag = 0;

double saldoInicial = 0;

try

{

nome = tb\_nome.Text;

ag = int.Parse(tb\_agencia.Text);

saldoInicial = double.Parse(tb\_saldoini.Text);

if (radioBtn\_poupanca.Checked)

{

Poupanca a = new Poupanca(nome, ag, saldoInicial, r);

Contole.lista.Add(a);

MessageBox.Show("Conta criada com sucesso!!", "Conta", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

else if (radioBtn\_poupanca.Checked)

{

Investimento a = new Investimento(nome, ag, saldoInicial, r);

Contole.lista.Add(a);

MessageBox.Show("Conta criada com sucesso!!", "Conta", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

else

{

MessageBox.Show("Marque uma opção", "COMPLETE", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch

{

MessageBox.Show("Valores inválidos", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

public partial class Saldo : Form

{

public Saldo()

{

InitializeComponent();

}

private void btn\_consulta\_Click(object sender, EventArgs e)

{

int a;

try

{

a = int.Parse(tb\_numconta.Text);

if (Contole.lista[a - 1].ativa)

{

lb\_display\_saldo.Text =Convert.ToString(Contole.lista[a - 1].Saldo);

}

else

{

MessageBox.Show("Conta não existente", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch

{

MessageBox.Show("Conta não existente", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

public partial class Sacar : Form

{

public Sacar()

{

InitializeComponent();

}

private void btn\_saque\_Click(object sender, EventArgs e)

{

int a;

double b;

try

{

a = int.Parse(tb\_numconta.Text);

b = double.Parse(tb\_saque.Text);

if (b < 0)

{

MessageBox.Show("Não sacamos valores negativos", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

Contole.lista[a - 1].sacar(b);

this.Close();

}

}

catch

{

MessageBox.Show("Valores inválidos", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

public partial class Relacao : Form

{

public Relacao()

{

InitializeComponent();

listBox1.Text = "";

for(int i = 0; i<Conta.cont; i++)

{

if (Contole.lista[i].ativa)

{

listBox1.Items.Add("NOME: " + Contole.lista[i].getNome() + " AGÊNCIA: " + Contole.lista[i].getAgencia());

}

}

}

}

public partial class ExcluirConta : Form

{

public ExcluirConta()

{

InitializeComponent();

}

private void btn\_excluir\_Click(object sender, EventArgs e)

{

int a;

try

{

a = int.Parse(tb\_numconta.Text);

Contole.lista[a - 1].ativa = false;

MessageBox.Show("Conta excluida!", "Excluir", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

catch

{

MessageBox.Show("Valores inválidos", "ERRO!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

class Contole

{

public static List<Conta> lista = new List<Conta>();

public static void Ler()

{

string[] dados\_arq;

StreamReader leitura = new StreamReader("Banco.txt");

while (!(leitura.EndOfStream))

{

string linha = leitura.ReadLine();

if( linha != null)

{

dados\_arq = linha.Split(';');

string nome\_titular = dados\_arq[0];

int age = int.Parse(dados\_arq[1]);

double saldo = double.Parse(dados\_arq[2]);

int num = int.Parse(dados\_arq[3]);

int tipo = int.Parse(dados\_arq[4]);

DateTime abertura = DateTime.Parse(dados\_arq[5]);

if (tipo == 1)

{

Poupanca a = new Poupanca(nome\_titular, age, saldo, num, abertura);

lista.Add(a);

}

else

{

Investimento a = new Investimento(nome\_titular, age, saldo, num, abertura);

lista.Add(a);

}

}

}

leitura.Close();

}

public static void Gravar()

{

StreamWriter arq = new StreamWriter("Banco.txt");

for (int i = 0; i < Conta.cont; i++)

{

if (lista[i].ativa)

{

arq.WriteLine(lista[i].getNome() + ";" + lista[i].getAgencia() + ";" + lista[i].Saldo + ";" + lista[i].getNum() + ";" + lista[i].getTipo\_Conta() + ";" + lista[i].Abertura.ToShortDateString());

}

}

arq.Close();

}

}

# EXERCÍCIO 6.3

Compile os programas da pasta Eventos\_delegados, disponível no SGA, traduza os comentários, explique o que os programas fazem, identifique em que partes dos códigos os conceitos de tratamento de exceções, eventos, delegados e interfaces gráficas foram utilizados.

// Fig 11.1: DivideByZeroTest.cs

// Basics of C# exception handling.

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

namespace divideZero1

{

///<Sumario>

///Essa classe demonstra como lidar com excecoes de

///divisao por zero em integrações aritmeticas e de

///formatos de expressoes numericas

///</Sumario>

public class DivideByZeroTest : System.Windows.Forms.Form

{

//Criação dos botões e labels do programa

private System.Windows.Forms.Label numeratorLabel;

private System.Windows.Forms.TextBox numeratorTextBox;

private System.Windows.Forms.Label denominatorLabel;

private System.Windows.Forms.TextBox denominatorTextBox;

private System.Windows.Forms.Button divideButton;

private System.Windows.Forms.Label outputLabel;

/// <summary>

/// Requer variavel designada

/// </summary>

private System.ComponentModel.Container components = null;

// Construtor Padrao

public DivideByZeroTest()

{

// Requerido pelo designer do windons form

InitializeComponent();

}

/// <summary>

/// Limpa qualquer recurso que estava sendo usado

/// </summary>

protected override void Dispose( bool disposing )

{

if ( disposing )

{

if ( components != null )

{

components.Dispose();

}

}

base.Dispose( disposing );

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// Metodo requerido para suporte do designer -não modifique

/// o conteudo desse metodo com o editor do codigo

/// </summary>

private void InitializeComponent()

{

//Inicializa as labels e os botoes que serao usados no programa

this.divideButton = new System.Windows.Forms.Button();

this.denominatorTextBox = new System.Windows.Forms.TextBox();

this.numeratorTextBox = new System.Windows.Forms.TextBox();

this.denominatorLabel = new System.Windows.Forms.Label();

this.outputLabel = new System.Windows.Forms.Label();

this.numeratorLabel = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// Botao divisor

//

this.divideButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 11F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));

this.divideButton.Location = new System.Drawing.Point(16, 88);

this.divideButton.Name = "divideButton";

this.divideButton.Size = new System.Drawing.Size(120, 32);

this.divideButton.TabIndex = 3;

this.divideButton.Text = "Click to Divide";

this.divideButton.Click += new System.EventHandler(this.divideButton\_Click);

//

// Dominador da caixa de texto

//

this.denominatorTextBox.Location = new System.Drawing.Point(136, 48);

this.denominatorTextBox.Name = "denominatorTextBox";

this.denominatorTextBox.Size = new System.Drawing.Size(120, 20);

this.denominatorTextBox.TabIndex = 5;

this.denominatorTextBox.Text = "";

//

// Numerador caixa de texto

//

this.numeratorTextBox.Location = new System.Drawing.Point(136, 8);

this.numeratorTextBox.Name = "numeratorTextBox";

this.numeratorTextBox.Size = new System.Drawing.Size(120, 20);

this.numeratorTextBox.TabIndex = 4;

this.numeratorTextBox.Text = "";

//

// Dominador da label

//

this.denominatorLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));

this.denominatorLabel.Location = new System.Drawing.Point(8, 50);

this.denominatorLabel.Name = "denominatorLabel";

this.denominatorLabel.Size = new System.Drawing.Size(112, 16);

this.denominatorLabel.TabIndex = 1;

this.denominatorLabel.Text = "Enter denominator";

//

// Label de saida

//

this.outputLabel.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;

this.outputLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));

this.outputLabel.Location = new System.Drawing.Point(168, 96);

this.outputLabel.Name = "outputLabel";

this.outputLabel.Size = new System.Drawing.Size(72, 23);

this.outputLabel.TabIndex = 2;

//

// Numerador de Label

//

this.numeratorLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));

this.numeratorLabel.Location = new System.Drawing.Point(8, 10);

this.numeratorLabel.Name = "numeratorLabel";

this.numeratorLabel.Size = new System.Drawing.Size(112, 16);

this.numeratorLabel.TabIndex = 0;

this.numeratorLabel.Text = "Enter numerator";

//

// Teste divisao por zero

//

//Desenha na tela os botoes

this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);

this.ClientSize = new System.Drawing.Size(264, 133);

this.Controls.AddRange(new System.Windows.Forms.Control[] {

this.divideButton,

this.denominatorTextBox,

this.numeratorTextBox,

this.numeratorLabel,

this.outputLabel,

this.denominatorLabel});

this.Name = "DivideByZeroTest";

this.Text = "DivideByZero";

this.ResumeLayout(false);

}

#endregion

/// <summary>

/// The main entry point for the application.

/// O ponto de entrada principal da aplicação

/// </summary>

[STAThread]

static void Main()

{

Application.Run( new DivideByZeroTest() );

}

// obtain integers input by user and divide numerator

// Obtem os inteiros de entrada do usuario e o divide numerador por denominador

private void divideButton\_Click(

object sender, System.EventArgs e )

{

outputLabel.Text = "";

// recupera inteiros do usuario e chama o quociente

try

{

// Convert.TpInt32 gera FormatException se argumentos não forem inteiros

int numerator = Convert.ToInt32( numeratorTextBox.Text );

int denominator =

Convert.ToInt32( denominatorTextBox.Text );

// divisao gera DivideByZeroException se denominador for 0

int result = numerator / denominator;

outputLabel.Text = result.ToString();

} // Finaliza o try

// process invalid number format

catch ( FormatException )

{

MessageBox.Show( "You must enter two integers",

"Invalid Number Format",

MessageBoxButtons.OK, MessageBoxIcon.Error );

}

// usuario tenta fazer a divisao por 0

catch ( DivideByZeroException divideByZeroException )

{

MessageBox.Show( divideByZeroException.Message,

"Attempted to Divide by Zero",

MessageBoxButtons.OK, MessageBoxIcon.Error );

}

} // end method divideButton\_Click

} // end class DivideByZeroTest

}

/\*

Esse programa tem como objetivo criar uma interface grafica para gerar botões para se fazer um calculo de divisao

por 2 numeros. Esse programa possui o tratemento de excessao caso os numero de entrada entrem de maneira indevida,

exemplo, o usuario tenta entra com uma letra no lugar do numero ou se ele tenta entra com o digito 0 no lugar do numerador.

Com isso ele possui um metodo para tratar execessões de divideByZeroException e formatException caso os valores sejam

entrados de forma errada. Esse programa ao final nos mostra o resultado da divisao dos dois numeros e em seguida ele

reescreve o metodo Dispose para poder fazer liberação da memoria utilizada.

\*/

// Fig 11.5: SquareRootTest.cs

// Demonstrating a user-defined exception class.

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

namespace NegativeNumberSquareRoot

{

/// <summary>

/// Aceita a entrada e computa a raiz quadrada da entrada

/// </summary>

public class SquareRootTest : System.Windows.Forms.Form

{

private System.Windows.Forms.Label inputLabel;

private System.Windows.Forms.Label outputLabel;

private System.Windows.Forms.TextBox inputTextBox;

private System.Windows.Forms.Button squareRootButton;

/// <summary>

/// Requer variavel designada

/// </summary>

private System.ComponentModel.Container components = null;

// Construtor padrao

public SquareRootTest()

{

//

// requerido pelo suporte de designer windons form

//

InitializeComponent();

}

/// <summary>

/// limpa qualquer recurso alocado

/// </summary>

protected override void Dispose( bool disposing )

{

if ( disposing )

{

if ( components != null )

{

components.Dispose();

}

}

base.Dispose( disposing );

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// Metodo requerido para suporte do designer -não modifique

/// o conteudo desse metodo com o editor do codigo

/// </summary>

private void InitializeComponent()

{

//Metodo responsavel por criar os botoes que serao vistos na tela.

this.outputLabel = new System.Windows.Forms.Label();

this.squareRootButton = new System.Windows.Forms.Button();

this.inputTextBox = new System.Windows.Forms.TextBox();

this.inputLabel = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// Label de saida

//

this.outputLabel.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;

this.outputLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));

this.outputLabel.Location = new System.Drawing.Point(14, 71);

this.outputLabel.Name = "outputLabel";

this.outputLabel.Size = new System.Drawing.Size(264, 23);

this.outputLabel.TabIndex = 2;

//

// Bota de raiz quadrada

//

this.squareRootButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));

this.squareRootButton.Location = new System.Drawing.Point(90, 37);

this.squareRootButton.Name = "squareRootButton";

this.squareRootButton.Size = new System.Drawing.Size(112, 27);

this.squareRootButton.TabIndex = 1;

this.squareRootButton.Text = "Square Root";

this.squareRootButton.Click += new System.EventHandler(this.squareRootButton\_Click);

//

// Caixa de entrada de texto

//

this.inputTextBox.Location = new System.Drawing.Point(164, 8);

this.inputTextBox.Name = "inputTextBox";

this.inputTextBox.TabIndex = 3;

this.inputTextBox.Text = "";

//

// Label de entrada

//

this.inputLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));

this.inputLabel.Location = new System.Drawing.Point(28, 10);

this.inputLabel.Name = "inputLabel";

this.inputLabel.Size = new System.Drawing.Size(124, 16);

this.inputLabel.TabIndex = 0;

this.inputLabel.Text = "Please enter a number:";

//

// Teste de raiz quadrada

//

this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);

this.ClientSize = new System.Drawing.Size(292, 100);

this.Controls.AddRange(new System.Windows.Forms.Control[] {

this.inputTextBox,

this.outputLabel,

this.squareRootButton,

this.inputLabel});

this.Name = "SquareRootTest";

this.Text = "Computing the Square Root";

this.ResumeLayout(false);

}

#endregion

/// <summary>

/// Ponte de entrada principal do programa

/// </summary>

[STAThread]

static void Main()

{

Application.Run( new SquareRootTest() );

}

//computa a raiz quadrada de seus parametros; arremessa

//NegativeNumberException se o parametro é negativo

public double SquareRoot( double operand )

{

// Se operando negativo, arremessa NegativeNumberException

if ( operand < 0 )

throw new NegativeNumberException(

"Square root of negative number not permitted" );

// compute the square root

return Math.Sqrt( operand );

} // finaliza classe raiz quadrada

// obtain user input, convert to double and calculate

// Obtem a entrada do usuario, converte para double e calcula

// razi quadrada

private void squareRootButton\_Click(

object sender, System.EventArgs e )

{

outputLabel.Text = "";

// Pega qualquer NegativeNumberException

try

{

double result =

SquareRoot( Double.Parse( inputTextBox.Text ) );

outputLabel.Text = result.ToString();

}

// processa numero no formato errado

catch ( FormatException notInteger )

{

MessageBox.Show( notInteger.Message,

"Invalid Number Format", MessageBoxButtons.OK,

MessageBoxIcon.Error );

}

// Exibe a caixa de mensagem se numero de entrada for negativo

catch ( NegativeNumberException error )

{

MessageBox.Show( error.Message, "Invalid Operation",

MessageBoxButtons.OK, MessageBoxIcon.Error );

}

} // Finaliza metodo squareRootButton\_Click

} // finaliza classe SquareRootTest

}

/\*

Esse programa tem como finalidade desenhar na tela botões para receber um numero

do usuario e com isso calcular sua raiz quadrada. Com a entrada dos valores, sera

feita uma analise para verificar se os valores entrados são corretos. Vemos se o numero

entrado é negativo e se o usuario entrou com um caracter ao invez de um numero. Ao final

do programa vemos o resultado da raiz quadrada do numero.

\*/

using System;

namespace NegativeNumberSquareRoot

{

/// <summary>

/// NegativeNumberException representam excessoes causadas por

/// operacoes ilegais realizadas com numeros negativos

/// </summary>

class NegativeNumberException : ApplicationException

{

// Construtor padrao

public NegativeNumberException()

: base( "Illegal operation for a negative number" )

{

}

// Construtor para mensagens de erro customizados

public NegativeNumberException( string message )

: base( message )

{

}

// constructor for customizing error message and

// Construtor para mensagens de erro customizados e especificadores de excessoes internas de objetos

public NegativeNumberException(

string message, Exception inner )

: base( message, inner )

{

}

}

//Finaliza a classe NegativeNumberException

}

/\*

Este programa tem como finalidade informar ao usuario que está

sendo realizado uma operação ilegal com numeros negativos.

Por exemplo caso o usuario esteja fazendo uma raiz quadrada

de um numero, esse programa ira rodar, caso ocorra de a parcela

ser um numero negativo, o programa irá informar que a operacao de

raiz quadrada não pode ser efetuada em numeros negativos.

\*/

using System;

// Demonstra o uso de operadores checados e não checados

class Overflow

{

static void Main( string[] args )

{

int number1 = Int32.MaxValue; // 2,147,483,647

int number2 = Int32.MaxValue; // 2,147,483,647

int sum = 0;

Console.WriteLine(

"number1: {0}\nnumber2: {1}", number1, number2 );

// Calcula o somatorio de number1 e number2

try

{

Console.WriteLine(

"\nSum integers in checked context:" );

sum = checked( number1 + number2 );

}

// Pega excessao de overflow

catch ( OverflowException overflowException )

{

Console.WriteLine( overflowException.ToString() );

}

Console.WriteLine(

"\nsum after checked operation: {0}", sum );

Console.WriteLine(

"\nSum integers in unchecked context:" );

sum = unchecked( number1 + number2 );

Console.WriteLine(

"sum after unchecked operation: {0}", sum );

} // Finaliza metodo Main

} // Finaliza classe Overflow

/\*

Esse programa tem como objetivo detectar quando ocorre uma excessao

do tipo Overflow. Essa excessao ocorre quando tentamos armazenar em

uma variavel um valor superior ao valor suportado pelo tipo da variavel.

Caso o usuario faça uma operacao, esse programa ira rodar, testando

para ver se ocorrera uma soma que o resultado seja maior do que o

suportado pelo tipo da variavel.

\*/

using System;

// Demosntra o uso da mensagem, propriedades StackTrace e InnerException

class Properties

{

static void Main( string[] args )

{

// Chama o metodo1, qualquer excessao gerada irá ser

// pega pelo bloco catch que segue

try

{

Method1();

}

// Representação da string de saida da excessao,

// entao valores de saida da InnerException, mensagem

// propriedades da StackTrace

catch ( Exception exception )

{

Console.WriteLine(

"exception.ToString(): \n{0}\n",

exception.ToString() );

Console.WriteLine( "exception.Message: \n{0}\n",

exception.Message );

Console.WriteLine( "exception.StackTrace: \n{0}\n",

exception.StackTrace );

Console.WriteLine(

"exception.InnerException: \n{0}",

exception.InnerException );

} // Finaliza catch

} // Finaliza Main

// Chama o metodo2

public static void Method1()

{

Method2();

}

// Chama o metodo3

public static void Method2()

{

Method3();

}

// Arremeca uma excessao contendo uma InnerException

public static void Method3()

{

// Tentativa de converter uma string nao inteira para int

try

{

Convert.ToInt32( "Not an integer" );

}

// Pega a FormatException e converte ela em uma nova excessao

catch ( FormatException error )

{

throw new Exception(

"Exception occurred in Method3", error );

}

} // Finaliza metodo3

} // Finaliza a classe usingException

/\*

//Este programa tem como objetivo captar qualquer tipo excessao causada pelo programa

//e mostrar que e possivel criar um metodo que pegue qualquer tipo de excessao e a converta

//em um tipo de excessao generico. Nesse caso ele tenta converter uma frase:"Not an integer"

//em um numero inteiro. Nesse caso ocorrerá uma FormatException, porem ela foi convertida em

//uma exception do tipo format. Com isso o usuario verá as informacoes dessa excessao.

\*/

# EXERCÍCIO 6.4

Compile os programas da pasta Excecoes, disponível no SGA, traduza os comentários, explique o que os programas fazem, identifique em que partes dos códigos os conceitos de tratamento de exceções, eventos, delegados e interfaces gráficas foram utilizados.

//Usando eventos para simularo um departamento de incendio

using System;

//Classe FireAlarmEventArgs deriva de EventArgs e contem informacoes de eventos

class FireAlarmEventArgs : EventArgs

{

private string location;

public FireAlarmEventArgs( string location )

{

this.location = location;

}

public string Location{get{return location;}}

} // finaliza classe FireAlarmEventArgs

//Classe SmokeDetector define o evento delegado, objeto e chama o delegate quando o evento e levantado

class SmokeDetector

{

string location;

//Define um evento delegado

public delegate void FireAlarmEventHandler( object sender, FireAlarmEventArgs e );

//Cria uma referencia de evento delegado

public event FireAlarmEventHandler FireAlarm;

public SmokeDetector( string location )

{

this.location = location;

Console.WriteLine( "Smoke detector installed at " + location );

}

//Chama o cuidadores de eventos se o delegate nao vazio,

//argumento de evento de objeto passado por parametro

protected void OnFireAlarm( FireAlarmEventArgs e )

{

//FireAlarm e a referencia de delegate - tenha certeza que nao esteja vazio

if ( FireAlarm != null )

{

FireAlarm( this, e ); //passa o lancadir e os argumentos do evento

}

}

//Cria objeto de evento, passa para OnFireAlarm

public void Alarm()

{

Console.WriteLine( "Smoke alarm at " + location );

FireAlarmEventArgs e = new FireAlarmEventArgs( location );

OnFireAlarm( e );

}

} // finaliza classe SmokeDetector

//Classe Firedepartment rigistra um cuidador de evento e responde quando o evento ocorre

class FireDepartment

{

SmokeDetector smokeDetector;

public FireDepartment( SmokeDetector smokeDetector )

{

this.smokeDetector = smokeDetector;

//Rigistra metodo de Respond como um cuidador de evento

smokeDetector.FireAlarm += new SmokeDetector.FireAlarmEventHandler( Respond );

Console.WriteLine( "Smoke detector registered " + "at fire department" );

}

//Metodo assinatura confirma delegate

private void Respond( object sender, FireAlarmEventArgs e )

{

Console.WriteLine( "Fire department responding to alarm at " + e.Location + ", we are on our way!" );

}

} //Finaliza classe FireDepartment

//Motorista

class EventExample

{

static void Main( string[] args )

{

//Cria um novo detector de fumaca

SmokeDetector smokeDetector = new SmokeDetector( "102 East Main St." );

//Adiciona ao departamento de incendio

FireDepartment fireDepartment = new FireDepartment( smokeDetector );

smokeDetector.Alarm(); // chama evento Alarm

Console.ReadLine(); // Atrasa saida do programa

}

} // finaliza classe EventExample

/\*

Esse programa tem como finalidade mostrar como funciona os eventos em C#, para isso foi criado uma classe

Departamento de incendio. Com isso e criado um evento que sera chamado quando ocorrer um incendio. Com isso

caso ocorra o invencio, o delegate ira passar a funcao responsavel para tratar quando o evento ocorrer. Com isso

caso o evento ocorra o delegate passa a funcao que ira realizar as medidas corretas para tratar esse evento.

\*/

# EXERCÍCO 6.5

Compile os programas da pasta GUI, disponível no SGA, traduza os comentários, explique o que os programas fazem, identifique em que partes dos códigos os conceitos de tratamento de exceções, eventos, delegados e interfaces gráficas foram utilizados.

CheckBoxTest – O programa modifica o estilo do texto de acordo com a seleção do usuário, Por de eventos gerados pelas checkBox o programa alternar entre itálico e negrito, o texto que está em uma Label. O programa não apresenta tratamento de erros.

// Fig. 12.24: CheckBoxTest.cs

// Usando CheckBoxes parar alternar entre o italico e o negrito.

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

namespace CheckBoxTest

{

/// <summary>

/// o formulário contém checkboxes para permitir

/// o usuário modificar um texto exemplo

/// </summary>

public class CheckBoxTest : System.Windows.Forms.Form

{

private System.Windows.Forms.CheckBox boldCheckBox;

private System.Windows.Forms.CheckBox italicCheckBox;

private System.Windows.Forms.Label outputLabel;

/// <summary>

/// Variável de designer necessária.

/// </summary>

private System.ComponentModel.Container components = null;

// construtor padrão

public CheckBoxTest()

{

InitializeComponent();

}

/// <summary>

/// Limpar todos os recursos a ser utilizado.

/// </summary>

protected override void Dispose(bool disposing)

{

if (disposing)

{

if (components != null)

{

components.Dispose();

}

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Metodo requerido para Designer support – não modificar

/// os componentes deste metodo com o editor de código.

/// </summary>

private void InitializeComponent()

{

this.boldCheckBox =

new System.Windows.Forms.CheckBox();

this.outputLabel =

new System.Windows.Forms.Label();

this.italicCheckBox =

new System.Windows.Forms.CheckBox();

this.SuspendLayout();

//

// boldCheckBox

//

this.boldCheckBox.Location =

new System.Drawing.Point(72, 56);

this.boldCheckBox.Name = "boldCheckBox";

this.boldCheckBox.Size =

new System.Drawing.Size(48, 24);

this.boldCheckBox.TabIndex = 1;

this.boldCheckBox.Text = "Bold";

// Registrar manipuladores de eventos

this.boldCheckBox.CheckedChanged +=

new System.EventHandler(

this.boldCheckBox\_CheckedChanged);

//

// outputLabel

//

this.outputLabel.Location =

new System.Drawing.Point(64, 24);

this.outputLabel.Name = "outputLabel";

this.outputLabel.Size =

new System.Drawing.Size(152, 16);

this.outputLabel.TabIndex = 0;

this.outputLabel.Text =

"Watch the font style change";

//

// italicCheckBox

//

this.italicCheckBox.Location =

new System.Drawing.Point(160, 56);

this.italicCheckBox.Name = "italicCheckBox";

this.italicCheckBox.Size =

new System.Drawing.Size(48, 24);

this.italicCheckBox.TabIndex = 1;

this.italicCheckBox.Text = "Italic";

this.italicCheckBox.CheckedChanged +=

new System.EventHandler(

this.italicCheckBox\_CheckedChanged);

//

// CheckBoxTest

//

this.AutoScaleBaseSize =

new System.Drawing.Size(5, 13);

this.ClientSize =

new System.Drawing.Size(264, 109);

this.Controls.AddRange(

new System.Windows.Forms.Control[] {

this.italicCheckBox,

this.boldCheckBox,

this.outputLabel });

this.Name = "CheckBoxTest";

this.Text = "CheckBoxTest";

this.ResumeLayout(false);

} // fim do metodo InitializeComponent

// fim da região beginning on line 50

#endregion

/// <summary>

/// O principal ponto de partida para o aplicativo.

/// </summary>

[STAThread]

static void Main()

{

Application.Run(new CheckBoxTest());

}

// Deixar o texto em negrito se não estiver em negrito,

// se já estiver em negrito deixa-lo normal

private void boldCheckBox\_CheckedChanged(

object sender, System.EventArgs e)

{

outputLabel.Font =

new Font(outputLabel.Font.Name,

outputLabel.Font.Size,

outputLabel.Font.Style ^ FontStyle.Bold);

}

// Deixar o texto em italico se não estiver em italico,

// se já estiver em italico deixa-lo normal

private void italicCheckBox\_CheckedChanged(

object sender, System.EventArgs e)

{

outputLabel.Font =

new Font(outputLabel.Font.Name,

outputLabel.Font.Size,

outputLabel.Font.Style ^ FontStyle.Italic);

}

} // fim da class CheckBoxTest

} // fim do namespace CheckBoxTest

GroupBoxPanelExample – O aplicativo mostra um GroupBox com os botões Hi e Bye, e um painel com os botões Far Left e Far Right. Caso um dos botões for pressionado uma label mostrará qual foi. Cada vez que um botão é pressionado, é gerado um evento que muda o texto da label. O programa não tem tratamento de erro.

// Fig. 12.22: GroupBoxPanelExample.cs

// Usando GroupBoxes e Panels para mostrar botões.

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

// namespace que contém a class

// para nossa demostração de groupbox e panel

namespace GroupBoxPanelExample

{

/// <summary>

/// Formulário para mostrar um groupbox versus um panel

/// </summary>

public class GroupBoxPanelExample : System.Windows.Forms.Form

{

private System.Windows.Forms.Button hiButton;

private System.Windows.Forms.Button byeButton;

private System.Windows.Forms.Button leftButton;

private System.Windows.Forms.Button rightButton;

private System.Windows.Forms.GroupBox mainGroupBox;

private System.Windows.Forms.Label messageLabel;

private System.Windows.Forms.Panel mainPanel;

private System.ComponentModel.Container components = null;

// construtor padrão

public GroupBoxPanelExample()

{

InitializeComponent();

}

/// <summary>

/// Limpar todos os recursos a ser utilizado.

/// </summary>

protected override void Dispose( bool disposing )

{

if ( disposing )

{

if ( components != null )

{

components.Dispose();

}

}

base.Dispose( disposing );

}

#region Windows Form Designer generated code

/// <summary>

/// Metodo requerido para Designer support – não modificar

///os componentes deste metodo com o editor de código

/// </summary>

private void InitializeComponent()

{

this.messageLabel = new System.Windows.Forms.Label();

this.byeButton = new System.Windows.Forms.Button();

this.mainGroupBox = new System.Windows.Forms.GroupBox();

this.hiButton = new System.Windows.Forms.Button();

this.mainPanel = new System.Windows.Forms.Panel();

this.rightButton = new System.Windows.Forms.Button();

this.leftButton = new System.Windows.Forms.Button();

this.mainGroupBox.SuspendLayout();

this.mainPanel.SuspendLayout();

this.SuspendLayout();

//

// messageLabel

//

this.messageLabel.Location =

new System.Drawing.Point( 24, 144 );

this.messageLabel.Name = "messageLabel";

this.messageLabel.Size =

new System.Drawing.Size( 176, 32 );

this.messageLabel.TabIndex = 2;

//

// byeButton

//

this.byeButton.Location =

new System.Drawing.Point( 96, 48 );

this.byeButton.Name = "byeButton";

this.byeButton.Size =

new System.Drawing.Size( 72, 32 );

this.byeButton.TabIndex = 0;

this.byeButton.Text = "Bye";

// register event handler

this.byeButton.Click +=

new System.EventHandler( this.byeButton\_Click );

//

// mainGroupBox

//

// add controls to mainGroupBox

this.mainGroupBox.Controls.AddRange(

new System.Windows.Forms.Control[] {

this.byeButton,

this.hiButton } );

this.mainGroupBox.Location =

new System.Drawing.Point( 24, 24 );

this.mainGroupBox.Name = "mainGroupBox";

this.mainGroupBox.Size =

new System.Drawing.Size( 176, 104 );

this.mainGroupBox.TabIndex = 0;

this.mainGroupBox.TabStop = false;

this.mainGroupBox.Text = "Main GroupBox";

//

// hiButton

//

this.hiButton.Location =

new System.Drawing.Point( 8, 48 );

this.hiButton.Name = "hiButton";

this.hiButton.Size =

new System.Drawing.Size( 72, 32 );

this.hiButton.TabIndex = 0;

this.hiButton.Text = "Hi";

this.hiButton.Click += new System.EventHandler(

this.hiButton\_Click );

//

// mainPanel

//

this.mainPanel.AutoScroll = true;

this.mainPanel.BorderStyle =

System.Windows.Forms.BorderStyle.FixedSingle;

// add controls to mainPanel

this.mainPanel.Controls.AddRange(

new System.Windows.Forms.Control[] {

this.rightButton,

this.leftButton } );

this.mainPanel.Location =

new System.Drawing.Point( 24, 192 );

this.mainPanel.Name = "mainPanel";

this.mainPanel.Size =

new System.Drawing.Size( 176, 112 );

this.mainPanel.TabIndex = 1;

//

// rightButton

//

this.rightButton.Location =

new System.Drawing.Point( 264, 40 );

this.rightButton.Name = "rightButton";

this.rightButton.Size =

new System.Drawing.Size( 80, 40 );

this.rightButton.TabIndex = 0;

this.rightButton.Text = "Far Right";

this.rightButton.Click += new System.EventHandler(

this.rightButton\_Click );

//

// leftButton

//

this.leftButton.Location =

new System.Drawing.Point( 8, 40 );

this.leftButton.Name = "leftButton";

this.leftButton.Size =

new System.Drawing.Size( 80, 40 );

this.leftButton.TabIndex = 0;

this.leftButton.Text = "Far Left";

this.leftButton.Click +=

new System.EventHandler( this.leftButton\_Click );

//

// GroupBoxPanelExample

//

this.AutoScaleBaseSize =

new System.Drawing.Size( 5, 13 );

this.ClientSize =

new System.Drawing.Size( 224, 317 );

this.Controls.AddRange(

new System.Windows.Forms.Control[] {

this.messageLabel,

this.mainPanel,

this.mainGroupBox } );

this.Name = "GroupBoxPanelExample";

this.Text = "GroupBoxPanelExample";

this.mainGroupBox.ResumeLayout( false );

this.mainPanel.ResumeLayout( false );

this.ResumeLayout( false );

} // end method InitializeComponent

// end region beginning on line 53

#endregion

/// <summary>

/// O principal ponto de partida para o aplicativo.

/// </summary>

[STAThread]

static void Main()

{

Application.Run( new GroupBoxPanelExample() );

}

// Manipuladores de eventos para mudar messageLabel

// Manipulador de evento para hi button

private void hiButton\_Click(

object sender, System.EventArgs e )

{

messageLabel.Text= "Hi pressed";

}

// Manipulador de evento para bye button

private void byeButton\_Click(

object sender, System.EventArgs e )

{

messageLabel.Text = "Bye pressed";

}

// Manipulador de evento para far left button

private void leftButton\_Click(

object sender, System.EventArgs e )

{

messageLabel.Text = "Far left pressed";

}

// Manipulador de evento para far right button

private void rightButton\_Click(

object sender, System.EventArgs e )

{

messageLabel.Text = "Far right pressed";

}

} // fim da classe GroupBoxPanelExample

} // fim do namespace GroupBoxPanelExample

KeyDemo – O programa mostra as informações das teclas pressionadas. Relaciona-se a eventos que vem do teclado.

// Fig. 12.32: KeyDemo.cs

// Mostra informações das teclas pressionadas

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

namespace KeyDemo

{

/// <summary>

/// Formulário para mostrar tecla pressionada

/// Possui duas labels

/// </summary>

public class KeyDemo : System.Windows.Forms.Form

{

private System.Windows.Forms.Label charLabel;

private System.Windows.Forms.Label keyInfoLabel;

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.Container components = null;

// construtor padrão

public KeyDemo()

{

InitializeComponent();

}

/// <summary>

/// Limpar recursos a ser usados

/// </summary>

protected override void Dispose( bool disposing )

{

if ( disposing )

{

if ( components != null )

{

components.Dispose();

}

}

base.Dispose( disposing );

}

[STAThread]

static void Main()

{

Application.Run( new KeyDemo() );

}

// mostra o caracter pressionado usando uma tecla de caracter

protected void KeyDemo\_KeyPress(

object sender, System.Windows.Forms.KeyPressEventArgs e )

{

charLabel.Text = "Key pressed: " + e.KeyChar;

}

// Mostrar teclas modificadas, codigos de tecla, key data and valor da tecla

private void KeyDemo\_KeyDown(

object sender, System.Windows.Forms.KeyEventArgs e )

{

keyInfoLabel.Text =

"Alt: " + (e.Alt ? "Yes" : "No") + '\n' +

"Shift: " + (e.Shift ? "Yes" : "No" ) + '\n' +

"Ctrl: " + (e.Control ? "Yes" : "No" ) + '\n' +

"KeyCode: " + e.KeyCode + '\n' +

"KeyData: " + e.KeyData + '\n' +

"KeyValue: " + e.KeyValue;

}

// limpar labels quando a tecla for solta

private void KeyDemo\_KeyUp(

object sender, System.Windows.Forms.KeyEventArgs e )

{

keyInfoLabel.Text = "";

charLabel.Text = "";

}

} // fim da class KeyDemo

}

LabelTextBoxButtonTest – O programa utiliza uma label, uma TextBox e um botão. A label mostra o que foi digitado da TextBox depois que for pressionado o botão.

// Fig. 12.xx: LabelTextBoxButtonTest.cs

// Usando uma TextBox, Label e Botão para mostrar

// o texto escondido numa caixa de senha.

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

// namespace contém nosso formulário para mostrar os texto escondido

namespace LabelTextBoxButtonTest

{

/// <summary>

/// formulário que cria uma textbox para senha e

/// uma label para mostrar o conteudo da textbox

/// </summary>

public class LabelTextBoxButtonTest :

System.Windows.Forms.Form

{

private System.Windows.Forms.Button displayPasswordButton;

private System.Windows.Forms.Label displayPasswordLabel;

private System.Windows.Forms.TextBox inputPasswordTextBox;

/// <summary>

/// Variavel de designer requerida

/// </summary>

private System.ComponentModel.Container components = null;

// construtor padrão

public LabelTextBoxButtonTest()

{

InitializeComponent();

}

/// <summary>

/// Limpar todos os recursos a ser usados

/// </summary>

protected override void Dispose( bool disposing )

{

if ( disposing )

{

if ( components != null )

{

components.Dispose();

}

}

base.Dispose( disposing );

}

/// <summary>

/// Ponto de partida da aplicação

/// </summary>

[STAThread]

static void Main()

{

Application.Run( new LabelTextBoxButtonTest() );

}

// Mostra o que o usuário digitou na label

protected void displayPasswordButton\_Click(

object sender, System.EventArgs e )

{

// texto não mudou

displayPasswordLabel.Text =

inputPasswordTextBox.Text;

}

} // fim da class LabelTextBoxButtonTest

} // fim da namespace LabelTextBoxButtonTest

MouseTracker – Demonstração de eventos com mouse (clicks, pressinar e movimentos).

// Fig. 12.xx: MouseTracker.cs

// Demonstração de eventos com mouse (clicks, pressinar e movimentos).

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

namespace MouseTracker

{

/// <summary>

/// Summary description for Form1.

/// </summary>

public class MouseTracker : System.Windows.Forms.Form

{

private System.Windows.Forms.Label outputLabel;

/// <summary>

/// Variavel de designer requerida

/// </summary>

private System.ComponentModel.Container components = null;

public MouseTracker()

{

//

// requerido por Windows Form Designer suporte

//

InitializeComponent();

//

// TODO: adcionar qualquer código de construtor depois da chamada de InitializeComponent

//

}

/// <summary>

/// Clean up any resources being used.

/// </summary>

protected override void Dispose( bool disposing )

{

if( disposing )

{

if (components != null)

{

components.Dispose();

}

}

base.Dispose( disposing );

}

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.Run( new MouseTracker() );

}

// manipulador de evento quando o mouse é pressionado e depois solto

protected void MouseTracker\_MouseUp( object sender,

System.Windows.Forms.MouseEventArgs e )

{

outputLabel.Text = "Pressed at [" + e.X + "," + e.Y + "]";

}

// manipulador de evento quando o mouse move

protected void MouseTracker\_MouseMove( object sender,

System.Windows.Forms.MouseEventArgs e )

{

outputLabel.Text = "Moved at [" + e.X + "," + e.Y + "]";

}

// manipulador de evento quando o mouse sai do formulário

protected void MouseTracker\_MouseLeave( object sender,

System.EventArgs e )

{

outputLabel.Text = "Mouse outside Window";

}

// manipulador de evento quando o mouse entra no formulário

protected void MouseTracker\_MouseEnter( object sender,

System.EventArgs e )

{

outputLabel.Text = "Mouse in Window";

}

// manipulador de evento quando o mouse é clickado

protected void MouseTracker\_MouseDown( object sender,

System.Windows.Forms.MouseEventArgs e )

{

outputLabel.Text = "Clicked at [" + e.X + "," + e.Y + "]";

}

} // fim da class class MouseTracker

}

Paiter

// Fig 12.30: Painter.cs

// Usando o mouse para desenhar uma forma

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

namespace Painter

{

/// <summary>

/// creates a form as a drawing surface

/// </summary>

public class Painter : System.Windows.Forms.Form

{

bool shouldPaint = false; // se pintar

/// <summary>

/// Variavel de designer requerida

/// </summary>

private System.ComponentModel.Container components = null;

// construtor padrão

public Painter()

{

InitializeComponent();

}

/// <summary>

/// Clean up any resources being used.

/// </summary>

protected override void Dispose( bool disposing )

{

if ( disposing )

{

if ( components != null )

{

components.Dispose();

}

}

base.Dispose( disposing );

}

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.Run( new Painter() );

}

// deve pintar depois do botão do mouse ter sido pressionado

private void Painter\_MouseDown(

object sender, System.Windows.Forms.MouseEventArgs e )

{

shouldPaint = true;

}

// parar de pintar depois que o botão mouse for solto

private void Painter\_MouseUp(

object sender, System.Windows.Forms.MouseEventArgs e )

{

shouldPaint = false;

}

// desenhar um circulo sempre que o botão do mouse

// movimentar (e o mouse é clickado)

protected void Painter\_MouseMove(

object sender, System.Windows.Forms.MouseEventArgs e )

{

if ( shouldPaint )

{

Graphics graphics = CreateGraphics();

graphics.FillEllipse(

new SolidBrush( Color.BlueViolet ),

e.X, e.Y, 4, 4 );

}

} // fim Painter\_MouseMove

} // fim da class Painter

} // fim do namespace Painter

PictureBoxTest

// Fig. 12.28: PictureBoxTest.cs

// Usando PictureBox para mostrar imagens.

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

using System.IO;

namespace PictureBoxTest

{

/// <summary>

/// formulário para mostrar diferentes imagens quando clickado

/// </summary>

public class PictureBoxTest : System.Windows.Forms.Form

{

private System.Windows.Forms.PictureBox imagePictureBox;

private System.Windows.Forms.Label promptLabel;

private int imageNum = -1;

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.Container components = null;

// construtor padrÃ£o

public PictureBoxTest()

{

InitializeComponent();

}

/// <summary>

/// Clean up any resources being used.

/// </summary>

protected override void Dispose( bool disposing )

{

if ( disposing )

{

if ( components != null )

{

components.Dispose();

}

}

base.Dispose( disposing );

}

/// <summary>

/// Ponto de partida da aplicação

/// </summary>

[STAThread]

static void Main()

{

Application.Run( new PictureBoxTest() );

}

// muda a imagem sempre que a PictureBox é clickada

private void imagePictureBox\_Click(

object sender, System.EventArgs e )

{

imageNum = ( imageNum + 1 ) % 3; // imageNum de 0 a 2

// cria Image object do arquivo, mostra na PictureBox

imagePictureBox.Image = Image.FromFile(

Directory.GetCurrentDirectory() + "\\images\\image" +

imageNum + ".bmp" );

}

} // fim da class PictureBoxTest

} // fim da namespace PictureBoxTest

RadioButtonsTest

// Fig. 12.26: RadioButtonsTest.cs

// Usando RadioButtons para definir as opcoes de caixa de mensagem.

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

namespace RadioButtonsTest

{

/// <summary>

/// o formulário contém muitos RadioButtons- o usuário escolhe um

/// de cada grupo para definir uma caixa de mensagem customizada

/// </summary>

public class RadioButtonsTest : System.Windows.Forms.Form

{

private System.Windows.Forms.Label promptLabel;

private System.Windows.Forms.Label displayLabel;

private System.Windows.Forms.Button displayButton;

private System.Windows.Forms.RadioButton questionButton;

private System.Windows.Forms.RadioButton informationButton;

private System.Windows.Forms.RadioButton exclamationButton;

private System.Windows.Forms.RadioButton errorButton;

private System.Windows.Forms.RadioButton retryCancelButton;

private System.Windows.Forms.RadioButton yesNoButton;

private System.Windows.Forms.RadioButton yesNoCancelButton;

private System.Windows.Forms.RadioButton okCancelButton;

private System.Windows.Forms.RadioButton okButton;

private System.Windows.Forms.RadioButton

abortRetryIgnoreButton;

private System.Windows.Forms.GroupBox iconTypeGroupBox;

private System.Windows.Forms.GroupBox buttonTypeGroupBox;

private MessageBoxIcon iconType = MessageBoxIcon.Error;

private MessageBoxButtons buttonType =

MessageBoxButtons.OK;

/// <summary>

/// Variavel de designer requerida

/// </summary>

private System.ComponentModel.Container components = null;

// // construtor padrÃ£o

public RadioButtonsTest()

{

InitializeComponent();

}

/// <summary>

/// Clean up any resources being used.

/// </summary>

protected override void Dispose( bool disposing )

{

if ( disposing )

{

if ( components != null )

{

components.Dispose();

}

}

base.Dispose( disposing );

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.informationButton =

new System.Windows.Forms.RadioButton();

this.buttonTypeGroupBox =

new System.Windows.Forms.GroupBox();

this.retryCancelButton =

new System.Windows.Forms.RadioButton();

this.yesNoButton =

new System.Windows.Forms.RadioButton();

this.yesNoCancelButton =

new System.Windows.Forms.RadioButton();

this.abortRetryIgnoreButton =

new System.Windows.Forms.RadioButton();

this.okCancelButton =

new System.Windows.Forms.RadioButton();

this.okButton =

new System.Windows.Forms.RadioButton();

this.iconTypeGroupBox = new System.Windows.Forms.GroupBox();

this.questionButton =

new System.Windows.Forms.RadioButton();

this.exclamationButton =

new System.Windows.Forms.RadioButton();

this.errorButton =

new System.Windows.Forms.RadioButton();

this.displayLabel = new System.Windows.Forms.Label();

this.displayButton =

new System.Windows.Forms.Button();

this.promptLabel = new System.Windows.Forms.Label();

this.buttonTypeGroupBox.SuspendLayout();

this.iconTypeGroupBox.SuspendLayout();

this.SuspendLayout();

//

// informationButton

//

this.informationButton.Location =

new System.Drawing.Point( 16, 104 );

this.informationButton.Name = "informationButton";

this.informationButton.Size =

new System.Drawing.Size( 100, 23 );

this.informationButton.TabIndex = 4;

this.informationButton.Text = "Information";

this.informationButton.CheckedChanged +=

new System.EventHandler(

this.iconType\_CheckedChanged );

//

// buttonTypeGroupBox

//

this.buttonTypeGroupBox.Controls.AddRange(

new System.Windows.Forms.Control[] {

this.retryCancelButton,

this.yesNoButton,

this.yesNoCancelButton,

this.abortRetryIgnoreButton,

this.okCancelButton,

this.okButton } );

this.buttonTypeGroupBox.Location =

new System.Drawing.Point( 16, 56 );

this.buttonTypeGroupBox.Name = "buttonTypeGroupBox";

this.buttonTypeGroupBox.Size =

new System.Drawing.Size( 152, 272 );

this.buttonTypeGroupBox.TabIndex = 0;

this.buttonTypeGroupBox.TabStop = false;

this.buttonTypeGroupBox.Text = "Button Type";

//

// retryCancelButton

//

this.retryCancelButton.Location =

new System.Drawing.Point( 16, 224 );

this.retryCancelButton.Name = "retryCancelButton";

this.retryCancelButton.Size =

new System.Drawing.Size( 100, 23 );

this.retryCancelButton.TabIndex = 4;

this.retryCancelButton.Text = "RetryCancel";

// all radio buttons for button types are registered

// to buttonType\_CheckedChanged event handler

this.retryCancelButton.CheckedChanged +=

new System.EventHandler(

this.buttonType\_CheckedChanged );

//

// yesNoButton

//

this.yesNoButton.Location = new System.Drawing.Point( 16, 184 );

this.yesNoButton.Name = "yesNoButton";

this.yesNoButton.Size = new System.Drawing.Size( 100, 23 );

this.yesNoButton.TabIndex = 0;

this.yesNoButton.Text = "YesNo";

this.yesNoButton.CheckedChanged +=

new System.EventHandler(

this.buttonType\_CheckedChanged );

//

// yesNoCancelButton

//

this.yesNoCancelButton.Location =

new System.Drawing.Point( 16, 144 );

this.yesNoCancelButton.Name = "yesNoCancelButton";

this.yesNoCancelButton.Size =

new System.Drawing.Size( 100, 23 );

this.yesNoCancelButton.TabIndex = 3;

this.yesNoCancelButton.Text = "YesNoCancel";

this.yesNoCancelButton.CheckedChanged +=

new System.EventHandler(

this.buttonType\_CheckedChanged );

//

// abortRetryIgnoreButton

//

this.abortRetryIgnoreButton.Location =

new System.Drawing.Point( 16, 104 );

this.abortRetryIgnoreButton.Name =

"abortRetryIgnoreButton";

this.abortRetryIgnoreButton.Size =

new System.Drawing.Size( 120, 23 );

this.abortRetryIgnoreButton.TabIndex = 2;

this.abortRetryIgnoreButton.Text = "AbortRetryIgnore";

this.abortRetryIgnoreButton.CheckedChanged +=

new System.EventHandler(

this.buttonType\_CheckedChanged );

//

// okCancelButton

//

this.okCancelButton.Location =

new System.Drawing.Point( 16, 64 );

this.okCancelButton.Name = "okCancelButton";

this.okCancelButton.Size =

new System.Drawing.Size( 100, 23 );

this.okCancelButton.TabIndex = 1;

this.okCancelButton.Text = "OKCancel";

this.okCancelButton.CheckedChanged +=

new System.EventHandler(

this.buttonType\_CheckedChanged );

//

// okButton

//

this.okButton.Checked = true;

this.okButton.Location =

new System.Drawing.Point( 16, 24 );

this.okButton.Name = "okButton";

this.okButton.Size =

new System.Drawing.Size( 100, 23 );

this.okButton.TabIndex = 0;

this.okButton.TabStop = true;

this.okButton.Text = "OK";

this.okButton.CheckedChanged +=

new System.EventHandler(

this.buttonType\_CheckedChanged );

//

// iconTypeGroupBox

//

this.iconTypeGroupBox.Controls.AddRange(

new System.Windows.Forms.Control[] {

this.questionButton,

this.informationButton,

this.exclamationButton,

this.errorButton } );

this.iconTypeGroupBox.Location =

new System.Drawing.Point( 200, 56 );

this.iconTypeGroupBox.Name = "iconTypeGroupBox";

this.iconTypeGroupBox.Size =

new System.Drawing.Size( 136, 176 );

this.iconTypeGroupBox.TabIndex = 1;

this.iconTypeGroupBox.TabStop = false;

this.iconTypeGroupBox.Text = "Icon";

//

// questionButton

//

this.questionButton.Location =

new System.Drawing.Point( 16, 144 );

this.questionButton.Name = "questionButton";

this.questionButton.Size =

new System.Drawing.Size( 100, 23 );

this.questionButton.TabIndex = 0;

this.questionButton.Text = "Question";

// all radio buttons for icon types are registered

// to iconType\_CheckedChanged event handler

this.questionButton.CheckedChanged +=

new System.EventHandler(

this.iconType\_CheckedChanged );

//

// exclamationButton

//

this.exclamationButton.Location =

new System.Drawing.Point( 16, 64 );

this.exclamationButton.Name = "exclamationButton";

this.exclamationButton.Size =

new System.Drawing.Size( 104, 23 );

this.exclamationButton.TabIndex = 2;

this.exclamationButton.Text = "Exclamation";

this.exclamationButton.CheckedChanged +=

new System.EventHandler(

this.iconType\_CheckedChanged );

//

// errorButton

//

this.errorButton.Location =

new System.Drawing.Point( 16, 24 );

this.errorButton.Name = "errorButton";

this.errorButton.Size =

new System.Drawing.Size( 100, 23 );

this.errorButton.TabIndex = 1;

this.errorButton.Text = "Error";

this.errorButton.CheckedChanged +=

new System.EventHandler(

this.iconType\_CheckedChanged );

//

// displayLabel

//

this.displayLabel.Location =

new System.Drawing.Point( 200, 304 );

this.displayLabel.Name = "displayLabel";

this.displayLabel.Size =

new System.Drawing.Size( 136, 24 );

this.displayLabel.TabIndex = 4;

//

// displayButton

//

this.displayButton.Location =

new System.Drawing.Point( 200, 240 );

this.displayButton.Name = "displayButton";

this.displayButton.Size =

new System.Drawing.Size( 136, 48 );

this.displayButton.TabIndex = 3;

this.displayButton.Text = "Display";

this.displayButton.Click +=

new System.EventHandler( this.displayButton\_Click );

//

// promptLabel

//

this.promptLabel.Font =

new System.Drawing.Font(

"Microsoft Sans Serif", 9.5F,

System.Drawing.FontStyle.Regular,

System.Drawing.GraphicsUnit.Point,

( ( System.Byte )( 0 ) ) );

this.promptLabel.Location =

new System.Drawing.Point( 8, 16 );

this.promptLabel.Name = "promptLabel";

this.promptLabel.Size =

new System.Drawing.Size( 344, 24 );

this.promptLabel.TabIndex = 5;

this.promptLabel.Text = "Choose the type of " +

"MessageBox you would like to display!";

//

// RadioButtonsTest

//

this.AutoScaleBaseSize =

new System.Drawing.Size( 5, 13 );

this.ClientSize =

new System.Drawing.Size( 360, 341 );

this.Controls.AddRange(

new System.Windows.Forms.Control[] {

this.promptLabel,

this.displayLabel,

this.displayButton,

this.iconTypeGroupBox,

this.buttonTypeGroupBox } );

this.Name = "RadioButtonsTest";

this.Text = "Demonstrating RadioButtons";

this.buttonTypeGroupBox.ResumeLayout( false );

this.iconTypeGroupBox.ResumeLayout( false );

this.ResumeLayout( false );

} // end method InitializeComponent

#endregion

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.Run( new RadioButtonsTest() );

}

// muda o botÃ£o baseado na opÃ§Ã£o escolhida pelo sender

private void buttonType\_CheckedChanged(

object sender, System.EventArgs e )

{

if ( sender == okButton ) // mostra botÃ£o de OK

buttonType = MessageBoxButtons.OK;

// mostra os botoes OK e Cancel

else if ( sender == okCancelButton )

buttonType = MessageBoxButtons.OKCancel;

// Mostra os botoes Abort, Retry e Ignore

else if ( sender == abortRetryIgnoreButton )

buttonType = MessageBoxButtons.AbortRetryIgnore;

// mostra os botoes Yes, No e Cancel

else if ( sender == yesNoCancelButton )

buttonType = MessageBoxButtons.YesNoCancel;

// Mostra os botoes Yes e No buttons

else if ( sender == yesNoButton )

buttonType = MessageBoxButtons.YesNo;

// Apenas uma opção sobrando-- mostra os

// botÃµes Retry e Cancel

else

buttonType = MessageBoxButtons.RetryCancel;

} // fim do metodo buttonType\_CheckedChanged

// muda o icone baseda na opcao escolhida pelo sender

private void iconType\_CheckedChanged(

object sender, System.EventArgs e )

{

if ( sender == errorButton ) // Mostra o icone de erro

iconType = MessageBoxIcon.Error;

// mostra o ponto de exclamacao

else if ( sender == exclamationButton )

iconType = MessageBoxIcon.Exclamation;

// mostra o icone de informa

else if ( sender == informationButton )

iconType = MessageBoxIcon.Information;

else // apenas uma opcao restante--mostra a interrogacao

iconType = MessageBoxIcon.Question;

} // fim do metodo iconType\_CheckedChanged

// mostra a MessageBox e botao que o usuario pressionou

protected void displayButton\_Click(

object sender, System.EventArgs e )

{

DialogResult result =

MessageBox.Show( "This is Your Custom MessageBox.",

"Custom MessageBox", buttonType, iconType, 0, 0 );

// checar dialog para obter um resultado e mostrar o resutado na label

switch ( result )

{

case DialogResult.OK:

displayLabel.Text = "OK was pressed.";

break;

case DialogResult.Cancel:

displayLabel.Text = "Cancel was pressed.";

break;

case DialogResult.Abort:

displayLabel.Text = "Abort was pressed.";

break;

case DialogResult.Retry:

displayLabel.Text = "Retry was pressed.";

break;

case DialogResult.Ignore:

displayLabel.Text = "Ignore was pressed.";

break;

case DialogResult.Yes:

displayLabel.Text = "Yes was pressed.";

break;

case DialogResult.No:

displayLabel.Text = "No was pressed.";

break;

} // fim switch

} // fim metodo displayButton\_Click

} // fim da class RadioButtonsTest

} // fim do namespace RadioButtonsTest