Gestão de Condutores e Veículos

Alaene Rufino de Sousa
609992

Sumário

- Descrição
- Tarefas e solução proposta

Gestão de Condutores&Veículos

Descrição

As autoridades que fiscalizam as regras de trânsito referentes à aplicação de multas e controle de permissão de direção (CNH) necessitam de um sistema que as ajudem a verificar se motoristas e automóveis se encontram ou não em situação regular. Você e seus companheiros de trabalho foram, assim, contratados para produzir este sistema.

Regras

• Cidadãos habilitados são condutores de veículos.

• A habilitação tem uma data de validade. Condutores com habilitação fora da validade são considerados irregulares.

• Veículos devem pagar três tipos de taxas anualmente: IPVA, seguro obrigatório e licenciamento. Um veículo só é regular se tiver toda as taxas pagas para os últimos 3 anos.

• Existem diversos tipos de multas que podem ser aplicadas condutores que conduzem veículos em uma data. Tais multas podem ser classificadas como leves, graves ou muito graves. Cada tipo de multa gera, respectivamente, 3, 5 e 7 pontos de penalização. Os valores das multas são, respectivamente, R$125, R$250 e R$500.

• Se um condutor somar mais que 21 pontos em um ano, está irregular.

Diagrama de Classe da Solução

Seu sistema deve então:

- Ler os dados acima de arquivos de condutores, veículos e multas.
- Listar os veículos que um condutor dirigiu e levou multa.
- Listar multas por veículo e criar um relatório de veículos sem multas.
- Responder se um veículo ou condutor está irregular.
- Mostrar o extrato de multas de um condutor.
- Ordenação das lista de todas as multas em ordem crescente de data

Ler os dados acima de arquivos de condutores, veículos e multas.

```
private void condutoresToolStripMenuItem_Click(object sender, EventArgs e)
{
    String[] cnhCondutores = File.ReadAllLines("condutores.txt");
    string cnhNum, nomeCondutor;
    DateTime dataVencimentoCnh;
    int cont = 0;

    for (int i = 0; i < cnhCondutores.Length; i++)
    {
        cnhNum = cnhCondutores[i].Split(';')[1];
        nomeCondutor = cnhCondutores[i].Split(';')[0];
        dataVencimentoCnh = Convert.ToDateTime(cnhCondutores[i].Split(';')[2]);

        CNH cnh = new CNH(cnhNum, dataVencimentoCnh);
        Condutores condutor = new Condutores(nomeCondutor, cnh);
        tabelaHashCondutores.inserir(condutor);
        cont++;
    }
    condutoresToolStripMenuItem.Enabled = false;
    MessageBox.Show("Foram Lidos: " + cont + " Condutores");
    condutoresFlag = true;
}
```

```
private void multasToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (condutoresFlag && veiculosFlag)
    {
        String[] multas = File.ReadAllLines("multas.txt");
        DateTime dataEmissaoMulta;
        string cnh, placa, tipoMulta;
        int cont = 0;

        for (int i = 0; i < multas.Length; i++)
        {
            cnh = multas[i].Split(';')[0];
            placa = multas[i].Split(';')[1];
            dataEmissaoMulta = Convert.ToDateTime(multas[i].Split(';')[3]);
            tipoMulta = multas[i].Split(';')[2];

            Condutores condutor = tabelaHashCondutores.buscar(cnh);
            Veiculos veiculo = tabelaHashVeiculos.buscar(placa);
            Multas multa = null;
            switch (tipoMulta)
            {
                case "1":
                    multa = new Leves(dataEmissaoMulta, veiculo, condutor);
                    break;

                case "2":
                    multa = new Medias(dataEmissaoMulta, veiculo, condutor);
                    break;

                case "3":
                    multa = new Graves(dataEmissaoMulta, veiculo, condutor);
                    break;
```

```
                default:
                    break;
            }
            tabelaHashMultas.inserir(multa);
            condutor.listaMultasPorCondutor.inserir(multa);
            veiculo.listaMultasPorVeiculo.inserir(multa);
            cont++;
        }
        multasToolStripMenuItem.Enabled = false;
        MessageBox.Show("Foram Lidas: " + cont + " Multas");
    }
    else
    {
        MessageBox.Show("Os dados de Veículos("+veiculosFlag+") ou Condutores("+condutoresFlag+") não foram carregados
    }
}
```

Listar os veículos que um condutor dirigiu e levou multa.

```
private void button1_Click(object sender, EventArgs e)
{
    if (cnhMaskTextBox.Text == "")
    {
        MessageBox.Show("Campo Vazio - Informe uma CNH");
    }
    else
    {
        veiculosComMultaPorCondutorlistView.Items.Clear();
        string cnhCondutor = cnhMaskTextBox.Text;
        Condutores condutor = condutoresHash.buscar(cnhCondutor);
        if (condutor != null)
        {
            VeiculosListas veiculosComMultaPorCondutor = condutor.GetVeiculosComMulta();
            VeiculosNo auxVeiculo = veiculosComMultaPorCondutor.sentinela;
            while (auxVeiculo.prox != null)
            {
                auxVeiculo = auxVeiculo.prox;
                veiculosComMultaPorCondutorlistView.Items.Add(auxVeiculo.veiculo.placa);
            }
        }
        else
        {
            MessageBox.Show("Condutor não encontrado");
        }

    }
}
```

Listar multas por veículo e criar um relatório de veículos sem multas.

```
private void listaMultaPorVeiculoButton_Click(object sender, EventArgs e)
{
    if (placaMaskedTextBox.Text == "")
    {
        MessageBox.Show("Campo Vazio - Informe uma Placa");
    }
    else
    {
        multasPorVeiculoListView.Items.Clear();
        string placa = placaMaskedTextBox.Text.ToUpper();
        Veiculos veiculo = veiculosHash.buscar(placa);

        if (veiculo != null)
        {
            Multas[] vetMulta = veiculo.listaMultasPorVeiculo.OrdenaMultas();

            foreach (var multa in vetMulta)
            {
                multasPorVeiculoListView.Items.Add(multa.condutor.habilitacaoCondutor.cnh + " " + multa.dataEmissao.ToShortDateString());
            }

            //MultasNo auxMultas = veiculo.listaMultasPorVeiculo.sentinela;
            //while (auxMultas.prox != null)
            //{
            //    auxMultas = auxMultas.prox;
            //    multasPorVeiculoListView.Items.Add(auxMultas.multa.condutor.habilitacaoCondutor.cnh + " " + auxMultas.multa.dataEmissao.T
            //}
        }
        else
        {
            MessageBox.Show("Veículo não encontrado");
        }
    }
```

```
private void RelatorioDeVeiculoSemMulta_Load(object sender, EventArgs e)
{
    for (int i = 0; i < veiculosHash.hashtab.Length; i++)
    {
        VeiculosNo auxVeiculosNo;
        auxVeiculosNo = veiculosHash.hashtab[i].sentinela;
        while (auxVeiculosNo.prox != null)
        {
            auxVeiculosNo = auxVeiculosNo.prox;
            if (auxVeiculosNo.veiculo.listaMultasPorVeiculo.vazia())
            {
                veiculosSemMultalistView.Items.Add(auxVeiculosNo.veiculo.placa);
            }
        }
    }
}
```

Responder se um veículo ou condutor está irregular.

```
private void verificaPlacaButton_Click(object sender, EventArgs e)
{
    if (placaMaskedTextBox.Text == "")
    {

        MessageBox.Show("Campo Vazio - Informe uma Placa");
    }
    else
    {
        string placa = placaMaskedTextBox.Text.ToUpper();
        Veiculos veiculo = veiculos.buscar(placa);
        if (veiculo != null)
        {
            if (veiculo.verificaRegularidadeVeiculo())
            {
                MessageBox.Show("Veículo Regular");
            }
            else
            {
                MessageBox.Show("Veículo Irregular");
            }
        }
        else
        {
            MessageBox.Show("Veículo não encontrado");
        }
    }
}
```

```
private void verificaCNHButton_Click(object sender, EventArgs e)
{
    if (cnhMaskedTextBox.Text == "")
    {
        MessageBox.Show("Campo Vazio - Informe uma CNH");
    }
    else
    {
        string cnhCondutor = cnhMaskedTextBox.Text;
        Condutores condutor = condutores.buscar(cnhCondutor);
        if (condutor != null)
        {
            if (condutor.verificarRegularidadeCNH())
            {
                MessageBox.Show("Condutor Regular");
            }
            else
            {
                MessageBox.Show("Condutor Irregular");
            }
        }
        else
        {
            MessageBox.Show("Condutor não encontrado");
        }
    }
}
```

Mostrar o extrato de multas de um condutor.

```
private void listarExtratoButton_Click(object sender, EventArgs e)
{
    listView1.Clear();
    if (cnhExtratoMaskTextBox.Text == "")
    {
        MessageBox.Show("Campo Vazio - Informe uma CNH");
    }
    else
    {
        string cnhCondutor = cnhExtratoMaskTextBox.Text;
        Condutores condutor = condutores.buscar(cnhCondutor);
        if (condutor != null)
        {
            label1.Text = condutor.nomeCondutor;
            Multas[] vetMulta = condutor.listaMultasPorCondutor.OrdenaMultas();

            foreach (var multa in vetMulta)
            {
                listView1.Items.Add(multa.dataEmissao.ToShortDateString() + " " + multa.veiculo.placa);
            }

            //MultasNo auxMultas = condutor.listaMultasPorCondutor.sentinela;
            //while (auxMultas.prox != null)
            //{
            //    auxMultas = auxMultas.prox;
            //    listView1.Items.Add(condutor.nomeCondutor + " " + auxMultas.multa.veiculo.placa + " " + auxMultas.multa.dataEm
            //}
        }
        else
        {
            MessageBox.Show("Condutor não encontrado!", "Condutor não encontrado", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

    }

}
```

Ordenação das lista de todas as multas em ordem crescente de data.

```
class QuickSort
{
        public static Multas[] quickSort(Multas[] vetor)
        {
            int inicio = 0;
            int fim = vetor.Length - 1;
            quickSort(vetor, inicio, fim);

            return vetor;
        }

        private static void quickSort(Multas[] vetor, int inicio, int fim)
        {
            if (inicio < fim)
            {
                Multas p = vetor[inicio];
                int i = inicio + 1;
                int f = fim;
                while (i <= f)
                {
```

```
            if (vetor[i].dataEmissao <= p.dataEmissao)
            {
                i++;
            }
            else if (p.dataEmissao < vetor[f].dataEmissao)
            {
                f--;
            }
            else
            {
                Multas troca = vetor[i];
                vetor[i] = vetor[f];
                vetor[f] = troca;
                i++;
                f--;
            }
        }
        vetor[inicio] = vetor[f];
        vetor[f] = p;
        quickSort(vetor, inicio, f - 1);
        quickSort(vetor, f + 1, fim);
    }
}
}
```

```
internal Multas[] OrdenaMultas()
{
    Multas[] vet = this.ListaParaVetor();
    return QuickSort.quickSort(vet);
}

private Multas[] ListaParaVetor()
{
    MultasNo aux = sentinela;
    int cont = 0;
    while (aux.prox != null)
    {
        aux = aux.prox;
        cont++;
    }

    Multas[] ret = new Multas[cont];

    aux = sentinela;
    cont = 0;
    while (aux.prox != null)
    {
        aux = aux.prox;
        ret[cont] = aux.multa;
        cont++;
    }
    return ret;
}
```

FIM! Go to Execution -->

Algoritmos e Estruturas de Dados

2/2018 - João Caram

GitHub: https://github.com/alaeners/AED/tree/TI_FINAL_AED/TI_FINAL_AED