**\*\*\*\*\* Exercício - POO \*\*\*\*\***

**Classes e Objetos**

1. Crie uma classe chamada Jogo com seguintes atributos:

* Código - inteiro
* Nome - string
* Categoria – string
* Data de lançamento - DateTime

Instancie um objeto da classe Jogo, preencha os seus atributos e exiba em vídeo os dados.

1. Utilizando a classe jogo criada no exercício anterior, faça um programa que instancie até 10 jogos. Após efetuar os cadastros, exiba os dados em vídeo.
2. Crie uma classe para representar um funcionário. A classe deve ter os seguintes atributos:

Código

Nome

RG

Salario

Inss (somente leitura: não tem o método SET)

Crie também os seguintes métodos:

CalcularSalario()

Este método deverá pegar o salário do funcionário e:

subtrair 11% do inss e retornar o salário subtraído do inss.

O método getINSS deve calcular o valor do inss (11% do salario) e devolver este valor.

Crie um formulário para preencher os campos do cadastro.

Crie um botão para exibir os valores gravados na classe, como foi feito no exemplo do aluno.

**Exercício sobre propriedades**

1. Altere o exercício anterior para que ele use propriedades.
2. Crie uma classe chamada Aluno

Crie os seguintes atributos

string Nome

double nota1

double nota2

Crie propriedades para acessar os atributos da classe

Crie uma propriedade chamada Media que será somente leitura e devolverá a média aritmética (nota1 + nota2 ) /2

Nota válida deve estar entre 0 e 10

Nome válido deve possuir nome e sobrenome.

Crie um formulário para instanciar o aluno um botão para exibir a média.

1. Altere o exercício anterior para que ele permita salvar até 10 alunos.

**Exercício de sobrecarga de métodos**

1. Crie um método para somar 2 valores e devolver o resultado. O método pode receber como parâmetro tanto valores inteiros como string. Crie um programa teste para testar os 2 métodos.

Faça método sobrecarregado para salvar em um arquivo texto o conteúdo passado por parâmetro.

Quando não for passado por parâmetro o caminho, salve na pasta da aplicação.

Crie as seguintes versões do método:

Salva (string texto)

Salva (string texto, string caminho)

Salva(string[] vetor)

Salva(string[] vetor, string caminho)

Altere o EX 1 adicionando um construtor parametrizado. Altere o programa principal para que ele utilize este construtor.

\*\*\*\* exercício sobre sobrecarga e construtores \*\*\*\*\*\*

Crie uma classe chamada ContaCorrente

A classe deverá ter os seguintes atributos:

agencia int

numero da conta string

nometitular string

Saldo (apenas leitura) double (inicia com zero)

Deverá haver um construtor parametrizado para que seja possível passar todos os parâmetros

já na criação do objeto. Deverá haver os métodos:

\* void Saque (double valor) - subtrai do saldo. Deve gerar uma exceção caso o saldo não seja suficiente.

\* void Deposito (double valor) - acrescenta o valor ao saldo.

EX 11

\*\*\*\*\* LISTAS \*\*\*\*\*

Altere o exercício 3 para que os dados sejam salvos em uma lista.

Crie um botão para pesquisar um aluno pelo nome.

Crie um botão para listar todos os alunos, exibindo todas as propriedades.

Crie um botão para remover um aluno da lista.

EX\_12

\*\*\*\*\* Métodos, atributos e classes estáticas e enumeradores \*\*\*\*\*

Crie um enumerador para representar o sexo (Masculino e Feminino) e outro para representar os estados civis (casado, solteiro, viúvo) de um funcionário.

Crie a classe funcionário com os seguintes atributos:

string nome

SexoEnum sexo

ECivilEnum estado civil

datetime data de nascimento

Crie uma classe chamada métodos e crie os seguintes métodos estáticos:

bool validanome (string nome) - deverá validar se o nome possui nome e sobrenome.

bool validaData (string data, out idade) - deverá ver se a data informada é valida. Se for válida, retornar a idade no parâmetro de saída.

Crie um construtor padrão e outro parametrizado onde será possível informar todos os atributos.

Crie um formulário e valide os dados utilizando estes métodos

EX\_13

Crie uma Interface ICadatro que possua os métodos Salvar e Pesquisar. As assinaturas dos métodos são:

Void Salvar(string conteudo);

String Pesquisar(string valor);

Crie 2 classes: Funcionario e Produto e implemente esta interface em ambos.

Atributos:

Funcionario: codigo: int, string nome

Produto: codigo: int, string nome, double preco.

O método salvar de ambos deve salvar os dados em um arquivo texto (dados.txt), acumulando os registros, um por linha, de forma que fique assim no arquivo:

F|codigo|nome 🡪 *para nomes*

P|codigo|nome|preco 🡪 *para produtos*

Faça a interface gráfica para cadastrar produtos e funcionários.

A pesquisa se dará pelo nome. Preencha os campos na tela após efetuar a pesquisa.

EX\_13

\*\*\* Exercício de treinamento para prova difícil \*\*\*

Crie uma classe para representar jogos.

Deve haver os seguintes campos:

Codigo : int ( > 0)

Descricao : string (obrigatorio)

Dificuldade: Enumerador(easy, normal, hard) (obrigatorio)

Valor : double (entre 1 e 10,00)

Fabricante : Obrigatório. Exiba Uma lista de fabricantes e permita que o usuário selecione 1 valor da lista.

Crie a classe fabricante:

Fabricante ( int codigo, string nome)

Crie uma lista com os seguintes fabricantes (pode ser fixo, no código)

{1 – Konami} { 2 – Capcom } { 3 - Nintendo } { 4 – EA} {5 – Acclaim} { 6 – IronHide } { 7 – SEGA }

Crie formulário para teste da classe um método para carregar os dados do arquivo texto e preencher os atributos do objeto com os valores lidos, jogando os objetos em uma lista.

No formulário para teste da classe, inclua os botões para salvar a Lista (que irá salvar todos os objetos da lista em um arquivo texto, usando para isso o salvar que há disponível na classe Jogo) e um botão para carregar, que irá preencher uma lista com os objetos carregados do arquivo.

Deve haver um botão para listar todos os jogos cadastrados (verificar na lista).

O mesmo botão deve listar apenas os jogos de uma determinada dificuldade caso seja selecionada a dificuldade em uma caixa combo que ficará ao lado do botão. Para estes 2 utilize sobrecarga de métodos (métodos com o mesmo nome porém com parâmetros diferentes)

EX\_14

Faça um programa que leia de um arquivo texto as categorias de um produto.

As categorias são: (categorias.txt)

1,brinquedos

2,alimentos

3,bebidas

4,eletrodomésticos

5,eletrônicos

Faça um cadastro de produtos com os seguintes atributos: código, descrição, categoria. A categoria deve estar em um combobox.

Guarde os produtos em uma lista.

Depois, liste todos os produtos em um listbox, listando o código, a descrição e a descrição da categoria.

EX\_8.1

Crie uma classe básica funcionário com os seguintes atributos:

Código int

Nome string

Salário: Double

CalculoSalario() método que calcula o salário.

Crie uma classe chamada FuncionarioPiao que herda de funcionário e terá, adicionalmente:

Double HoraExtra. Este campo deve entrar no cálculo do salário.

Crie uma classe chamada FuncionarioGerente que herda de funcionário e terá, adicionalmente:

Double Bonus . Este campo deve entrar no cálculo do salário.

Int QtdeFuncionariosSubordinados

Crie uma classe chamada FuncionarioVendedor que herda de funcionário e terá, adicionalmente:

Double MetaDeVendaMes

Double VendasdoMes

Double PorcentagemSobreVendas (para incorporar no salário)

Crie um formulário para instanciar as 3 classes derivadas, e guarde em uma lista. Crie um botão para exibir as propriedades e o salário final de cada funcionário instanciado.

EX\_9

Exercício: Banco

Abstract Conta Bancaria : Object

Atributos:

String Nome do cliente

String Número da conta (formato 9999-9)

Double Saldo

Métodos:

void Deposito(double valor);

void Saque(double valor);

ContaCorrente: ContaBancaria

Atributos:

Double Limite de crédito

Bool UtilizaTalaoCheque

Métodos:

void Saque(double valor); - O saldo + limite de crédito não pode ser < 0

ContaPoupanca: ContaBancaria

Atributos:

Int Dia de aniversário:

string Nº da c.Corrente atrelada

Métodos:

void Saque(double valor); - O saldo não pode ser < 0

Faça uma tela para cadastrar conta corrente e poupança. Salve em uma lista.

Crie um botão para depositar e 1 botão para sacar. Para tanto, o usuário deverá informar o nº da conta e o sistema irá efetuar a operação na conta selecionada.

Sobrescreva o método toString() para listar todas as informações da conta.

Faça uma opção para listar todas as contas cadastradas (cc. ou poupança) informando inclusive o tipo da conta (Ex: Conta corrente / Poupança)

EX\_11

\*\*\*\*\*\*\*\*\*\*\*\*\*\* herança \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Baseado no exemplo do aluno, faça:

\* coloque mais 3 botões para listagem:

- apenas dos alunos do EM.

- apenas dos alunos da FTT.

- apenas dos alunos da Pós.

\* Crie um atributo na classe Aluno do tipo int

para guardar a quantidade de faltas do aluno

\* Crie um método na classe Aluno para calcular a freqüência do aluno. O método deverá devolver

a porcentagem de freqüência do aluno com base na seguinte formula: 80 aulas = 100%. Subtraia as faltas

para saber qual foi a freqüência do aluno.

\* Crie um método situação que irá devolver um enumerador que indica se está aprovado ou reprovado. Ele deve levar em consideração a média e também a freqüência do aluno( >= 75) para retornar se ele está aprovado ou não. Ou seja, para estar aprovado é necessário tirar a nota mínima do curso e também ter freqüência maior ou igual a 75.

EX\_12

Crie uma Interface ICadatro que possua os métodos Salvar e Pesquisar. As assinaturas dos métodos são:

Void Salvar(string conteudo);

String Pesquisar(string valor);

Crie 2 classes: Funcionario e Produto e implemente esta interface em ambos.

Atributos:

Funcionario: código: int, string nome

Produto: código: int, string nome, double preco.

O método salvar de ambos deve salvar os dados em um arquivo texto (dados.txt), acumulando os registros, um por linha, de forma que fique assim no arquivo:

F|codigo|nome para nomes

P|codigo|nome|preco para produtos

Faça a interface gráfica para cadastrar produtos e funcionários.

A pesquisa se dará pelo nome.

EX\_14

Crie as classes em C# (modo Windows forms) que melhor representem as necessidades abaixo. Será analisada a capacidade do aluno de abstrair e utilizar da melhor forma os conceitos de orientação a objetos.

O modelo abaixo tem por objetivo representar automóveis, que podem ser carros, caminhões, ônibus e até motos.

Independentemente do tipo de veículo automotor, todos eles devem possuir os seguintes atributos e métodos:

Atributos:

string descricao

double capacidadeMaximaEmKg

double velocidadeAtualEmKM (atributo apenas leitura)

double capacidadeCarregadaemKg (atributo apenas leitura)

Métodos:

double Carregar (double peso)// -> gerar exceção (personalizada) se exceder a capacidadeMaximaEmKg//. Exibir em vídeo a capacidade informada após carregar.

double Acelerar() ; // aumenta em 1 km a velocidade atual do veículo e exibe no console a nova velocidade

double PagarPedagio(); // seu cálculo depende de características do tipo de veículo. Todo veículo deve pagar pedágio.

o método ToString()// deverá exibir todos os atributos da classe, concatenados.

Obs: um veículo automotor não deve ser instanciado. Apenas os seus descendentes podem ser instanciados.

Segue abaixo uma descrição dos diversos tipos de automóveis

Um carro é um tipo de veículo automotor que possui, além dos atributos e métodos de um veículo automotor, adicionalmente:

Atributos:

int quantidade de portas.

bool UtilizandoReboque ;

Métodos:

O método ToString deve refletir todos os atributos da classe.

No método de cálculo do pedágio: caso esteja usando reboque cobrar R$ 10,00. Caso contrário, cobrar R$7,00. Exibir o valor pago em vídeo.

Uma moto é um tipo de veículo automotor que possui, além dos atributos e métodos de um veículo automotor, adicionalmente:

Atributos:

int cilindradas

Métodos

string Empinar() -> exibe no console "Empinando...";

O método ToString deve refletir todos os atributos da classe.

Cálculo do pedágio: Para moto, considerar o valor R$ 2,00. Exibir o valor pago em vídeo

Um caminhão é um tipo de veículo automotor que possui, além dos atributos e métodos de um veículo automotor, adicionalmente:

Atributos:

int QuantidadeEixos;

Métodos

double Carregar(double peso) -> exibe no console "Carregado". Caso o peso ultrapasse o peso máximo, NÃO gere uma exceção, mas exiba em vídeo a mensagem “Sobrecarregado”;

O método Acelerar não deve acelerar o caminhão caso o peso carregado no caminhão ultrapasse o peso máximo. Neste caso, gere uma exceção com o texto “Sobrecarregado!”.

O método ToString deve refletir todos os atributos da classe.

Crie o método Descarregar() , que irá retirar todo o peso e escrever "Vazio" no console;

Cálculo do pedágio: Considerar o valor R$ 5,00 por eixo. Exibir o valor pago em vídeo.

Crie também botões para executar os métodos específicos de cada classe, como o descarregar do caminhão.

Esses métodos deverão atuar sobre 1 objeto na lista. Para isso, você precisará primeiro efetuar uma pesquisa pelo nome do veículo, achá-lo na lista e então executar o método escolhido.

Após terminado o sistema, foi solicitado que fossem incluídas algumas funcionalidades no carro e no caminhão:

int VelocidadeLimpador

void AcionarLimpador();

void AbrirPorta();

Implemente estas funcionalidades nos veículos onde elas se aplicam e crie os botões para testá-las.

EX\_15

Crie uma lista que seja capaz de armazenar qualquer tipo de objeto.

utilize-a depois para armazenar alunos (utilize o exemplo dos alunos, substituindo a lista do C# pela sua lista)