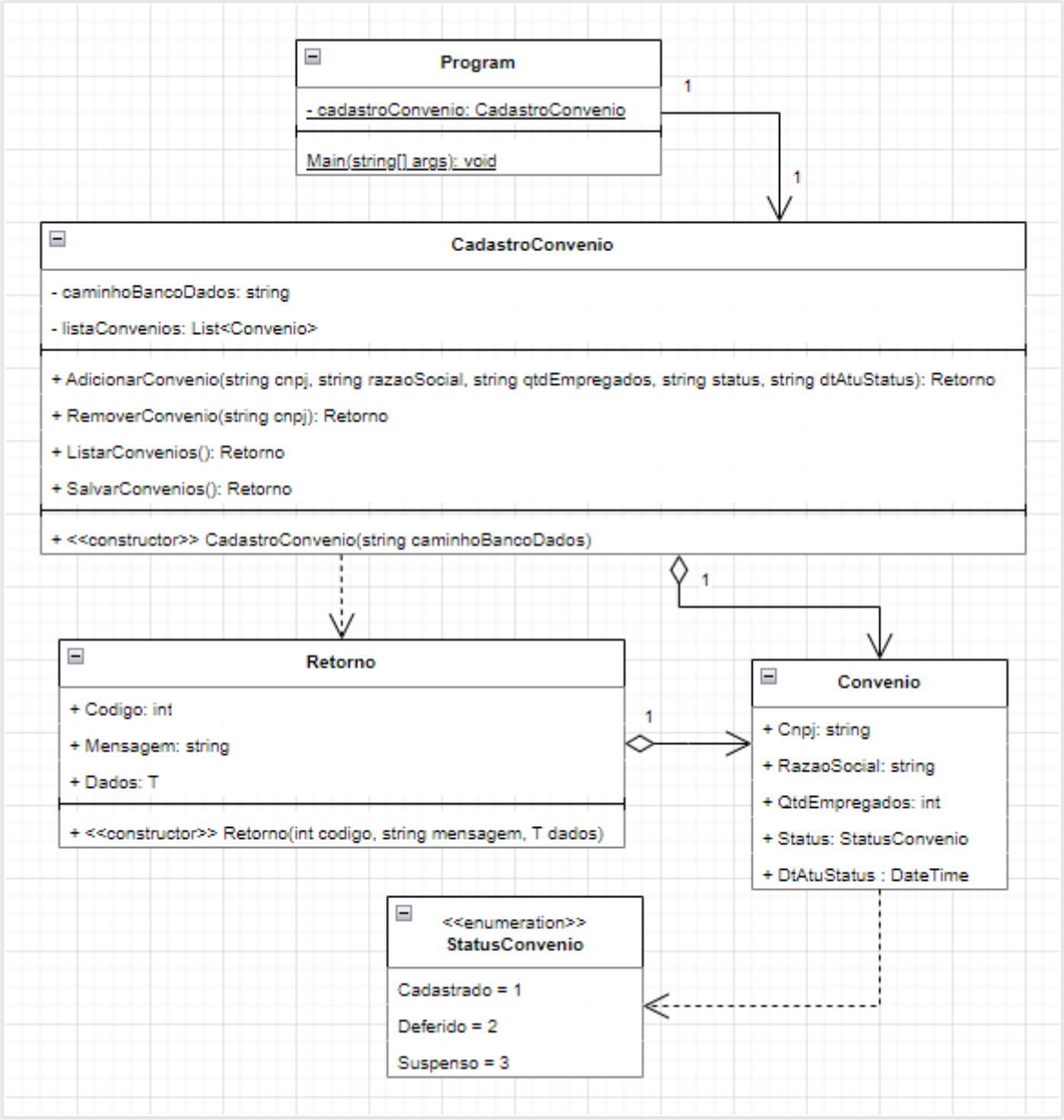
Avaliação de C#

Cadastro de Convênios para Crédito Consignado

Para que uma empresa possa fornecer empréstimos consignados (com desconto em folha) para seus funcionários em parceria com o Banrisul, é necessário que um seja feito cadastro específico. Abaixo segue uma definição simplificada deste cadastro. Você deve criar uma aplicação a partir desta definição.

Diagrama de Classes

No diagrama abaixo estão todas as classes, atributos, propriedades e métodos obrigatórios. O padrão de nomenclatura do diagrama deve ser seguido na implementação. Todas as classes devem estar contidas no namespace Bergs.AvaliacaoCSharp. Elementos privados adicionais ficam conforme a sua organização pessoal.



Legenda:
- “-”: Elemento privado;
- “+”: Elemento público;
- Elemento sublinhado: Elemento estático.

Criar um console application com nome AvaliacaoCSharp, preferencialmente na pasta C:\desenvhome-pxc\. Desmarcar a opção Create directory for solution. Utilizar o framework .NET Framework 4.8. Nas propriedades do projeto, em Build, selecione em Platform target a opção x86 e altere o Output path para C:\soft\PXC\BIN\.

Ao iniciar a aplicação, apresentar o menu abaixo com o total de convênios zerado.

```
Cadastro de convênios para crédito consignado
---------------------------------------------

Total de convênios: 0

1. Adicionar convênio
2. Remover convênio
3. Listar convênios
4. Salvar convênios
5. Sair

Informe a operação desejada:
```

Após a execução de uma operação, exibir o código e a mensagem de retorno com o padrão a seguir, onde 'Codigo' deve ter sempre dois dígitos: "Codigo: Mensagem".

A propriedade do tipo genérica Dados, quando o tipo genérico for inteiro, deverá ter os seguintes valores:
- Quando o retorno for de sucesso (código 00), ela terá a quantidade de registros envolvidos na operação;
- Nos casos de falha (código diferente de 00), ela receberá 0.

A propriedade Dados de Retornos do tipo inteiro não precisa ser exibida no console.

Depois de exibir a mensagem de retorno da operação, esperar que o usuário pressione alguma tecla para então limpar a tela e reapresentar o menu. Ao exibir o menu, sempre apresentar o total de convênios atualizado.

Operações

Abaixo segue a especificação do que deve ser feito na execução de cada operação disponível no menu. Mais abaixo, encontra-se a lista dos códigos de retorno com suas respectivas mensagens.

1. Adicionar convênio
- Solicitar e ler os dados de CNPJ, razão social, quantidade de empregados, status e data de atualização do status;
- Validar se o CNPJ é válido através da expressão regular "^[1-9]\\d{2,13}$" (código 10);
- Validar se o CNPJ já existe na lista (código 11);
- Validar se a razão social possui pelo menos 3 caracteres (código 12);
- Validar que a quantidade de empregados é do tipo inteiro (código 13);
- Validar se a quantidade de empregados é maior que 0 (código 14);
- Validar se o status é um status válido (código 15);
- Validar se a data de atualização do status é uma data válida (código 16);
- Validar se a data de atualização do status não é uma data futura (código 17);
- Adicionar o convênio na lista de convênios do cadastro (código 00).

2. Remover convênio
- Solicitar e ler um CNPJ;
- Validar se o CNPJ é válido através da expressão regular "^[1-9]\\d{2,13}$" (código 20);
- Validar se o CNPJ existe na lista parar poder ser removido (código 21);
- Remover o convênio da lista de convênios do cadastro (código 00).

3. Listar convênios
- Validar se existem registros para serem exibidos (código 30);

- Limpar a tela do console e listar todos os registros da lista de convênios. Exibir todos os campos na seguinte ordem: CNPJ, razão social, quantidade de empregados, status por extenso (Cadastrado, Deferido ou Suspenso) e data de atualização do status com o formato dd/MM/yyyy (código 00).

4. Salvar convênios

- Validar se existem registros para serem salvos (código 40);
- Limpar a tabela antes de inserir os novos dados;
- Inserir os registros da lista de convênios na tabela (código 00);
- Este processo deve ser atômico, ou seja, em caso de falha, a tabela deve voltar ao seu estado original, inclusive desfazendo a limpeza da tabela (código 41).

5. Sair

- Encerrar a aplicação.

Códigos e mensagens de retorno

- 00: “Operação realizada com sucesso”;
- 10: “O CNPJ informado é inválido”;
- 11: “Já existe um convênio com esse CNPJ”;
- 12: “A razão social deve ter pelo menos 3 caracteres”;
- 13: “A quantidade de empregados informada é inválida”;
- 14: “A quantidade de empregados deve ser maior que 0”;
- 15: “O status é inválido”;
- 16: “A data de atualização do status é inválida”;
- 17: “A data de atualização do status não pode ser uma data futura”;
- 20: “O CNPJ informado é inválido”;
- 21: “Não foi encontrado nenhum convênio com o CNPJ informado”;
- 30: “Nenhum registro encontrado”;
- 40: “Nenhum registro encontrado”;
- 41: “Exceção ao interagir com o banco de dados: {mensagem da exceção}”.

Acesso ao Banco de Dados

O banco de dados tem apenas uma tabela chamada CONVENIO:

Nome do campo	Tipo de dados	Tamanho do campo	Chave	Requerido
CNPJ	Texto Curto	14	Sim	Sim
RAZAO_SOCIAL	Texto Curto	20	Não	Sim
QTD_EMPREGADOS	Número	Inteiro	Não	Sim
STATUS	Número	Byte	Não	Sim
DT_ATU_STATUS	Data/Hora	-	Não	Sim

Para acessar o banco de dados utilize as instruções abaixo:

1. Copie a AcessoBancoDados.dll para a pasta c:\soft\pxc\bin; caso a pasta não exista, crie-a;
2. Copie o banco de dados Pxcz02da.mdb para a pasta c:\soft\pxc\data; caso a pasta não exista, crie-a;
3. A classe de acesso a dados pertence ao namespace Bergs.ProvacSharp.BD;
4. Como utilizar a classe AcessoBancoDados:
    a. Adicionar uma referência a esta DLL no seu projeto AvaliacaoCSharp (trocar o Copy Local para false);
    b. Ao criar uma instância, informar no construtor o caminho completo do arquivo de banco de dados Access (conforme item 2 acima);
    c. Após criar a instância, executar o método “Abrir”. Este método cria a conexão com o banco de dados e já inicia uma transação (BeginTrans);
    d. Para cada convênio a ser inserido, enviar o respectivo comando SQL para o método “ExecutarInsert”. Caso a inserção ocorra com sucesso, este método devolverá true. Para os demais casos, false, ou ainda pode ocorrer uma exceção;
    e. Para confirmar todas as modificações no banco de dados, ao término das inserções deve ser chamado o método “EfetivarComandos” (Commit);

f. Para encerrar a conexão com o banco de dados, basta invocar o método “Dispose”;
g. A classe AcessoBancoDados implementa a interface IDisposable, logo, para controlar o fechamento da conexão é possível utilizar a diretiva using;
h. Encerrar a conexão com o banco de dados SEM executar o comando “EfetivarComandos” acarretará em perda de TODAS as modificações realizadas no banco de dados (Rollback);

Regras

1. A avaliação é individual;
2. Não é permitido o auxílio de colegas da turma, somente dos monitores e instrutores. Estes poderão auxiliar em dúvidas relativas à definição da avaliação e de maneira limitada em dúvidas técnicas;
3. Não é permitido a utilização de e-mail, Skype, WhatsApp ou qualquer outro aplicativo de conversa entre integrantes da turma;
4. É permitido a consulta ao material de aula e seus artefatos localizados na máquina local ou na área privada;
5. É permitida a consulta à Internet a partir da estação de trabalho para auxílio na sintaxe e semântica de comandos.

Forma de Entrega

1. Compactar em um arquivo zip as pastas e arquivos que contemplam os componentes gerados (não enviar as pastas .vs, bin e obj). A pasta .vs pode aparecer ou não, dependendo da configuração de visibilidade das pastas no Windows Explorer. A pasta bin pode existir ou não, dependendo do momento da configuração do Output path da aplicação;
2. Enviar o arquivo zipado como anexo em um e-mail para “Dalton Torres, Frederico Selbach”, com o título “Avaliação C#”;
3. Confirmar com algum dos destinatários a recepção do arquivo.

Dicas

- A expressão regular “^[1-9]\\d{2,13}$” significa uma string com tamanho entre 3 e 14 dígitos que não começa com 0;

- Exemplo de teste de expressão regular:

```
if (System.Text.RegularExpressions.Regex.IsMatch(sValor, sExpressao)) { ... }
```

- Os dados “123”, “0123” e “00123” devem ser considerados como sendo o mesmo CNPJ (dica da dica: PadLeft);

- Exemplos de teste de falha de banco de dados:
  Razão social com mais de 20 caracteres;
  Quantidade de empregados maior que 32767.

- Para se evitar adições manuais de convênios durante os testes, é possível adicioná-los programaticamente no início do programa. Se fizer isso, lembre-se de removê-los do código antes de entregar a avaliação;

- Na hora de enviar a avaliação por e-mail, adicione os destinatários por último, evitando assim o envio do e-mail antes de ele estar pronto;

- Reserve pelo menos os últimos 15 minutos para preparo e envio da avaliação.