**CONTA**

Tipos de Conta: Conta Corrente OK, Poupança OK e Salário PEN

**Campos:**

* + Titular: obrigatório e readonly- OK
  + Agencia: obrigatório e readonly. Número não pode ser zero PEN
  + Número da conta: obrigatório e readonly. Número não pode ser zero PEN

**Características:**

* Saldo inicial CC: R$ 100,0 PEN
* Saldo inicial CP: R$ 0,0 PEN
* Saldo inicial Salário: R$ 0,0 PEN
* Total de Contas Criadas: Informar por tipo de conta o total de contas criadas. AND Dificuldade em acessar a classe principal e colocar um contador dentro do construtor.
* Se o cliente tiver uma conta corrente e poupança, não deve ser cobrada a taxa de saque PEN
* Criar uma Cesta de Serviços onde os primeiros 3 saques da CC serão gratuitos. Demais, cobrar R$0,10. PEN
* Criar um limite de cheque especial para clientes selecionados PEN
* Exibir um extrato com as operações executadas no dia PEN

**Funções e Restrições**

* **Método Depositar**
  + Não pode depositar valores negativos- OK
* **Método Sacar**
  + Taxa de R$ 0,10 por saque no caso de Conta Corrente - OK
  + Não pode sacar mais que o valor do saldo- OK
  + Não pode sacar valores zerados ou negativos- OK
  + (CC) Checar Cesta de Serviços e verificar quantidade de Saques realizados PEN
  + **Método Transferir**
    - Não pode transferir mais que o valor do saldo- OK
    - Não pode transferir valores negativos- OK
    - Criar parâmetro para permitir ou não transferência entre Conta Corrente e Poupança - PEN

Solução:

* Criada uma classe abstrata Conta com os métodos abstratos: Sacar, Depositar e Transferir
* Criada classe contaCorrente, contaPoupança contaSalario herdada de Conta
* Saldo inicial informado na classe abstrata Conta
* Os campos titular, agencia e numero da conta foram feitos com val para ser readonly
* O método Sacar foi feito com retorno Booleano, para poder ser usado como validação no método Transferir
* Na classe contaCorrente, foi criado um método **sacarRecursoSemTaxa** para ser usado no método **Transferir** sem a cobrança automática de Taxa
* Método transferir usa como argumentos a conta destino e os métodos Sacar e Depositar

**CLASSE CLIENTE**

**Como alinhar a classe cliente com a classe de criação de contas? Para que a criação de uma conta esteja vinculada à criação de de um cadastro de cliente?**

* Propriedade Nome .   
  Propriedade CPF  
  Propriedade Profissao  
  Contator de clientes criado
* Exigir no Construtor a informação de nome e CPF e somar o total de clientes criados

**CLASSE FUNCIONARIO**

* A classe funcionário será abstrata [...]
* Propriedade Nome  
  Propriedade Salario com SET protegido [...]
* Propriedade CPF com SET privado.   
  Contator de Funcionarios criados, com SET privado  
  Construtor da classe indica obrigatoriedade dos campos Nome e CPF, e somando o funcionário criado  
  Método para aumento de salario abstrato [...]  
  Metodo para receber Bonificação como abstrato [...]

**CLASSE AUXILIAR : FUNCIONARIO**

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionário | Salário Base | Aumento Salarial | Bonificação |
| Auxiliar | R$2.000 | 10% | 20% |

* Construtor da classe obedece a classe principal / base (Funcionario) e obriga o preenchimento de Nome e CPF
* Método AumentaSalario sobrescreve o método base com aumento de 10%. Propriedade Salario pertence à classe base.
* Método GetBonificação sobrescreve o método base com um retorno de 20% sobre o salário. Propriedade Salario pertence à classe base.

**CLASSE DESENVOLVEDOR: FUNCIONARIO**

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionário | Salário Base | Aumento Salarial | Bonificação |
| Desenvolvedor | R$3.000 | 15% | 10% |

* Construtor da classe obedece a classe principal / base (Funcionario) e obriga o preenchimento de Nome e CPF  
  Método AumentaSalario sobrescreve o método base com aumento de 15%. Propriedade Salario pertence à classe base.  
  Método GetBonificação sobrescreve o método base com um retorno de 10% sobre o salário. Propriedade Salario pertence à classe base.

**CLASSE DESIGNER : FUNCIONARIO**

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionário | Salário Base | Aumento Salarial | Bonificação |
| Designer | R$3.000 | 11% | 17% |

Construtor da classe obedece a classe principal / base (Funcionario) e obriga o preenchimento de Nome e CPF  
Método AumentaSalario sobrescreve o método base com aumento de 11%. Propriedade Salario pertence à classe base.  
Método GetBonificação sobrescreve o método base com um retorno de 17% sobre o salário. Propriedade Salario pertence à classe base.

**CLASSE DIRETOR : FUNCIONARIO**

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionário | Salário Base | Aumento Salarial | Bonificação |
| Diretor | R$5.000 | 15% | 50% |

Construtor da classe obedece a classe principal / base (Funcionario) e obriga o preenchimento de Nome e CPF  
Método AumentaSalario sobrescreve o método base com aumento de 15%. Propriedade Salario pertence à classe base.

Método GetBonificação sobrescreve o método base com um retorno de 50% sobre o salário. Propriedade Salario pertence à classe base.

**CLASSE GERENTE DE CONTA : FUNCIONARIO**

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionário | Salário Base | Aumento Salarial | Bonificação |
| Gerente de Conta | R$4.000 | 5% | 25% |

* Construtor da classe obedece a classe principal / base (Funcionario) e obriga o preenchimento de Nome e CPF
* Método AumentaSalario sobrescreve o método base com aumento de 5%. Propriedade Salario pertence à classe base.  
  Método GetBonificação sobrescreve o método base com um retorno de 25% sobre o salário. Propriedade Salario pertence à classe base.
* CLASSE PARCEIRO COMERCIAL : FUNCIONARIO
* CLASSE GERENCIADOR DE BONIFICAÇÃO
* SISTEMA INTERNO –
* CLASSE FUNCIONARIO AUTENTICAVEL
* CLASSE IAUTENTICAVEL
* CLASSE SISINTERNO
* CLASSE PROGRAM

Tecnologias

* Reuse código com Herança e Composição OK
* Use os modificadores virtual, abstract e override OK
* Use o modificador de visibilidade protected OK

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionário | Salário Base | Aumento Salarial | Bonificação |
| Auxiliar | R$2.000 OK | 10% OK | 20% OK |
| Desenvolvedor | R$3.000 OK | 15% OK | 10% OK |
| Designer | R$3.000 OK | 11% OK | 17% OK |
| Diretor | R$5.000 OK | 15% OK | 50% OK |
| Gerente de Conta | R$4.000 OK | 5% OK | 25% OK |

Implementar Interface e métodos abstratos

Desenvolver um Sistema interno, autenticável apenas pelo Gerente de Contas, Diretor e Parceiro Comercial

Há também um tipo que deve autenticar: Parceiro Comercial. Este não é um funcionário   
[Conceito de Herança múltipla Interface]

Desafio Interface:

Utilizamos as interfaces no contexto de funcionários, porém, podemos também estender o exemplos para o contexto de contas. Por exemplo, na Bytebank pode existir um novo tipo de conta chamada de **conta salário**.

Esse tipo de conta costuma ter as mesmas informações e comportamentos que uma conta comum, porém com mais restrições. Por exemplo, não é possível usar o serviço de transferência.

Considerando esse detalhe, aceite o desafio e crie a **conta salário** que tem tudo que uma conta tem, com exceção do comportamento de transferir. Nesta implementação, mantenha a mesma técnica de polimorfismo na transferência para contas que precisam desse comportamento (corrente e poupança).

Para testar o código, tente realizar a transferência com a conta salário e confira se apresenta um problema de compilação.