INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PIAUÍ	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ** **Curso: Análise e Desenvolvimento de Sistemas** **Disciplina: Programação Orientada a Objetos** **Professor: Ely**

Trabalho final

Implemente um pequeno sistema que envolva um cadastro “1 para N” ou “mestre-detalhe”.

Exemplos:

- Um usuário possui vários posts;
- Uma empresa possui várias ações no mercado
- Um cartão de crédito possui vários lançamentos de débito.

1) A classe mestre deve ter algum relacionamento de herança, como vimos nos exemplos de Conta e Poupança. A subclasse da relação deve ter atributos e métodos que justifiquem a herança. Além disso, cada classe deve ter um atributo id que seja um identificador “primário” de cada objeto sequencial começando de 1.
2) Uma outra classe deve ser a responsável pelos cadastros e ter métodos para consultar, alterar, inserir, alterar e excluir as classes mestre e detalhe como na classe Banco desenvolvida durante o curso.
3) A classe acima não deve ter conhecimento da forma de implementação da “persistência”, pois em vez de um atributo que seja um array de objetos, deve ter uma interface como repositório.

   Ex:

```
interface IRepositorioDeContas {
    inserir(conta: Conta): void;
    consultar(numero: string): Conta;
    alterar(conta: Conta): void;
    //...
}

class Banco {
    private _repositorio: IRepositorioDeContas;

    constructor(repositorio: IRepositorioDeContas) {
        this._repositorio = repositorio
    }

    inserir(conta: Conta): void {
        try {
            this.consultar(conta.numero);
            throw new ContaJaCadastradaError(`...`);
```

```
            } catch (e: any) {
                if (e instanceof ContaJaCadastradaError) {
                    throw e;
                }
                this._repositorio.inserir(conta);
            }
        }
        //...
    }
```

4) Uma classe que implemente a interface de repositório deve, por sua vez, conhecer o mecanismo de armazenamento/persistência. No exemplo abaixo, temos uma classe que usa o mecanismo de “persistência” em memória com array:

```
class RepositorioArray implements IRepositorioDeContas {
    private _contas: Conta[] = [];

    inserir(conta: Conta): void {
        this._contas.push(conta);
    }
    //...
}
```

5) Proponha uma segunda implementação de persistência como arquivos, lista encadeada, WEB API ou mesmo banco de dados local.

6) A aplicação deve possuir duas instâncias de cadastros, cada um com um distinto mecanismo de persistência e devem ser propostas interações entre os dois cadastros. Exemplos:
    - Transferir uma conta do banco que usa array para o que usa lista encadeada;
    - Transferir valores de uma conta que use o mecanismo de banco de dados que usa arquivos como mecanismo de persistência.

7) Os cadastros devem ter regras de validação como:
    - Não aceitar objetos duplicados;
    - Não aceitar determinados atributos vazios;
    - Validar formato de valores monetários ou números inválidos etc.

8) As regras de validação que forem violadas devem lançar exceções específicas.

9) Deve ser criada uma aplicação que tenha opções de menu com a qual o usuário possa interagir realizando operações e obtendo feedback. Essa aplicação deve tratar exceções lançadas pelas “camadas” anteriores.

Pontuação:

- (2,0) Modelo escolhido com herança e relacionamento mestre-detalhe;
- (2,0) Validações, lançamento e tratamento de exceções;
- (4,0) Implementação do cadastro com interfaces e mecanismos de persistências;
- (2,0) Aplicação com menus e feedbacks de forma funcional.