	INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ **Curso: ADS** **Disciplina: Programação Orientada a** **Objetos Professor: Ely**

Exercício 08 – parte 2

1) Crie as classes AplicacaoError descendente de Error. Crie também classes ContaInexistenteError, ClienteNaoEncontradoError e SaldoInsuficienteError. Todas decendentes da classe AplicacaoError.

2) Implemente na classe Banco os métodos consultar e consultarPorIndice para Contas e Clientes, caso não sejam encontrados, a exceção ContaInexistente seja lançada.

3) Altere os métodos alterar, depositar, sacar, transferir, renderJuros removendo os "ifs/elses", pois caso haja exceção no método consultar, os respectivos códigos não serão mais necessários. Ex:

Antes	Depois

x(numero: string): void {
let procurada =
consultar(numero); if (procurada
!= null) {
conta.metodoY(...);
}
}
	x(numero: string): void {
	let procurada =
	consultar(numero);
	çconta.metodoY(...);
	}

4) Crie uma exceção chamada ValorInvalidoError que herda de AplicacaoException e altere a classe Conta para que ao receber um crédito/depósito, caso o valor seja menor ou igual a zero, seja lançada a exceção ValorInvalidoError. Altere também o construtor da classe Conta para que o saldo inicial seja atribuído utilizando o método depositar.

5) Você percebeu que o código que valida se o valor é menor ou igual a zero se repete nos métodos sacar e depositar? Refatore o código criando um método

privado chamado validarValor onde um valor é passado como parâmetro e caso o mesmo seja menor ou igual a zero, seja lançada uma exceção. Altere também os métodos sacar e depositar para chamar esse método de validação em vez de cada um lançar a sua própria exceção, evitando assim a duplicação de código.

6) Crie uma exceção chamada PoupancaInvalidaError que herda de AplicacaoError. Altere então o método render juros da classe Banco para que caso a conta não seja uma poupança, a exceção criada seja lançada.

7) Proponha validações para o cliente: CPF inválido, data de nascimento inválida. Lance também uma validação para que não seja permitida que uma conta pertença a mais de um cliente.

8) Crie exceções relacionadas a valores obtidos da entrada de dados que não sejam aceitáveis, como valores vazios, números inválidos também para clientes e etc. Além disso, mova o lançamento de exceções de ValorInvalido na classe conta para a classe App. Dessa forma, você tratará os erros já na entrada de dados e deixará as classes básicas menos “poluídas”.

Nota: nenhuma das exceções lançadas por você ou pela aplicação deve “abortar” o programa. Elas devem ser obrigatoriamente tratadas.