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

Exercício 07 – parte 2

1) Crie as classes AplicacaoError, ContaInexistenteError e SaldoInsuficienteError de acordo com a hierarquia apresentada em sala.
2) Implemente na classe Banco os métodos consultar e consultarPorIndice para que, caso a conta procurada não seja encontrada, a exceção ContaInexistente seja lançada.
3) Altere os métodos alterar, creditar, 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 ValorIvalidoException. Altere também o construtor da classe Conta para que o saldo inicial seja atribuído utilizando o método creditar.
5) Você percebeu que o código que valida se o valor é menor ou igual a zero se repete nos métodos sacar e creditar? 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 creditar 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 Error. Altere então o método render juros da classe Banco para que caso a conta não seja uma poupança, a exceção criada seja lançada.
7) Crie uma validação para não cadastrar mais de uma conta com o mesmo número. Para isso, chame o método consultar no método inserir da classe banco. Apenas o tratamento da exceção do método consultar, você deve incluir a conta.
8) Pesquise em TypeScript uma forma de entrada de dados e crie uma aplicação com opções de menu para todos os métodos da classe Banco passíveis de uso. Utilize um do {} while com opções lidas pelo teclado conforme mostra a estrutura do slide "Aplicação Robusta.
9) Por fim, crie exceções relacionadas a valores obtidos da entrada de dados que não sejam aceitáveis, como valores vazios, números inválidos etc. Na aplicação, trate todas as entradas de dados para que, caso o usuário infrinja regras de preenchimento, o sistema lance e trate as exceções e informe que a entrada foi inválida.

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