**Escola Superior Dom Helder Câmara**
Escola de Engenharia
Disciplina: Algoritmo e Estruturas de Dados II
Professor: Diego Silva Caldeira Rocha

Lista 2 - Programação Defensiva – Tratamento de Exceção

1. Implemente o código a seguir e resolva as questões especificadas com utilização de tratamento de exceção.

```java
package exception.creation;
public class Conta {
    private double saldo;
    private String nome, cpf;
    public Conta(String nome, String cpf, double saldo){
        this.setNome(nome);
        this.setCpf(cpf);
        this.setSaldo(saldo);
    }
    public String getNome() {return nome;}
    public void setNome(String nome) {this.nome = nome; }
    public String getCpf() {return cpf;}
    public void setCpf(String cpf) {this.cpf = cpf; }
    public double getSaldo() {return saldo;}

    public void setSaldo(double saldo) {
        this.saldo = (saldo < 0) ? 0:saldo;
    }
    public void deposita (double valor){
        setSaldo((valor < 0)? saldo:saldo+valor);
    }
    public void saca (double valor) throws SaldoInsuficienteException{
        //saldo é suficiente para saque
        if(getSaldo() > valor){
            setSaldo(getSaldo()-valor);
        }
        else{
            throw new SaldoInsuficienteException("Seu saldo é: " + this.saldo
                    + "\n Você tentou sacar: " + valor);
        }
    }
    public String toString(){
        return "Cliente: " + this.nome +
        "\nCPF: " + this.cpf +  "\nSaldo: " + this.saldo;
    }
}//fim da classe Conta
```

```java
package exception.creation;

public class SaldoInsuficienteException extends Exception {

    public SaldoInsuficienteException(String msg){
        super(msg);
    }

}
```

```java
package exception.creation;
import javax.swing.JOptionPane;
public class Banco {
    public static void main(String args[]){
        //Obtem dados
        String nome =  typeString("Digite o nome do cliente");
        String cpf = typeString("Digite o cpf do cliente");
        double saldo = typeDouble("Digite o saldo inicial do cliente");

        Conta c = new Conta(nome, cpf, saldo);
        double deposito = typeDouble("Digite o valor do depósito");
        c.deposita(deposito);

        double saque = typeDouble("Digite o valor do saque");

        try{
            c.saca(saque);
        }
        catch(SaldoInsuficienteException ex){
            JOptionPane.showMessageDialog(null, ex.getMessage()+
                    "\nOperação não realizada!");
        }
        JOptionPane.showMessageDialog(null, c.toString());
    }

    private static double typeDouble(String txt){
        String userInput = JOptionPane.showInputDialog(txt);
        double d = Double.parseDouble(userInput);
        return d;
    }

    private static String typeString(String txt){
        return JOptionPane.showInputDialog(null, txt);
    }
}
```

a) Adapte o código para resolver o problema de entrada inválida. Quando for solicitada a entrada de um valor (*double*) e o usuário digitar valor inválido (a,u,&, etc.), a aplicação deve exibir uma mensagem informando que foi digitado um valor errado e permitir que o usuário digite o valor novamente. Dica: Exceção a ser tratada: *NumberFormatException(classe padrão java existente ).*

b) Adapte o código para tratar a exceção de tentativa de saque no limite concedido pelo banco. Por exemplo, se o cliente tem saldo igual R$ 400,00 e o limite do banco é igual a R$ 200,00, quando for tentar sacar R$ 500,00 ele deverá ser informado que entrará no limite em R$ 100,00 (o saque deverá ser realizado mesmo com a utilização do limite). Caso o cliente tente sacar acima do saldo mais o limite (R$700,00, por exemplo), o tratamento dado deve ser o mesmo aplicado no código acima. Obs1: Crie classe, caso ache conveniente. Obs2: suponha que o limite do banco é sempre R$ 200,00. Obs3: Não é preciso reescrever todo o código, somente a parte modificada.

2- Modifique o código abaixo para que ele lance uma exceção quando acontecer uma divisão por zero.

```
1 public class Main {
2     public static double divisao(int a, int b) {
3         return (double) a / b;
4     }
5
6     public static void main(String[] args) {
7         double d = divisao(4, 0);
8         System.out.println(d);
9     }
10 }
```

3- Escreve o código para lançar um objeto de exceção *EntradaInvalida* se o número recebido pela função fatorial for maior do que 20. Você deve definir a classe e implementar o código necessário.

```
1 public class Main {
2     public static int fatorial(int numero) {
3         int fat = 1;
4         int n = 1;
5         while (++n <= numero) {
6             fat *= n;
7         }
8         return fat;
9     }
10
11    public static void main(String[] args) {
12        System.out.println(fatorial(10));
13    }
14 }
```

4- Escreva um programa para verificar se um triângulo é Equilátero, Isósceles ou Escaleno. Este método pertence à classe Triangulo e possui a seguinte assinatura: int determinarTipo(). O retorno do método é um inteiro que representa os seguintes casos: Equilátero (0), Isósceles (1), Escaleno (2).

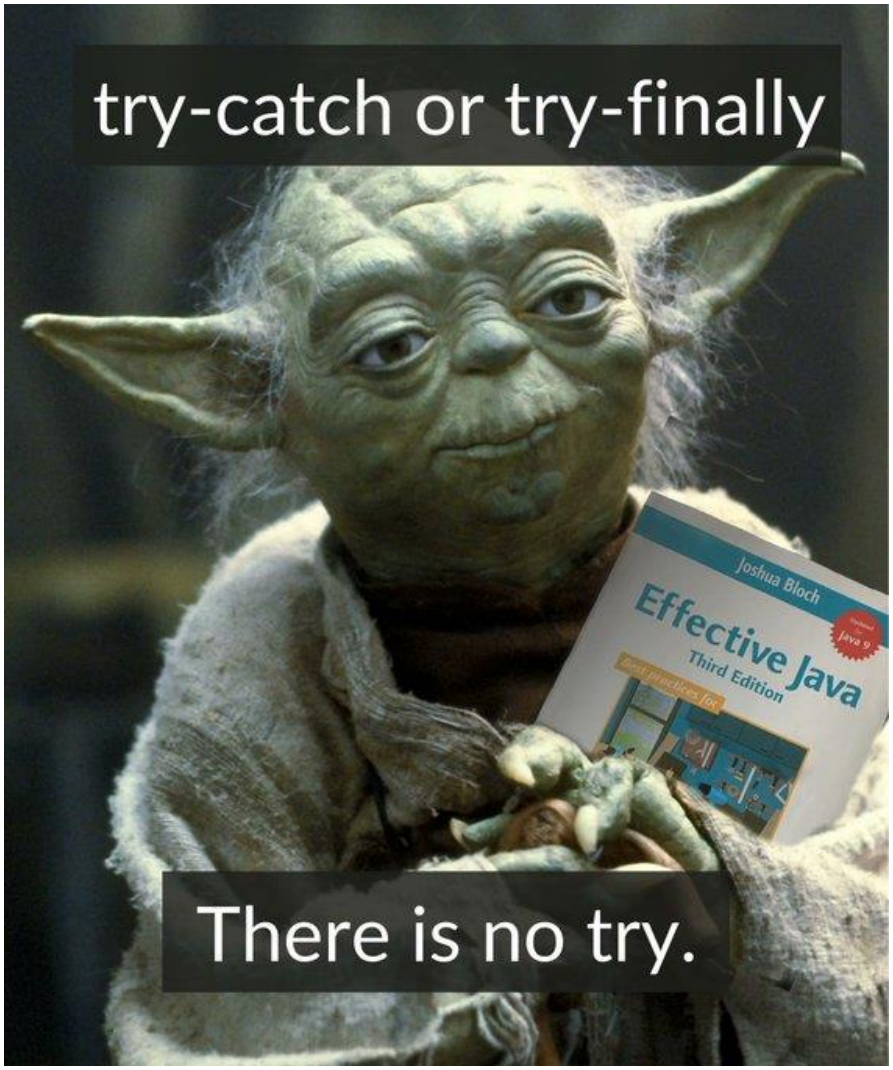
A classe triangulo deve possuir um construtor que recebe três parâmetros do tipo double, representando as dimensões dos lados do triângulo. Este construtor deve tratar casos excepcionais, como lados negativos, lados com valor zero e a desigualdade triangular não atendida. Em todos esses possíveis casos excepcionais, deve ser lançada uma exceção chamada TrianguloInvalidoError().

Escreva um programa principal com pelo menos 4 casos de testes (Equilátero (0), Isósceles (1), Escaleno (2) e TrianguloInvalidoError()).

**Informações sobre cópias**

As questões são individuais. Em caso de cópias de trabalho a pontuação será zero para os autores originais e copiadores. Não serão aceitas justificativas como: “Fizemos o trabalho juntos, por isso estão idênticos”.

**Para descontrair**


try-catch or try-finally
Joshua Bloch
Effective Java
Third Edition
There is no try.