

Lista 3 - Programação Defensiva – Tratamento de Exceção

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

```
1 package exception.creation;
2 public class Conta {
3     private double saldo;
4     private String nome, cpf;
5     public Conta(String nome, String cpf, double saldo){
6         this.setNome(nome);
7         this.setCpf(cpf);
8         this.setSaldo(saldo);
9     }
10    public String getNome() {return nome;}
11    public void setNome(String nome) {this.nome = nome; }
12    public String getCpf() {return cpf;}
13    public void setCpf(String cpf) {this.cpf = cpf; }
14    public double getSaldo() {return saldo;}
15
16    public void setSaldo(double saldo) {
17        this.saldo = (saldo < 0) ? 0:saldo;
18    }
19    public void deposita (double valor){
20        setSaldo((valor < 0)? saldo:saldo+valor);
21    }
22    public void saca (double valor) throws SaldoInsuficienteException{
23        //saldo é suficiente para saque
24        if(getSaldo() > valor){
25            setSaldo(getSaldo()-valor);
26        }
27        else{
28            throw new SaldoInsuficienteException("Seu saldo é: " + this.saldo
29                + "\n Você tentou sacar: " + valor);
30        }
31    }
32    public String toString(){
33        return "Cliente: " + this.nome +
34            "\nCPF: " + this.cpf + "\nSaldo: " + this.saldo;
35    }
36 }//fim da classe Conta
37
```



```
1 package exception.creation;
2
3 public class SaldoInsuficienteException extends Exception {
4
5     public SaldoInsuficienteException(String msg){
6         super(msg);
7     }
8
9 }
```

```

1 package exception.creation;
2 import javax.swing.JOptionPane;
3 public class Banco {
4     public static void main(String args[]){
5         //Obtem dados
6         String nome = typeString("Digite o nome do cliente");
7         String cpf = typeString("Digite o cpf do cliente");
8         double saldo = typeDouble("Digite o saldo inicial do cliente");
9
10        Conta c = new Conta(nome, cpf, saldo);
11        double deposito = typeDouble("Digite o valor do depósito");
12        c.deposita(deposito);
13
14        double saque = typeDouble("Digite o valor do saque");
15
16        try{
17            c.saca(saque);
18        }
19        catch(SaldoInsuficienteException ex){
20            JOptionPane.showMessageDialog(null, ex.getMessage()+
21                "\nOperação não realizada!");
22        }
23        JOptionPane.showMessageDialog(null, c.toString());
24    }
25
26    private static double typeDouble(String txt){
27        String userInput = JOptionPane.showInputDialog(txt);
28        double d = Double.parseDouble(userInput);
29        return d;
30    }
31
32    private static String typeString(String txt){
33        return JOptionPane.showInputDialog(null, txt);
34    }
35 }

```

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- Escreva 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 Triângulo 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 triângulo 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()`.

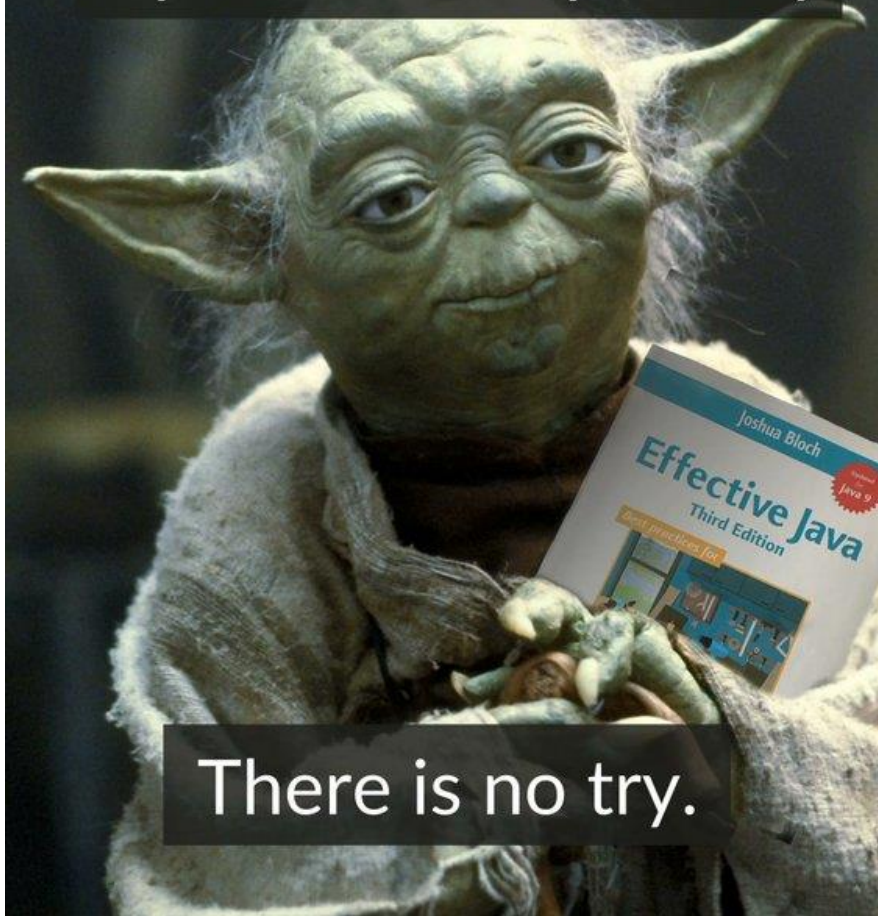
Escreva um programa principal com pelo menos 4 casos de testes (Equilátero (0), Isósceles (1), Escaleno (2) e `TrianguloInvalidoError()`).

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



There is no try.