Powered by GitBook

Estruturas excepcionais

Exceções

Ao executar o código Java, diferentes erros podem acontecer: erros de codificação feitos pelo programador, erros devido a entrada errada ou outros imprevistos.

Quando ocorre um erro, o Java normalmente para e gera uma mensagem de erro. O termo técnico para isso é: Java lançará uma **exceção** (jogará um erro).

De forma interpretativa em Java, um erro é algo irreparável, a aplicação trava ou é encerrada drásticamente. Já exceções é um fluxo inesperado da nossa aplicação, exemplo: Querer dividir um valor por zero, querer abrir um arquivo que não existe, abrir uma conexão de banco, com usuário ou senha inválida. Todos estes cenários e os demais, não são erros mas sim fluxos, não previstos pela aplicação.

É ai que entra mais uma responsabilidade do desenvolvedor, prever situações iguais a estas e realizar o que denominamos de ***Tratamento de Exceções***.

Mão na massa!

Vamos trazer o cenário que estudamos na aula, sobre Terminal e Argumentos onde via terminal informamos alguns dados pessoais.

```java
import java.util.Locale;
import java.util.Scanner;

public class AboutMe {
    public static void main(String[] args) {
        //criando o objeto scanner
        Scanner scanner = new Scanner(System.in).useLocale(Locale.US);

        System.out.println("Digite seu nome");
        String nome = scanner.next();

        System.out.println("Digite seu sobrenome");
        String sobrenome = scanner.next();

        System.out.println("Digite sua idade");
        int idade = scanner.nextInt();

        System.out.println("Digite sua altura");
        double altura = scanner.nextDouble();


        //imprimindo os dados obtidos pelo usuario
        System.out.println("Olá, me chamo " + nome.toUpperCase() + " " + sobre
        System.out.println("Tenho " + idade + " anos ");
        System.out.println("Minha altura é " + altura + "cm ");
        scanner.close();
    }
```

Aparentemente é um programa simples, mas vamos listar algumas possíveis exceções, que podem acontecer.

- Não informar o nome e sobrenome;
- O valor da idade ter um caractere NÃO numérico;
-

O valor da altura ter uma vírgula ao invés de ser um ponto ***(conforme padrão americano);***

Executando o nosso programa com os valores abaixo, vamos entender qual exceção acontecerá:

Entrada	Valor	Exceção	Causa
Digite seu nome:	**Marcelo**		
Digite seu sobrenome:	**Azevedo**		
Digite sua idade:	**quinze (15)**	java.util.InputMismatchException	O programa esperava o valor do tipo numérico inteiro.
Digite sua altura:	**1,65**	java.util.InputMismatchException	O programa esperava o valor do tipo numérico decimal no formato americano, exemplo: **1.65**

A melhor forma de prever, que pode ocorrer uma exceção, é pensar que ela pode ocorrer. ***Lei de Murphy***

Conhecendo algumas exceções já mapeadas

A linguagem Java, dispõe de uma vasta lista de classes que representam exceções, abaixo iremos apresentar as mais comuns:

Nome	Causa
java.lang.NullPointerException	Quando tentamos obter alguma informação de uma variável nula.
java.lang.ArithmeticException	Quando tentamos dividir um valor por zero.
java.sql.SQLException	Quando existe algum erro, relacionado a interação com banco de dados.
java.io.FileNotFoundException	Quando tentamos ler ou escrever, em um arquivo que não existe.

Tratamento de exceções

E quando inevitavelmente, ocorrer uma exceção? Como nós desenvolvedores podemos ajustar o nosso algoritmo para amenizar o ocorrido?

A instrução `try`, permite que você defina um bloco de código, para ser testado quanto a erros enquanto está sendo executado.

A instrução `catch`, permite definir um bloco de código a ser executado, caso ocorra um erro no bloco try.

A instrução `finally`, permite definir um bloco de código a ser executado, independente de ocorrer um erro ou não. As palavras-chave `try` e `catch` vem em pares:

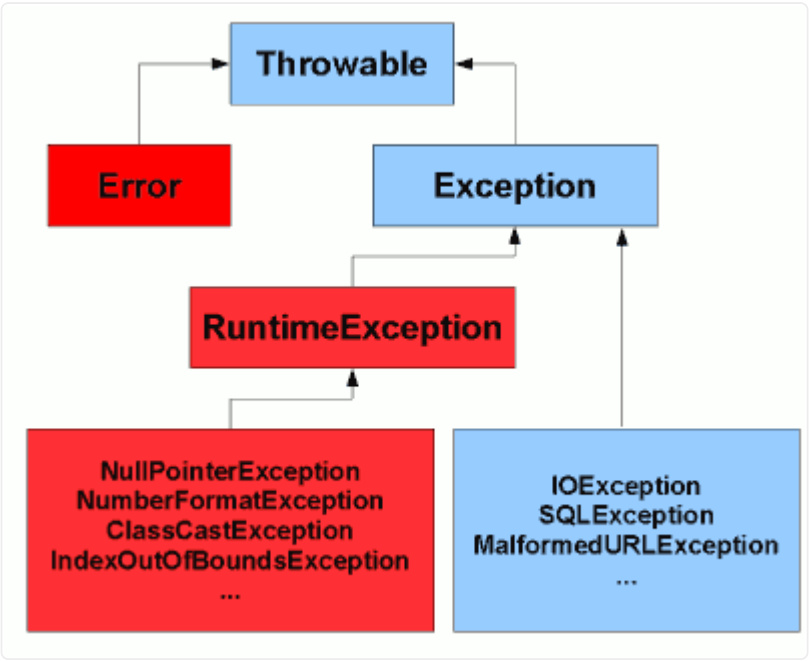
Estrutura de um bloco com try e catch:

```
try {
  //  bloco de código conforme esperado
}
catch(Exception e) {// precisamos saber qual exceção
  // bloco de código que captura as exceções que podem acontecer
  // em caso de um fluxo não previsto
}
```

> O bloco `try` / `catch` pode conter um conjunto de **catchs,** correspondentes a cada exceção **prevista** em uma funcionalidade do programa.

Hierarquia das exceções

A linguagem Java, dispõe de uma variedade de classes, que representam exceções e estas classes, são organizadas em uma hierarquia denominadas **Checked and Unchecked Exceptions** ou *Exceções Checadas e Não Checadas*.



O que determina uma exceção ser classificada como **checada** ou **não checada** ?
É o risco dela ser disparada, logo, você precisa tratá-la, exemplo:

Vamos imaginar que precisamos realizar de duas maneiras, a conversão de uma String para um número, porém o texto contém Alfanuméricos.

```
public class ExemploExcecao {
    public static void main(String[] args) {
        Number valor = Double.valueOf("a1.75");

        valor = NumberFormat.getInstance().parse("a1.75");

        System.out.println(valor);

    }
}
```

As linhas 3 e 5, apresentarão uma exceção ao serem executadas, e a linha 5 contém um método que pode disparar uma exceção checada, logo, nós programadores que iremos usar este método, teremos que tratá-la explicitamente com `try \ catch`.

Exceções customizadas

Nós podemos criar nossas próprias exceções, baseadas em regras de negócio e assim melhor direcionar quem for utilizar os recursos desenvolvidos no projeto, exemplo:

- Imagina que como regra de negócio, para formatar um cep, necessita sempre de ter 8 dígitos, caso contrário, lançará uma exceção que denominamos de **CepInvalidoException**.
- Primeiro criamos nossa exceção:

```
public class CepInvalidoException extends Exception {}
```

- Em seguida, criamos nosso método de formatação de cep:

```
static String formatarCep(String cep) throws CepInvalidoException{
        if(cep.length() != 8)
          throw new CepInvalidoException();

          //simulando um cep formatado
          return "23.765-064";
    }
```

Referências

Java Exceptions (Try...Catch)

11 Erros que desenvolvedores Java cometem quando usam Exceptions | Oracle Brasil

Previous
Estruturas de repetição

Next
Cases

Last updated 1 year ago