

Exceções e Erros

Maurício Linhares

Exceções e erros

- ▶ Em Java, existe uma forma padronizada de um programa avisar que estão acontecendo erros, essa funcionalidade é representada através dos objetos “Exception” ou Exceções;
- ▶ Em Java, objetos que herdam da classe **Throwable** podem ser lançados como erro;
- ▶ As duas subclasses de Throwable disponíveis para utilização na linguagem são Error e Exception;



Por que ter objetos que representam erros?

- ▶ Para poder adicionar informação de diagnóstico do erro, como quais objetos estavam trabalhando quando ele aconteceu, a linha de código que ele aconteceu e até mesmo uma mensagem contendo informações sobre o problema;
- ▶ Para documentar um problema que pode acontecer de forma padronizada, assim qualquer desenvolvedor vai saber quais os problemas que ele pode encontrar ao lidar com aquele código em específico;



Erros irre recuperáveis - **Error**

- ▶ Objetos que descendem da classe **Error** representam problemas irre recuperáveis encontrados pela máquina virtual durante a sua execução;
- ▶ Normalmente, quando um problema do tipo Error é encontrado, o melhor a se fazer é simplesmente “derrubar” a aplicação, é muito difícil se recuperar de um problema desses;
- ▶ Uma aplicação normal não deve lançar erros, a não ser em casos muito específicos;



Alguns erros comuns da JVM

- ▶ **OutOfMemoryError**
 - ▶ Quando não há mais memória disponível
- ▶ **StackOverflowError**
 - ▶ Recursão infinita
- ▶ **NoClassDefFoundError**
 - ▶ Quando uma classe não está disponível para a aplicação
- ▶ **UnsatisfiedLinkError**
 - ▶ Quando uma biblioteca nativa não pode ser encontrada



Erros recuperáveis – Exceptions

- ▶ São os erros recuperáveis do Java, representam problemas que podem acontecer durante a execução da aplicação, mas que ela pode prever e se preparar para solucionar o problema caso seja necessário;
- ▶ Um código deve lançar uma exceção para sinalizar que algum problema aconteceu e que ele não pode continuar a ser executado normalmente;
- ▶ As Exceptions existem em dois sabores, exceções “controladas” (checked) e exceções “livres” (unchecked);





Exceções controladas (checked exceptions)

- ▶ São os erros que o próprio compilador do Java obriga o código a tratar, o código não vai ser compilado até que o usuário declare o erro ou faça o tratamento dele;
- ▶ Normalmente são erros comuns de se encontrar em uma aplicação e que realmente deveriam ser tratados de qualquer forma;
- ▶ Todas as classes que herdam diretamente de Exception ou que não sejam subclasses de RuntimeException são exceções controladas;
- ▶ Normalmente representam problemas externos a aplicação;



Exemplos de exceções controladas

- ▶ **FileNotFoundException**

- ▶ Lançada quando o código tenta abrir um arquivo inexistente

- ▶ **SQLException**

- ▶ Lançada quando ocorrem erros durante a comunicação com um banco de dados

- ▶ **SocketException**

- ▶ Lançada quando ocorrem erros durante a comunicação entre máquinas em uma rede



Exceções livres - Unchecked exceptions

- ▶ São as exceções que não precisam ser tratadas pelo código onde elas podem ser lançadas;
- ▶ Elas normalmente representam problemas de programação (falhas do desenvolvedor) e o seu “lançamento” demonstra instabilidade em uma aplicação;
- ▶ Todas as exceções que são sub-classes de `RuntimeException` são exceções livres;



Exemplos de exceções livres

▶ NullPointerException

- ▶ Quando se tenta acessar uma propriedade ou invocar um método em uma referência para null;

▶ ArrayIndexOutOfBoundsException

- ▶ Quando se tenta acessar um índice que não existe em um array

▶ ClassCastException

- ▶ Quando é tentada uma operação de cast para tipos não compatíveis

▶ ArithmeticException

- ▶ Quando alguma operação matemática incorreta é tentada, como dividir um inteiro por 0;



Declarando e lançando uma exceção

- ▶ Uma exceção é uma classe comum, que herda de `Exception` ou de uma das suas subclasses;
- ▶ Ela pode ter atributos e métodos como qualquer outra classe em Java



Exemplo de exceção

```
public class SaldoInsuficienteException extends Exception {  
  
    private Conta conta;  
  
    public SaldoInsuficienteException( Conta conta ) {  
        super( "A conta do cliente " + conta.getClient() + " não tem  
saldo o suficiente" );  
        this.conta = conta;  
    }  
  
    public Conta getConta() {  
        return conta;  
    }  
  
}
```



Lançando uma exceção – throw e throws

- ▶ Para lançar uma exceção, você deve criar um objeto do tipo dela e utilizar a palavra reservada “throw” seguida da referência para a exceção que foi criada;
- ▶ Quando um “throw” é encontrado, a execução do programa retorna para o método que chamou o método atual, se houver um bloco de tratamento de erros, esse bloco é executado, se não houver um bloco de tratamento de erros,

