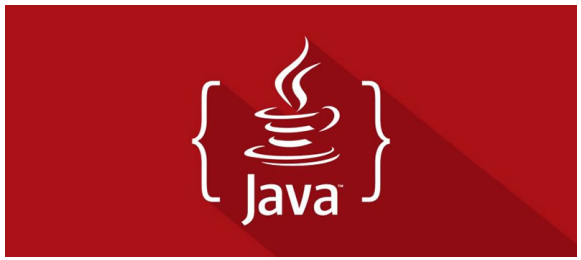
Programação II

Aula 04

Evandro J.R. Silva[1]

[1] Bacharelado em Ciência da Computação
Estácio Teresina



Sumário

1 Introdução

2 Exceções

3 Exercícios

4 FIM

Introdução

- Aula baseada em Java Tutotials by Oracle:
  - Exception.

Exceções

- Durante a compilação e execução, quando algum erro ocorre, um objeto *especial* é criado.
- Esse objeto possui informações sobre o erro ocorrido, incluindo o seu tipo e o estado do programa quando o erro aconteceu.
- Esse objeto é *lançado* (*throw*) pelo compilador em busca de algum trecho do programa que possa lidar com ele.
- Erros que são notificados durante a compilação são chamados de ***checked exceptions***, enquanto os que são notificados durante a execução são chamados de ***unchecked exception***.
- Quando ocorre um erro a compilação ou execução é interrompida, e o erro é mostrado no console.
- E se?
  - Eu não quiser que a compilação/execução seja interrompida?

Exceções

- Durante a compilação e execução, quando algum erro ocorre, um objeto *especial* é criado.
- Esse objeto possui informações sobre o erro ocorrido, incluindo o seu tipo e o estado do programa quando o erro aconteceu.
- Esse objeto é *lançado* (*throw*) pelo compilador em busca de algum trecho do programa que possa lidar com ele.
- Erros que são notificados durante a compilação são chamados de ***checked exceptions***, enquanto os que são notificados durante a execução são chamados de ***unchecked exception***.
- Quando ocorre um erro a compilação ou execução é interrompida, e o erro é mostrado no console.
- E se?
  - Eu não quiser que a compilação/execução seja interrompida?
- Devemos nos servir das *exceções*.

Exceções

- Uma exceção é uma forma de lidar com esses erros sem que o programa seja interrompido.
- Famoso bloco *try - catch*:

```
try
{
    ...
} catch(Exception e)
{
    ...
}
```

- Todo erro que ocorre dentro de um bloco `try` vai ser tratado pelo bloco `catch` correspondente (ou seja, é possível ter vários blocos `catch`).
- Se dois ou mais erros puderem ser tratados com as mesmas linhas de código, a partir do Java SE 7 é possível especificá-los dentro do mesmo bloco:

```
...
catch (IOException|SQLException ex)
{
    logger.log(ex);
    throw ex;
}
```

Exceções

- E se o erro que ocorrer não tiver sido antecipado? Ou seja, é o caso do erro não ser capturado por qualquer bloco `catch`.
- Podemos acrescentar o bloco `finally`, o que **sempre** vai executar assim que a execução do `try` terminar:

```
try{
...
} catch(ExceptionType e){
    ...
} finally{
    ...
}
```

Exceções

- Dependendo da situação, às vezes você quer que determinado método de alguma classe possa *lançar* uma exceção, caso venha ocorrer algum erro (por causa do usuário, por exemplo).
- Você pode declarar esse método como *throwable* (`throws`), ou apenas lançar uma exceção caso alguma condição seja satisfeita.
- Exemplo:

```
public class Exemplo{
    public void depositar(double valor) throws RemoteException{
        // implementacao ...
        throw new RemoteException();
    }
    ...
    ...
}
```

Exceções

- Exemplo 2:

```
public Object pop(){
    Object obj;

    if (size == 0){
        throw new EmptyStackException();
    }

    obj = ...
    ...
}
```

Exceções

- E se eu quiser fazer minha própria exceção?
- O Java te permite isso:

```
class MyException extends Exception{
    ...
}
```

Exceções

- E se eu quiser fazer minha própria exceção?
- O Java te permite isso:

```
class MyException extends Exception{
    ...
}
```

- Outro exemplo:

```
public class ChequeSemFundoException extends Exception{
    private double valor;

    public ChequeSemFundoException(double valor){
        this.valor = valor;
    }

    public double getValor(){
        return valor;
    }
}
```

- E depois dá pra utilizar em outro método:

```
// implementacoes de alguma classe ...
public void sacar(double valor) throws ChequeSemFundoExcepetion{
    if (saldo - valor < 0)
        throw ChequeSemFundoException(valor);
}
```

Exceções

- Como eu vou decorar todas as exceções nativas do Java?

Exceções

- Como eu vou decorar todas as exceções nativas do Java?
- Não vai!
- Na verdade, apenas com o tempo, com bastante prática, você vai aprendendo as exceções mais comuns, e também saber buscar quais exceções podem ser lançadas em determinadas situações.

Exceções

- Como eu vou decorar todas as exceções nativas do Java?
- Não vai!
- Na verdade, apenas com o tempo, com bastante prática, você vai aprendendo as exceções mais comuns, e também saber buscar quais exceções podem ser lançadas em determinadas situações.
- Agora é hora de PRATICAR!

Exercícios

1. Faça um programa que crie um array de 10 posições. O programa vai receber um índice do usuário e mostrar na tela o valor que está no array, no índice indicado, ou tratar uma exceção, caso o índice extrapole os limites do array.
2. Faça um programa que receba dois valores de um usuário, *a* e *b*, e retorne a divisão desses valores: a/b e b/a. Caso qualquer dos valores seja 0, o programa deve lançar tratar a exceção.
3. Faça um programa criando objetos de algumas classes. Tente criar um objeto de uma que classe não existe, e trate a exceção.
4. Crie uma classe Pessoa com os seguintes atributos: nome, idade, peso. Escolha um nome e faça com que toda vez que um objeto da classe Pessoa for criado com um nome diferente, uma exceção seja lançada. Faça o mesmo com os outros dois atributos.

FIM

- Você pode encontrar mais sobre o conteúdo dessa aula no livro **Java: Como Programar** do Deitel, oitava edição, capítulos 11. Disponível nas bibliotecas virtuais (creio que na bilbioteca física também).

Até a próxima!