Como criar sua própria exception

February 22, 2021 by Mauda (http://www.mauda.com.br/?author=1)

Conteúdo do Post:

1. Exceptions, por que devo criá-las?
2. Como criar uma exception “Checked”?
3. Como criar uma exception unchecked?
4. finally {

Olá Pessoal, tudo bem?

No artigo de hoje, vamos descrever um pouco sobre como criar sua própria exception. Dessa forma esse artigo é uma continuação do artigo sobre exceptions (http://mauda.com.br/?p=2313). Veja na sequência:

Exceptions, por que devo criá-las?

Quando estamos trabalhando em um software, criar suas próprias exceptions pode ser muito vantajoso. Primeiro, porque não há uma confusão sobre o que são exceções da linguagem Java, ou de alguma biblioteca terceira, com exceções do sistema.

Outro ponto interessante, é que poderíamos adicionar construtores que armazenam determinada informação que será útil para gerar logs ou mensagens de erro ao usuário.

Por fim, podemos gerar exceptions que tornem mais claros determinados tipos de erro, como por exemplo, objetos não preenchidos corretamente, objetos nulos, validações incorretas.

Como criar uma exception “Checked”?

Para criar uma exception checked, ou seja, uma exceção que vc precisa obrigatoriamente tratar com um **try catch** ou ainda utilizar a palavra chave **throws** na assinatura do método que gerou a exception. Uma exception Checked pode ser gerada a partir de qualquer *classe* Exception, exceto a RuntimeException e suas filhas. Como no exemplo abaixo:

```
1  package br.com.mauda.seminario.cientificos.exception;
2
3  import java.io.FileNotFoundException;
4
5  public class ArquivoCsvNaoEncontradoException extends FileNotFoundException {
6
7      private static final long serialVersionUID = -2346384470483785588L;
8
9      public ArquivoCsvNaoEncontradoException() {
10         super("Arquivo CSV não encontrado! Favor verificar o diretório user/files");
11     }
12
13 }
```

A *classe* ArquivoCsvNaoEncontradoException estende a *classe* java.io.FileNotFoundException, que não é filha de RuntimeException. Vamos deixar uma mensagem padrão quando lançamos essa exceção, “Arquivo CSV não encontrado! Favor verificar o diretório user/files”, assim quando for criar uma nova instância desta exception não é necessário passar a mensagem de erro. O problema aqui é que ao lançar essa exception temos que modificar a assinatura do método, como mostra o exemplo abaixo:

```
1  import br.com.mauda.seminario.cientificos.exception.ArquivoCsvNaoEncontradoException;
2
3  //Algumas linhas de código depois dentro de uma classe
```

```
4
5       public void abrirArquivo(String enderecoArquivo) throws ArquivoCsvNaoEncontradoException {
6           if(enderecoArquivo == null){
7               throw new ArquivoCsvNaoEncontradoException();
8           }
9           //continua o método de abrirArquivo
10      }
```

Como forma de evitar essa mudança na assinatura dos métodos, podemos criar uma exception “unchecked”

Como criar uma exception unchecked?

Para criar uma exception unchecked esta precisa ser filha, neta, bisneta, ou seja, descender da *classe* RuntimeException. Ao criar uma *classe* assim, não é necessário modificar a assinatura do método que lança a exception, pois essa exceção não é checada pela linguagem Java. O exemplo abaixo mostra a *classe* SeminariosCientificosException a qual estende a *classe* RuntimeException.

```
1  package br.com.mauda.seminario.cientificos.exception;
2
3  public class SeminariosCientificosException extends RuntimeException {
4
5      private static final long serialVersionUID = 4928599035264976611L;
6
7      public SeminariosCientificosException(String message) {
8          super(message);
9      }
10
11     public SeminariosCientificosException(Throwable t) {
12         super(t);
13     }
14 }
```

Repare que temos dois construtores nesse caso, um que recebe a mensagem de erro e outro que recebe um Throwable, a super*classe* de todas as exceptions. Para mostrar que não há modificações no método validateForDataModification() abaixo, o código abaixo exibe um lançamento de uma SeminariosCientificosException.

```
1      public void validateForDataModification() {
2          if (this.nome == null) {
3              throw new SeminariosCientificosException("Nome deve estar preenchido");
4          }
5      }
```

Dessa forma podemos aplicar essa exception em qualquer código existente sem se preocupar com alterações nas assinaturas dos métodos.

finally {

Existem inúmeras oportunidades para criar uma exception nova que melhore nosso código. Os exemplos acima são bem básicos, sendo possível ser construído em qualquer projeto. Agora imagine a criação de uma exception que aceitasse uma chave composta para criar uma mensagem de erro complexa, ou ainda atributos que facilitem a identificação e futura extração de informações de um lugar central para tratamento de erros.

Duvidas ou sugestões? Deixe seu feedback! Isso ajuda a saber a sua opinião sobre os artigos e melhorá-los para o futuro! Isso é muito importante!

Até um próximo post!