

02

Debugando uma exceção

Transcrição

Quando estamos trabalhando em uma equipe ou com um código desenvolvido por outra pessoa, é interessante "versioná-lo". É comum que ele tenha um erro ou outro, como neste caso. Vamos rodar esta aplicação em modo normal e não em *Debug*, clicando com o lado direito do mouse, e "Run As > Java Application".

O console nos mostra a mensagem de que foi dado um desconto de mais R\$20 no Processador Intel Core I7, e logo abaixo há uma mensagem de erro referente a uma exceção na *thread main*, em `java.lang.NumberFormatException`, cuja localização exata desconhecemos.

Como é que podemos, a partir de uma exceção, encontrar um erro? Primeiro, precisamos rodar a aplicação em modo *Debug* (clicaremos novamente com o botão direito do mouse, e em "Debug As > Java Application"), alterando a perspectiva de Java para *Debug*.

Abre-se a classe `BigDecimal.class`, em que `.class` remete à sua complicaçāo; é uma classe do `JDK`, do *core* do Java, que para na linha 494, em que foi lançada a exceção `NumberFormatException()`, como visto no console. Colocamos algum *breakpoint* nesta classe? Como ela veio parar aqui? Rodamos a classe principal e viemos parar nesta `BigDecimal ...`

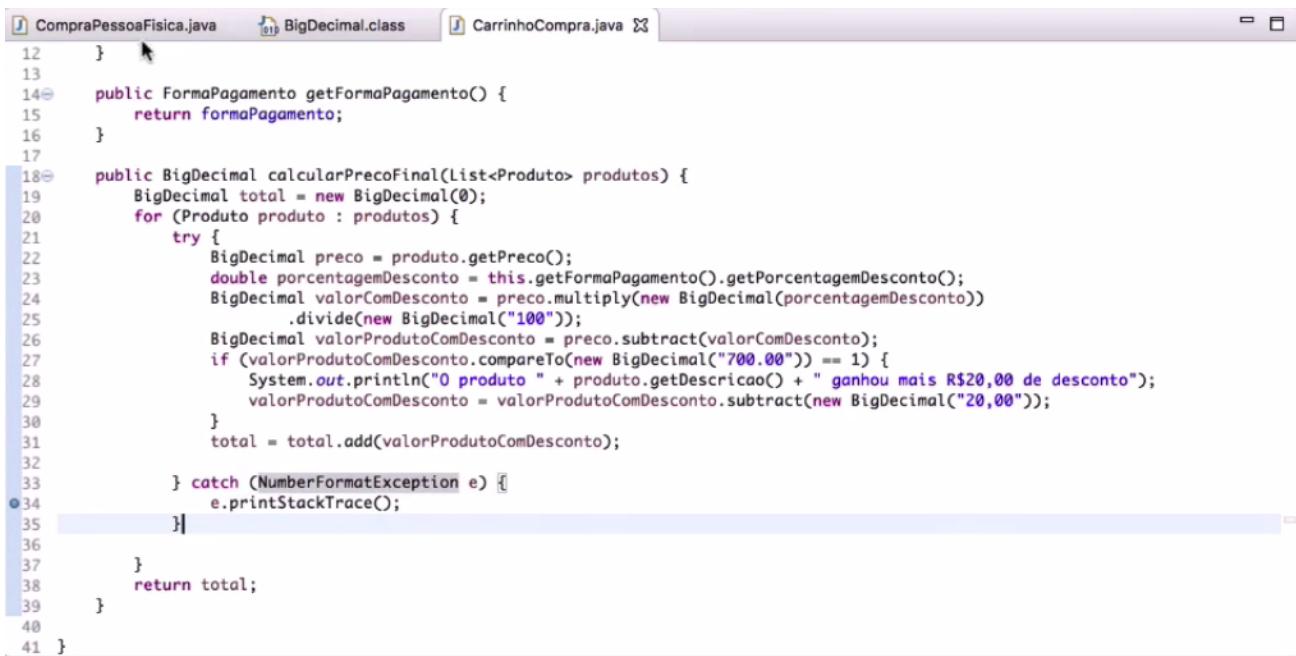
Estamos utilizando o Eclipse na versão 4.6.1 (informação obtida por meio de "Eclipse > Sobre Eclipse"), atualmente a mais recente. Quando se roda a aplicação em modo *Debug*, mesmo sem colocar um *breakpoint*, ela vai parar nas *exceptions* que não foram tratadas (sāo as *uncaught*, de *catch*).

Para tratarmos uma exceção e navegarmos nela até descobrirmos o erro, inicialmente deixaremos a aplicação rodar, apertando F8 ("Resume"). No *flow*, a exceção foi relançada ao console. Como podemos mexer com uma exceção específica? Não quero que todas elas sejam capturadas.

Ao clicarmos em "Eclipse > Preferências > Java > Debug" (no Windows ou Linux, basta clicar em "Window > Preference"), há a primeira opção, "*Suspend execution on uncaught exceptions*" (ou "Suspender a execução em exceções não tratadas"), que desmarcaremos - por padrão, ela vem sempre marcada -, pois queremos fazer isto manualmente. Apertaremos "Apply" para aplicarmos esta alteração, ou simplesmente em "OK".

Iremos ao método `main` na aba `CompraPessoaFísica.java`, clicando com o lado direito do mouse e em "Debug As > Java Application", e a app é rodada sem pausa. Voltaremos à perspectiva Java, repetindo o procedimento de "Debug As > Java Application", e a aplicação roda novamente sem *breakpoint* ou captura de exceção, assim como antes.

Sabemos que o problema se encontra após a mensagem de desconto no valor do processador (o produto), ou seja, no método `calcularPrecoFinal`. Para capturar esta exceção, podemos acrescentar novas linhas ao código:



```

12     }
13
14     public FormaPagamento getFormaPagamento() {
15         return formaPagamento;
16     }
17
18     public BigDecimal calcularPrecoFinal(List<Produto> produtos) {
19         BigDecimal total = new BigDecimal(0);
20         for (Produto produto : produtos) {
21             try {
22                 BigDecimal preco = produto.getPreco();
23                 double porcentagemDesconto = this.getFormaPagamento().getPorcentagemDesconto();
24                 BigDecimal valorComDesconto = preco.multiply(new BigDecimal(porcentagemDesconto))
25                     .divide(new BigDecimal("100"));
26                 BigDecimal valorProdutoComDesconto = preco.subtract(valorComDesconto);
27                 if (valorProdutoComDesconto.compareTo(new BigDecimal("700.00")) == 1) {
28                     System.out.println("O produto " + produto.getDescricao() + " ganhou mais R$20,00 de desconto");
29                     valorProdutoComDesconto = valorProdutoComDesconto.subtract(new BigDecimal("20,00"));
30                 }
31                 total = total.add(valorProdutoComDesconto);
32             } catch (NumberFormatException e) {
33                 e.printStackTrace();
34             }
35         }
36     }
37     }
38     return total;
39 }
40 }
41 }

```

Voltaremos à aba `CompraPessoaFisica` para rodarmos a aplicação em perspectiva *Debug* clicando com o botão direito do mouse, "Debug As > Java Application". Por haver um *breakpoint*, ela pausa no momento que determinamos.

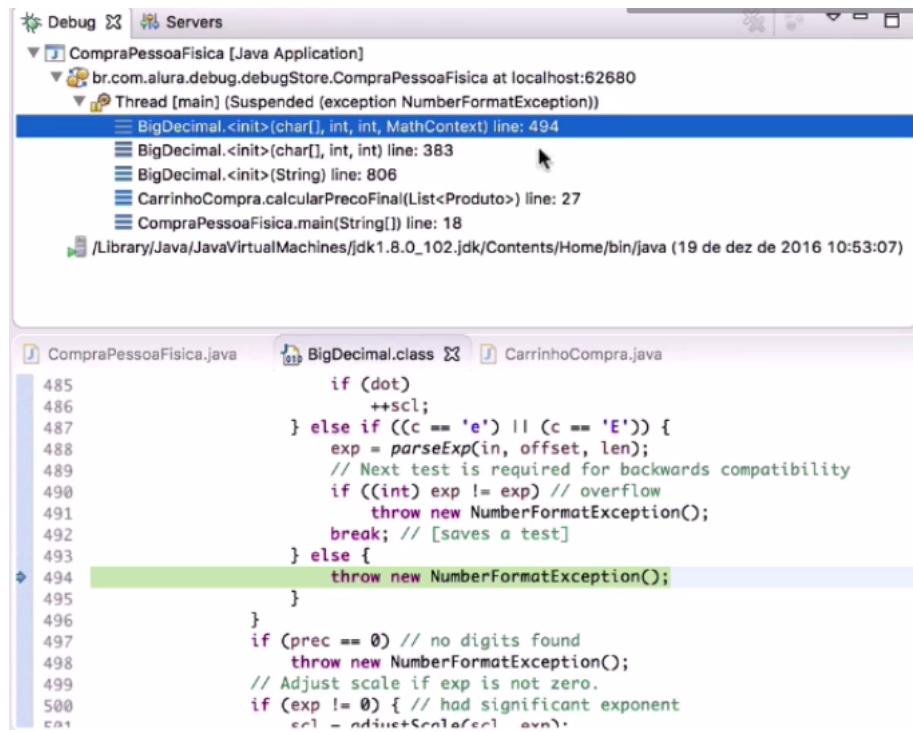
Qual é a desvantagem de se usar esta abordagem? Escrevemos um código a mais para capturar um erro que nem sempre existe. Neste caso funciona, porém o código a mais não deveria existir, sendo preferível tratá-lo de outras maneiras. Quanto menos código, melhor, caso contrário, será cada vez mais difícil implementar melhorias à aplicação.

Por querermos capturar apenas a exceção quando estivermos debugando, vamos pausá-la por meio do botão "*Terminate*", voltando à perspectiva Java e apertando "Ctrl + Z" para desfazer estas alterações. Quando utilizamos um *breakpoint* para capturar as exceções desejadas, podemos ir à aba denominada *Breakpoints*, existente ao lado de "*Variables*".

Feito isto, clicamos no botão "*Add Java Exception Breakpoint*", ou seja, iremos colocar um *breakpoint* que depende de uma exceção específica. Este botão abre uma nova janela que permite a seleção da classe a ser capturada.

Buscamos nesta lista o `NumberFormatException`, e um *breakpoint* condicional é colocado para esta exceção. Agora, podemos rodar em *Debug* para verificar se isto realmente acontece. Voltando à perspectiva Java, à aba `CompraPessoaFisica.java`, apertaremos "Debug As > Java Application". Indo à perspectiva *Debug*, a aplicação pausa na aba `BigDecimal.class`, exatamente em `NumberFormatException`, pois assim definimos.

A partir disto, pode-se verificar em qual *thread* ou parte do código se encontra o bug. Na aba *Debug* da perspectiva de mesmo nome, na parte superior à esquerda, temos os lugares por onde o código passou e, neste momento, ele se encontra aqui:



The screenshot shows the Eclipse IDE interface with the Java Debug perspective selected. In the top left, there are tabs for 'Debug' and 'Servers'. Below them, a tree view shows a project named 'CompraPessoaFísica [Java Application]' with a child node 'br.com.alura.debug.debugStore.CompraPessoaFísica at localhost:62680'. Under this node, a 'Thread [main] (Suspended (exception NumberFormatException))' is expanded, revealing the call stack. The stack trace includes frames from 'BigDecimal.<init>(char[], int, int, MathContext) line: 494', 'BigDecimal.<init>(char[], int, int) line: 383', 'BigDecimal.<init>(String) line: 806', 'CarrinhoCompra.calcularPrecoFinal(List<Produto>) line: 27', and 'CompraPessoaFísica.main(String[]) line: 18'. The bottom status bar indicates the path '/Library/Java/JavaVirtualMachines/jdk1.8.0_102.jdk/Contents/Home/bin/java' and the date '19 de dez de 2016 10:53:07'.

Below the debug view, the code editor displays the 'CompraPessoaFísica.java' file. Line 494 is highlighted with a green background, and the cursor is positioned there. The code for line 494 is:

```

485         if (dot)
486             ++scl;
487     } else if ((c == 'e') || (c == 'E')) {
488         exp = parseExp(in, offset, len);
489         // Next test is required for backwards compatibility
490         if ((int) exp != exp) // overflow
491             throw new NumberFormatException();
492         break; // [saves a test]
493     } else {
494         throw new NumberFormatException();
495     }
496     if (prec == 0) // no digits found
497         throw new NumberFormatException();
498     // Adjust scale if exp is not zero.
499     if (exp != 0) { // had significant exponent
500         scl = adjustScale(scl, exp);

```

Assim, os erros são empilhados: por exemplo, na linha 383 criou-se um construtor, recebendo uma *string*. Antes disto, outro construtor foi chamado (por meio de `this`), o qual, por sua vez, lançou o *exception*. O erro, capturado através de uma exceção, está na linha 27, em um construtor. Há uma vírgula sendo usada no lugar de um ponto final, como deveria ser.

Já temos o erro, porém vamos continuar rodando a aplicação, apertando "Resume" em seguida. O erro é lançado em `NumberFormatException`, o corrigimos, salvamos e rodados mais uma vez. Não se faz pausa em momento algum pois não há mais erros, e mostra-se a mensagem de desconto e valor total da compra. Agora, sabemos colocar um *breakpoint* condicional à uma exceção.