INF335 – Ambientes para Concepção de Software

Formação de Especialista em Engenharia de Software

INSTITUTO DE COMPUTAÇÃO – UNICAMP

1° SEMESTRE DE 2024

Prof. Rodrigo Bonacin rbonacin@unicamp.br

08/Junho/2024 TRABALHO 1

1 Objetivos

Esta atividade tem como objetivo desenvolver habilidades e avaliar o uso da IDE (Eclipse ou IntelliJ IDEA) e depurador para a detecção e correção de *bugs*. Este trabalho compõe a avaliação da disciplina INF335, e poderá ser realizado individualmente ou em dupla.

2 Atividade

Baixar código no moodle na mesma pasta desta atividade.

Este código contém 2 classes:

- *ProdutoBean.java* no pacote *br.unicamp.ic.inf335.beans*, que implementa um Java Bean com dados básicos de um produto.
- *Brecho.java* no pacote *br.unicamp.ic.inf335*, que executa o método *main* e (deveria) incluir 6 produtos, listar os produtos, ordenar os produtos, listar produtos ordenados e calcular e imprimir a média de valor dos produtos.

Essas classes contêm *bugs* que devem ser deputados e corrigidos. Para tanto:

1. Criar projeto java no workspace do Eclipse (ou IntelliJ) e incluir classes em seus respectivos pacotes.
2. Utilizar depurador do Eclipse (ou IntelliJ) para corrigir os erros (importante: como o objetivo é o uso do depurador, mesmo que perceba *bugs* por meio da leitura do código eles devem ser visualizados no depurador).
3. Após o uso do depurador, corrigir *bugs*.

3 Entrega

Os seguintes itens devem ser entregues:

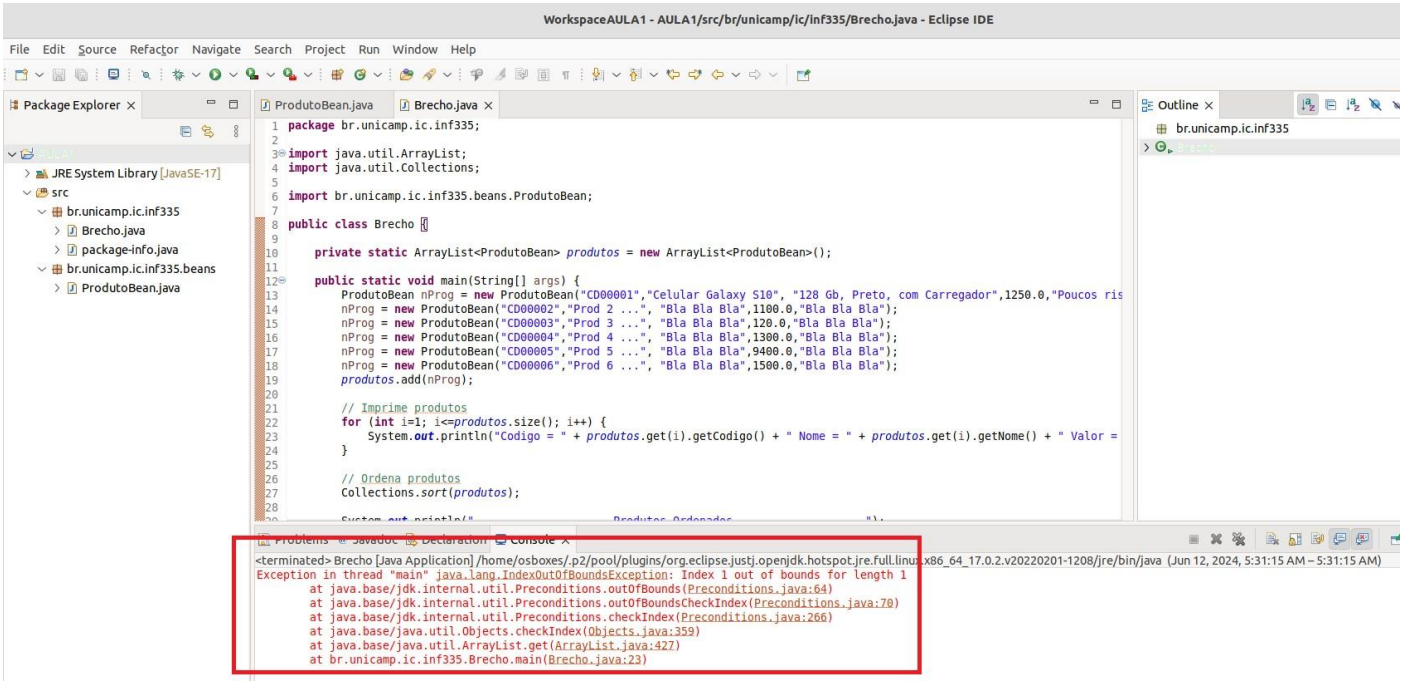
1. Relatório com cada tela (*print*) do depurador exibindo claramente o uso da ferramenta.
2. Código fonte corrigido.

Deverá ser entregue um arquivo por dupla, constando os respectivos nomes completos. As entregas deverão ser realizadas exclusivamente pelo Moodle.

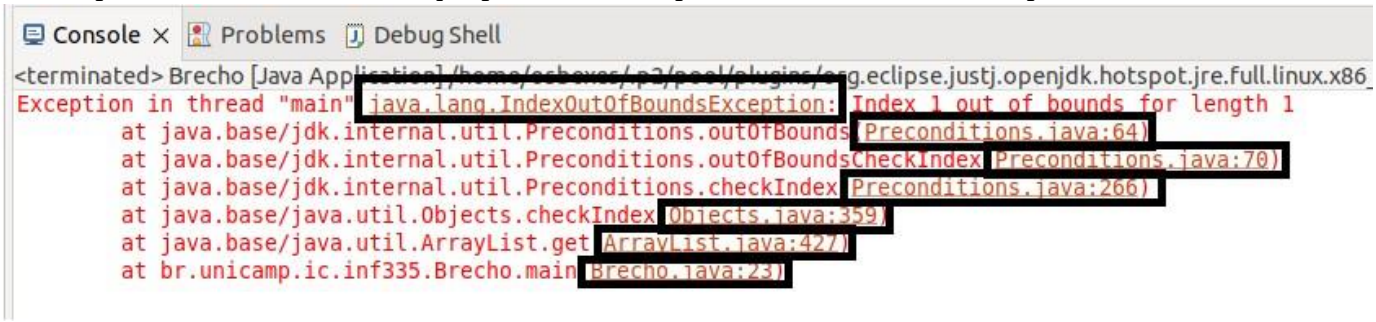
DATA FINAL DE ENTREGA: 14/06/2024

1. Relatório com cada tela (*print*) do depurador exibindo claramente o uso da ferramenta

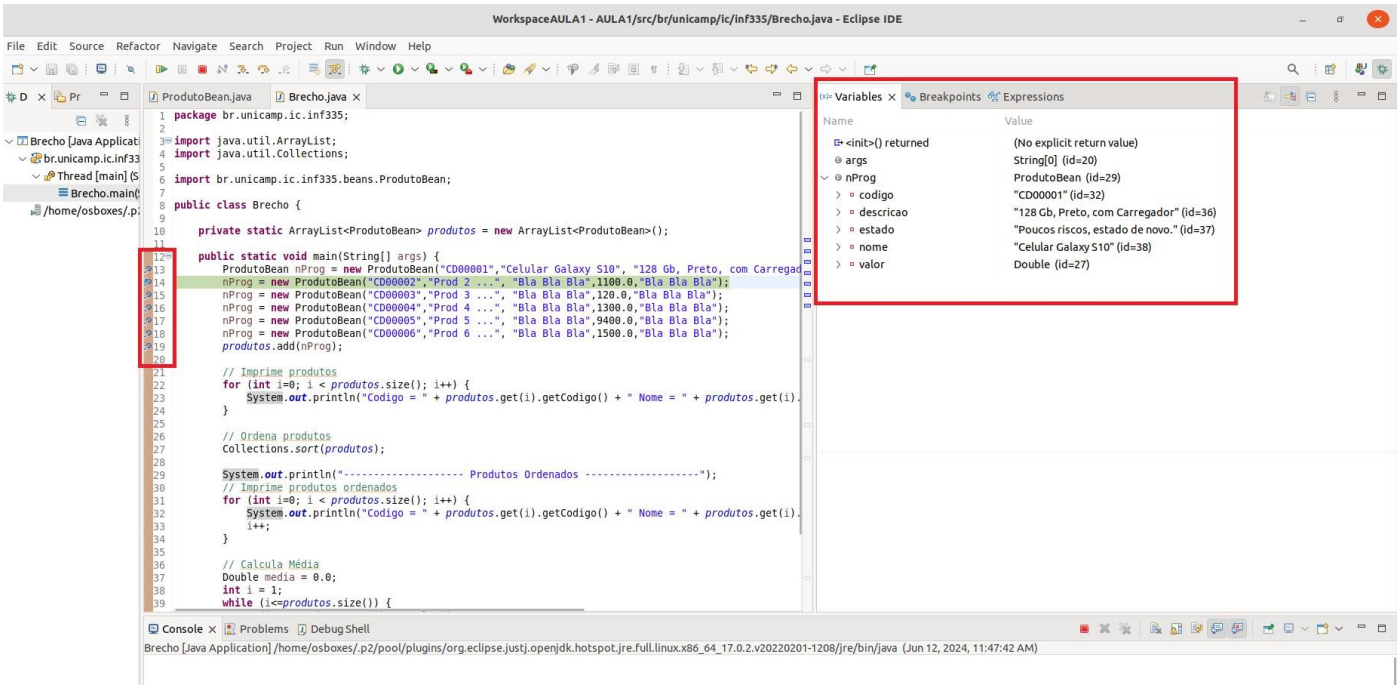
O projeto java foi criado no workspace do Eclipse e incluído classes em seus respectivos pacotes. A primeira compilação apresentou os seguintes erros:



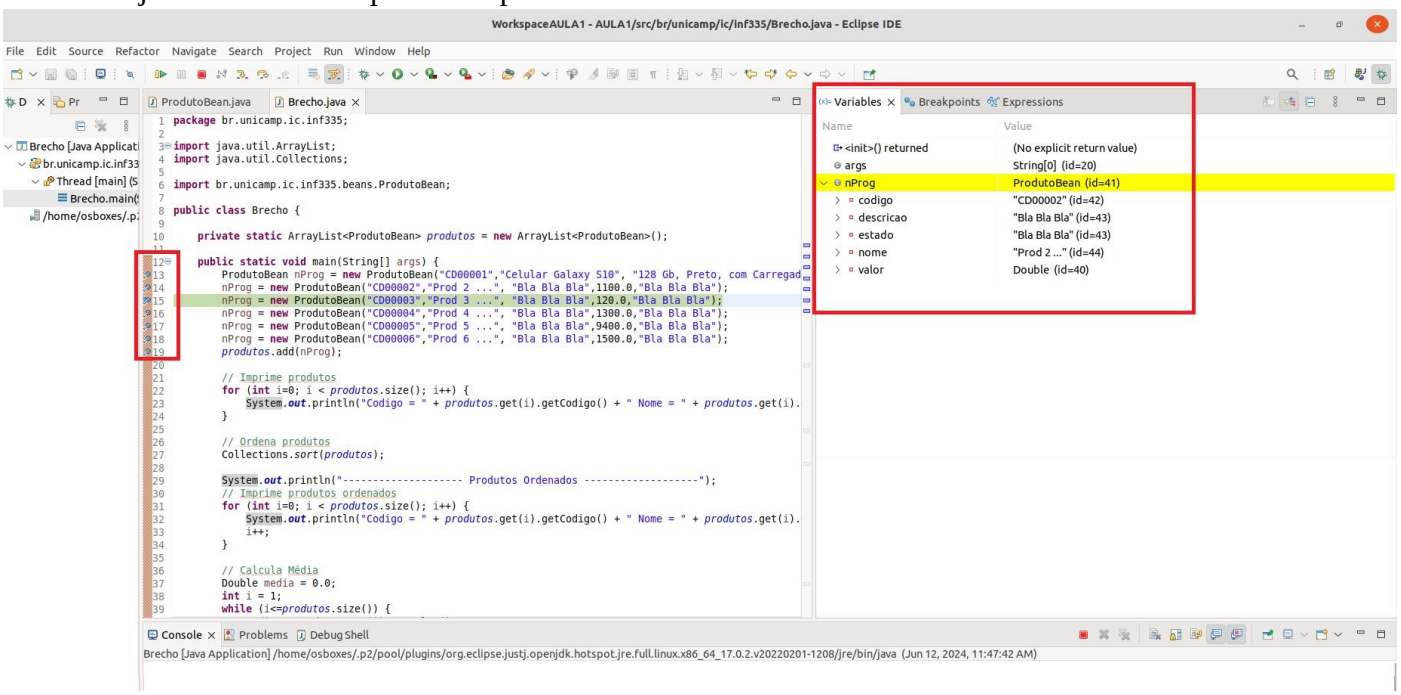
Os erros apresentados contêm links que permitem uma primeira análise dos erros apresentados.



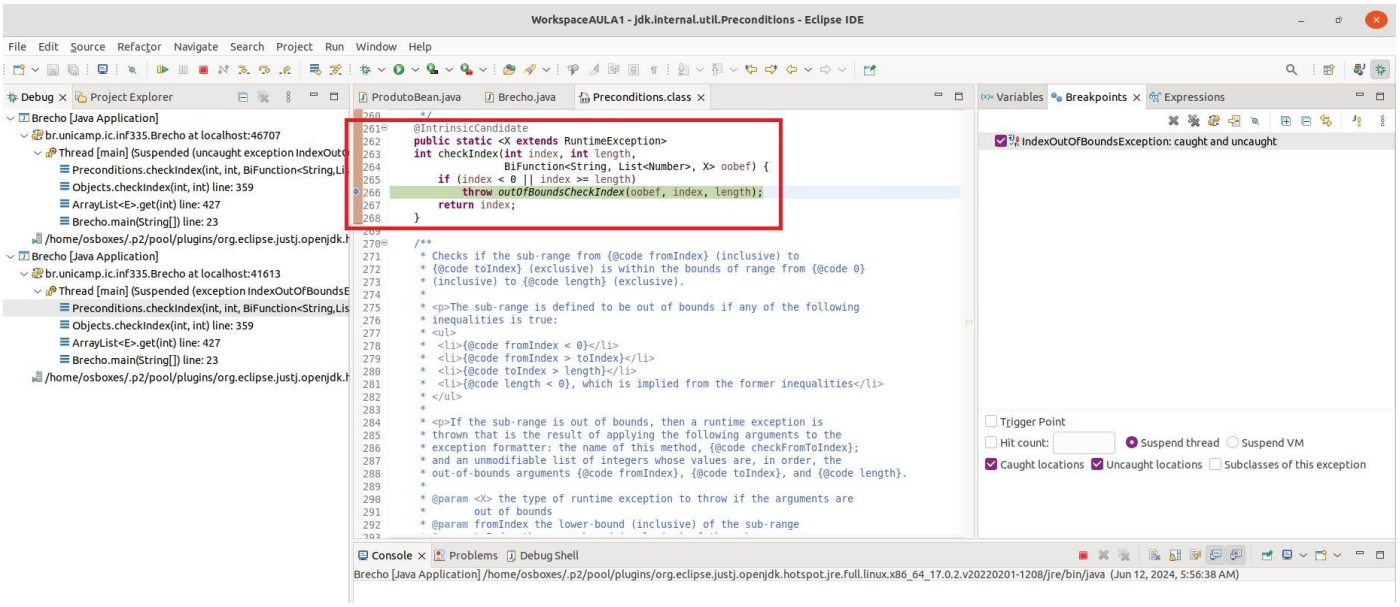
Para realizar debug inicial e verificar a criação de objetos foi utilizado o comando “Step Over (F6)” e criado break points. Na primeira linha 13 é instanciado o objeto nProg.



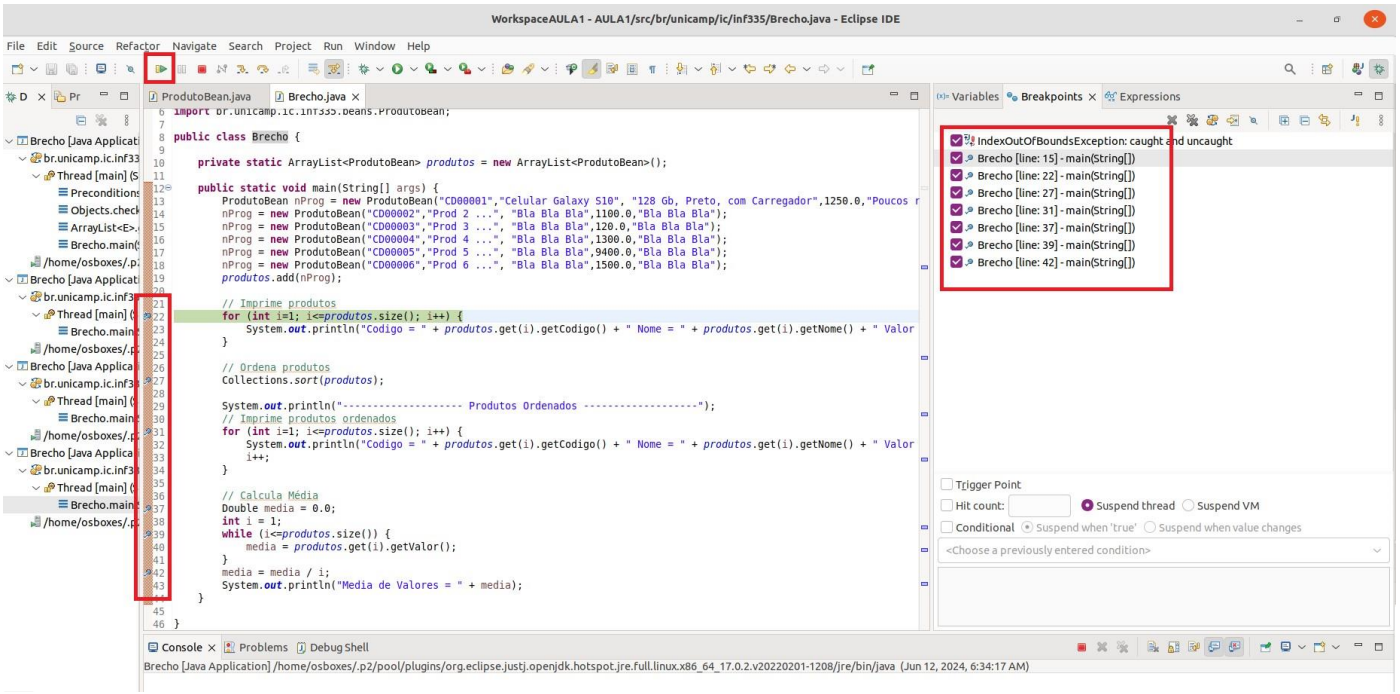
Já na linha 14, 15, 16, 17 e 18 a referência do objetivo nProd é sobrescrita. O que deve ser feito é instanciar um novo objeto ProdutoBean para cada produto.



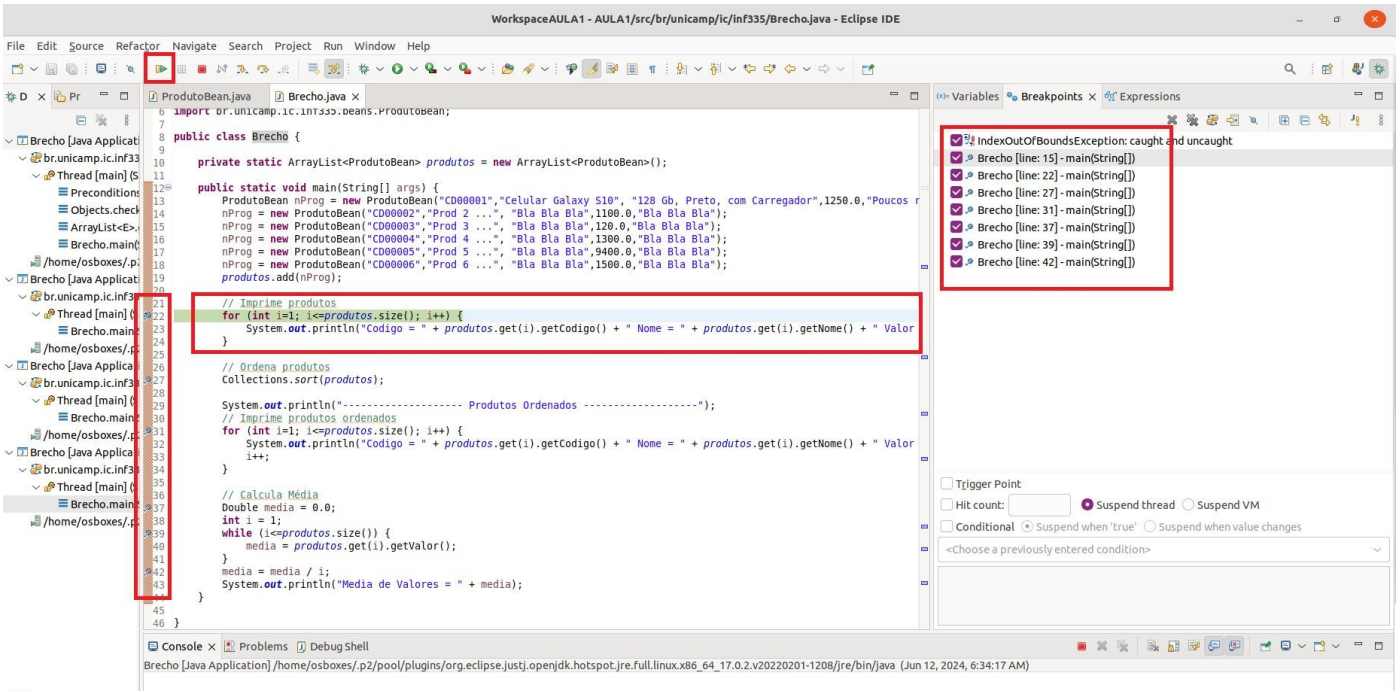
Ao eliminar os break points criados anteriormente foi executado o modo debug novamente e a execução foi interrompido, indicando uma exception OutOfBoundsException no código. Na pesquisa realizada sobre o exception é possível verificar que o método, onde ocorre o break, fornece uma maneira genérica de verificar se um índice está dentro dos limites de um determinado tamanho de comprimento.

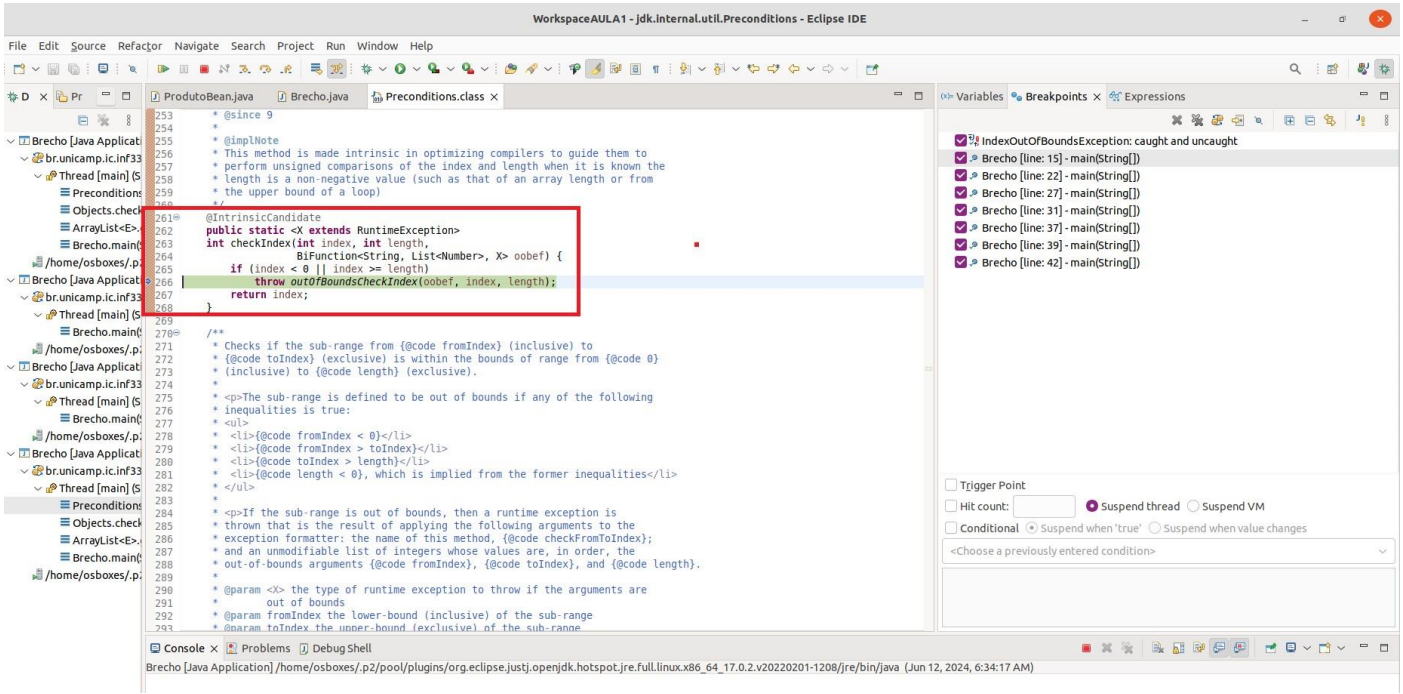


A indicação do erro de índice obriga analisar os pontos da lógica que usam índice, criando break points nas linhas antes e após utilização de índice. Com a utilização do comando “Resume (F8)” é possível parar a execução nos pontos de break e identificar a linha que possui erro de índice



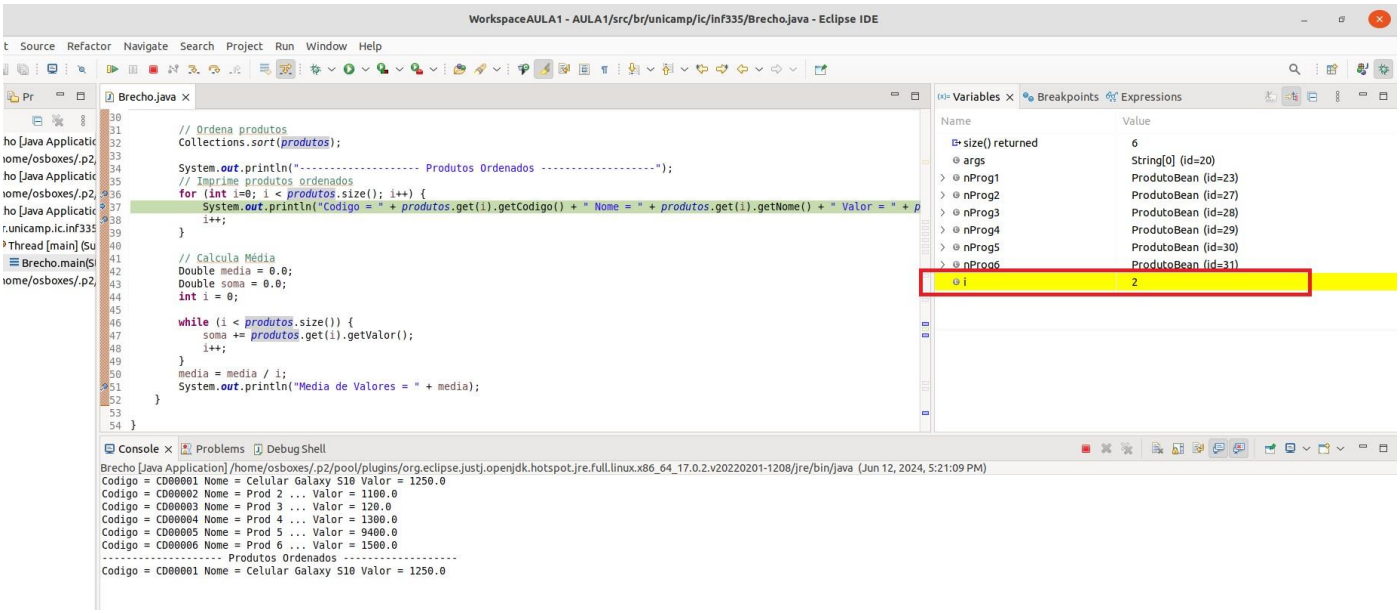
Na execução da sequência de comandos “Resume (F8)”. Ao executar o *for* da linha 22 é apresentada a mesma exceção descrita anteriormente OutOfBoundsException.



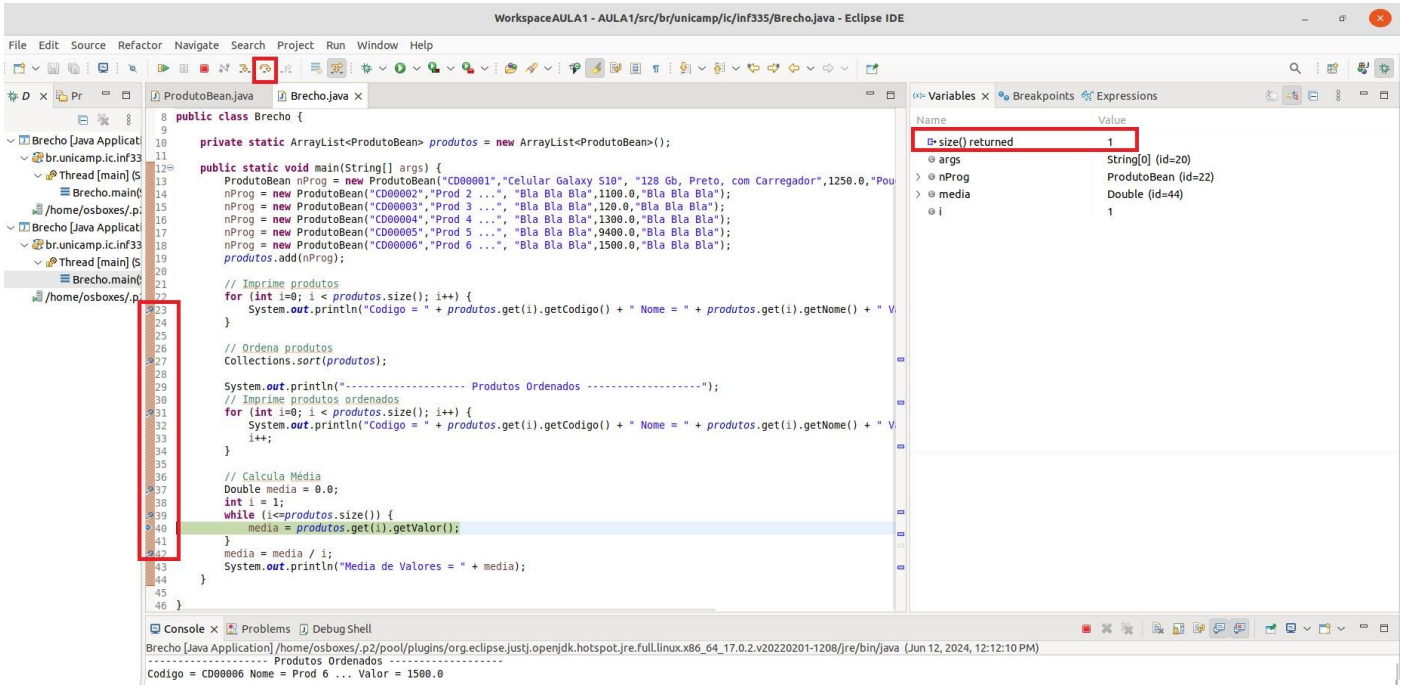


No *for* "imprime produtos", o índice está começando com valor 1 e o correto em Java é iniciar com valor 0. Além disso, o valor da condição de execução do *for* deve obedecer uma comparação válida. Isso faz com que o loop *for* ultrapasse o limite de tamanho da lista, causando a exceção OutOfBoundsException. O mesmo ocorre no *for "imprime produtos ordenados"*.

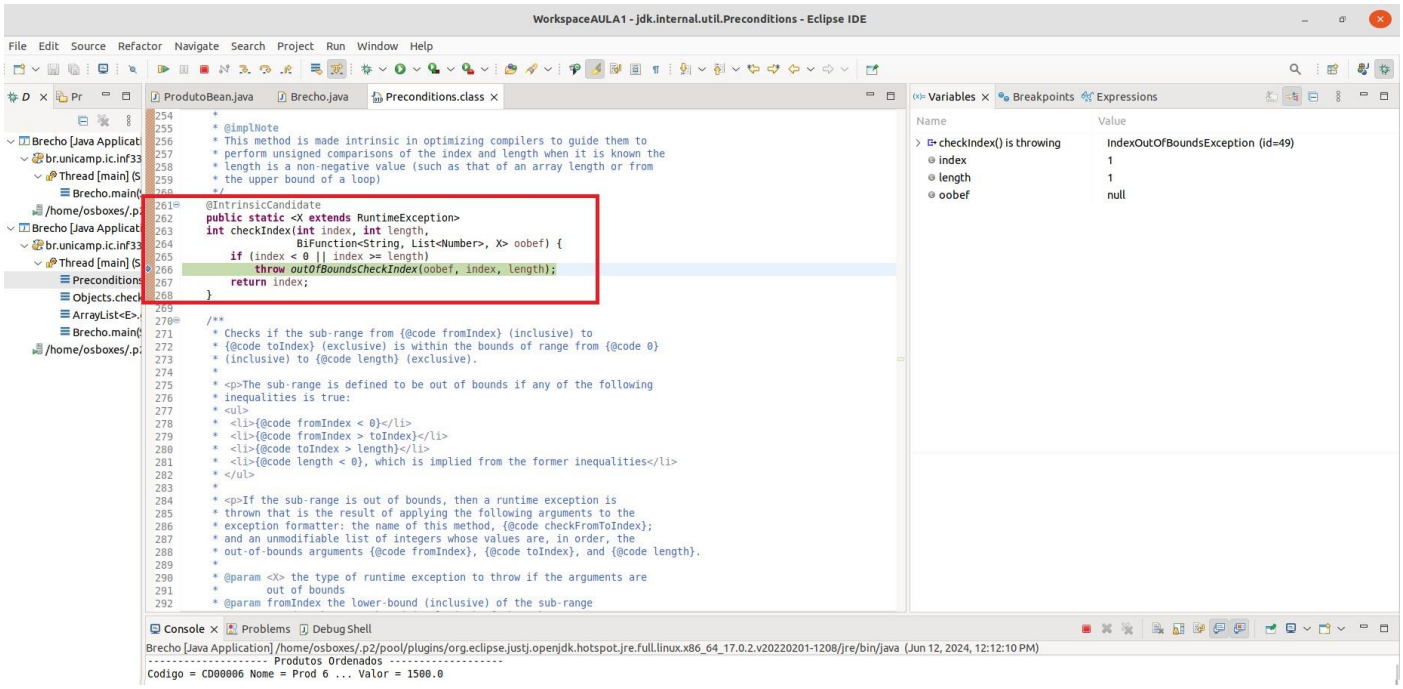
No *for* "imprime produtos ordenados", além do erro de índice, é possível observar incremento duplo da variável i. Na segunda execução deste *for*, antes do segundo println ser executado, o valor da variável i é 2, portanto será impresso posição 2 do array e não 1.



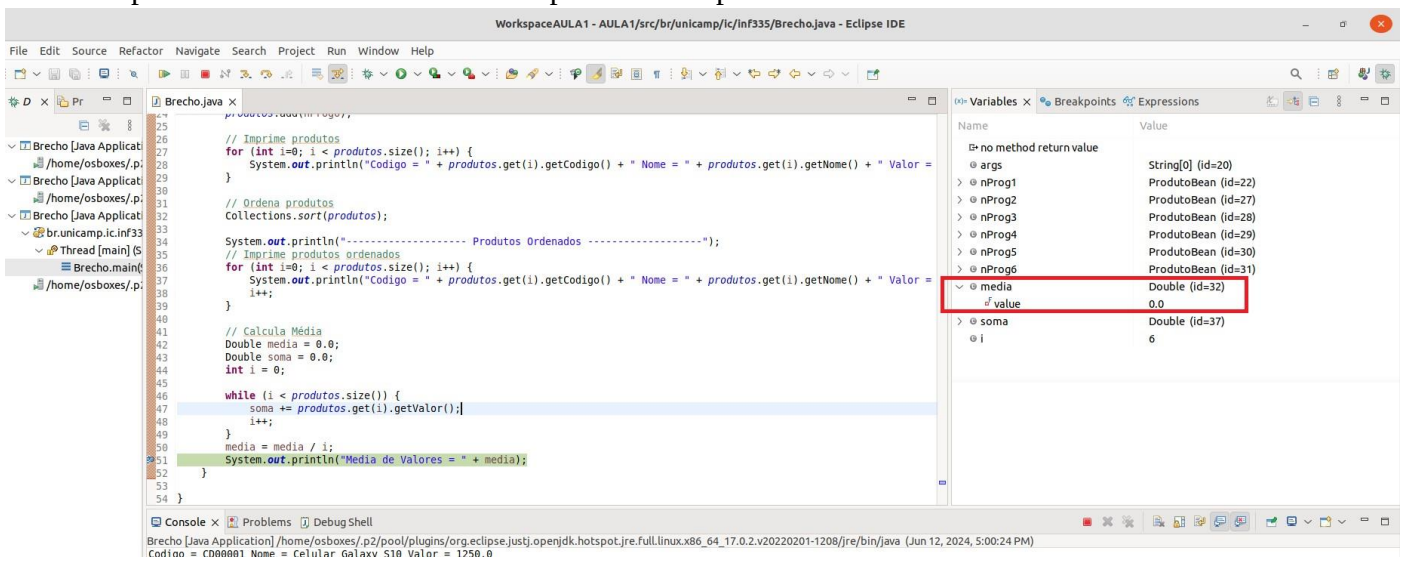
Para realizar debug no laço While foi utilizado o comando “Step Over (F6)”. Por meio do campo Variables, é possível observar a inicialização da variável i.



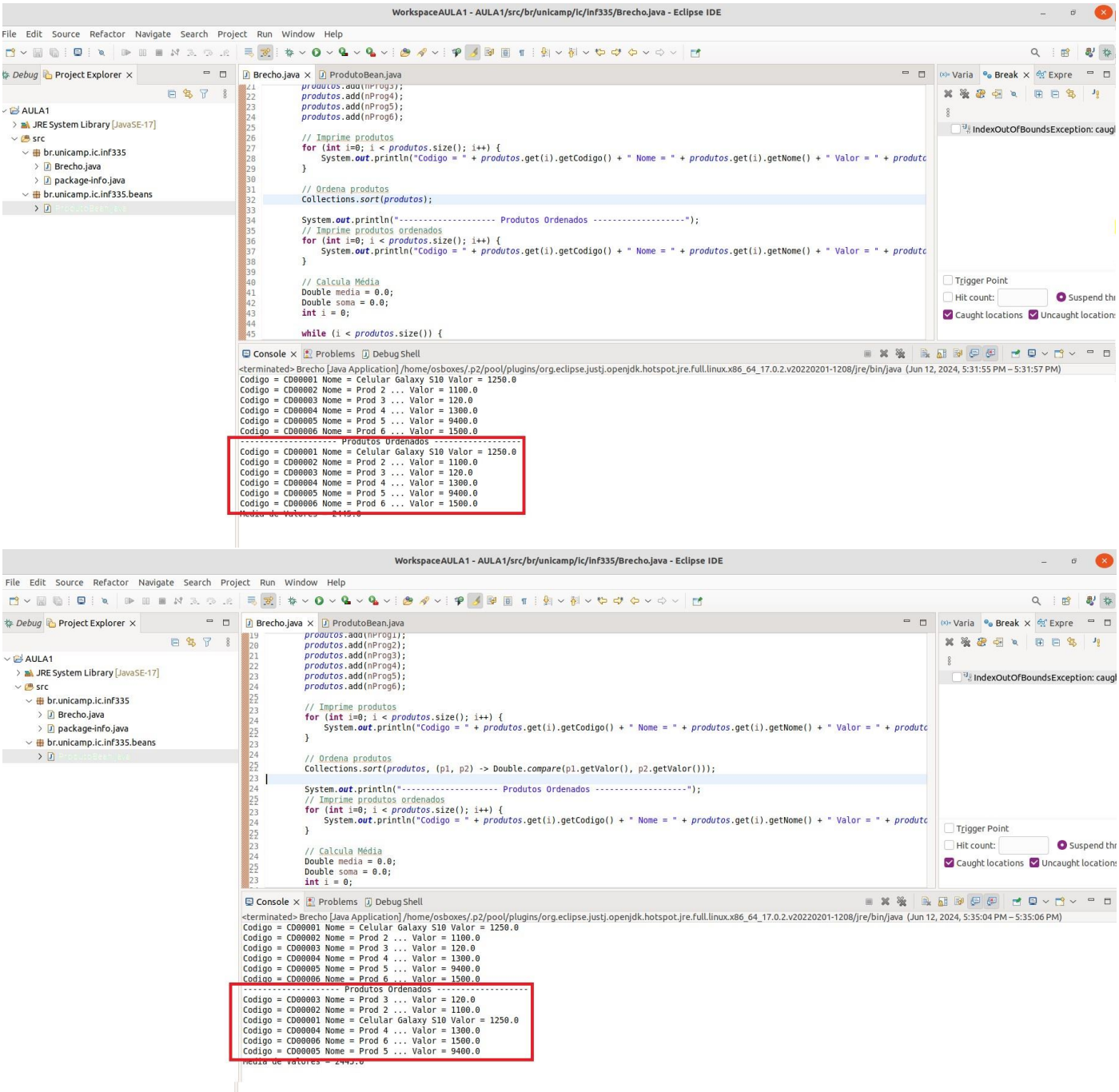
Ao executar a linha 40 ocorreu novamente a exceção outOfBoundsCheckIndex, descrita anteriormente. O correto seria ter inicializado a variável i com valor 0, para que índice buscado nesta linha existisse.



Após correção e inicialização da variável i com valor 0, é possível observar utilizando o comando “Step Over (F6)” que não existe condição de incremento do valor da variável i. Portanto, o While atual é infinito e precisa de uma condição de parada. Além disso, o valor da média não está sendo calculado, pois deveria ser somado todas os valores dentro do laço While para posterior cálculo da média. O valor da média permanece 0 após cálculo, representando erro no cálculo. A forma correta seria somar todos os valores dentro do While e utilizar essa soma para cálculo da média dividindo pelo total de produtos.



Após correções dos itens já citados, os produtos ainda não são apresentados ordenados por preço. Assim, foi feito ajusta para que produtos sejam impressos ordenados e finalizadas alterações dos erros encontrados por debug.



2. Código fonte corrigido

```
package br.unicamp.ic.inf335;

import java.util.ArrayList;
import java.util.Collections;

import br.unicamp.ic.inf335.beans.ProdutoBean;

public class Brecho {

	private static ArrayList<ProdutoBean> produtos = new ArrayList<ProdutoBean>();

	public static void main(String[] args) {
				ProdutoBean nProg1 = new ProdutoBean("CD00001","Celular Galaxy S10", "128 Gb, Preto, com
```

```
Carregador",1250.0,"Poucos riscos, estado de novo.");
                    ProdutoBean nProg2 = new ProdutoBean("CD00002","Prod 2 ...", "Bla Bla Bla",1100.0,"Bla Bla Bla");
                    ProdutoBean nProg3 = new ProdutoBean("CD00003","Prod 3 ...", "Bla Bla Bla",120.0,"Bla Bla Bla");
                    ProdutoBean nProg4 = new ProdutoBean("CD00004","Prod 4 ...", "Bla Bla Bla",1300.0,"Bla Bla Bla");
                    ProdutoBean nProg5 = new ProdutoBean("CD00005","Prod 5 ...", "Bla Bla Bla",9400.0,"Bla Bla Bla");
                    ProdutoBean nProg6 = new ProdutoBean("CD00006","Prod 6 ...", "Bla Bla Bla",1500.0,"Bla Bla Bla");
                    produtos.add(nProg1);          produtos.add(nProg2);          produtos.add(nProg3);
          produtos.add(nProg4);          produtos.add(nProg5);          produtos.add(nProg6);

                    // Imprime produtos
                    for (int i=0; i < produtos.size(); i++) {
                              System.out.println("Codigo = " + produtos.get(i).getCodigo() + " Nome = " + produtos.get(i).getNome()
+ " Valor = " + produtos.get(i).getValor());
                    }

                    // Ordena produtos
      Collections.sort(produtos, (p1, p2) -> Double.compare(p1.getValor(), p2.getValor()));

                    System.out.println("---------- Produtos Ordenados ----------");
                    // Imprime produtos ordenados
          for (int i=0; i < produtos.size(); i++) {
                              System.out.println("Codigo = " + produtos.get(i).getCodigo() + " Nome = " + produtos.get(i).getNome()
+ " Valor = " + produtos.get(i).getValor());
                    }

                    // Calcula Média
                    Double media = 0.0;
          Double soma = 0.0;
                    int i = 0;

                    while (i < produtos.size()) {
          soma += produtos.get(i).getValor();
                       i++;
                    }
                    media = soma / produtos.size();
                    System.out.println("Media de Valores = " + media);
          }

}
```