

# INF335 – Ambientes para Concepção de Software

Formação de Especialista em Engenharia de Software

INSTITUTO DE COMPUTAÇÃO – UNICAMP

1º SEMESTRE DE 2024

Prof. Rodrigo Bonacin [rbonacin@unicamp.br](mailto:rbonacin@unicamp.br)

**15/Junho/2024 TRABALHO 2**

## 1 Objetivos

Este trabalho tem como objetivo desenvolver habilidades modelagem com Draw.io e realizar testes unitários com o JUnit. 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 3 classes:

- *ProdutoBean.java* no pacote *br.unicamp.ic.inf335.beans*, que implementa um Java Bean com dados básicos de um produto, incluindo método *compareTo*.
- *AnuncioBean.java* no pacote *br.unicamp.ic.inf335.beans*, que implementa um Java Bean com dados básicos de um anúncio, incluindo produto associado e lista de fotos. Esta classe também possui um desconto associado, o método *getValor* deve retornar o valor do produto menos o percentual de desconto expresso por 0.0 (0% de desconto) a 1.0 (100% de desconto).
- *AnuncianteBean.java* no pacote *br.unicamp.ic.inf335.beans*, que implementa um Java Bean com dados básicos de um anunciante, incluindo nome, CPF e uma lista de anúncios associados. Esta classe possui métodos para adicionar e remover anúncios associados, bem como calcular o valor médio dos anúncios (*valorMedioAnuncios*).

1. Construir diagrama de classes da UML com draw.io para código anexo.
2. Testar usando JUnit, identificar *bugs* e propor correções. Para tanto:
  - Criar classes de teste (uma para cada classe) contemplando as especificações supracitadas com objetivo de encontrar *bugs* (incluindo funcionamento fora do esperado).
  - Executar testes, documentar resultado e propor correções.

## 3 Entrega

Os seguintes itens devem ser entregues:

1. Código fonte das classes JUnit.
2. Relatório com:
  - a. Diagrama de Classe UML
  - b. Cada tela (*print*) da execução dos testes unitários com JUnit e *bugs* detectados.
3. Código fonte corrigido.

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

**DATA FINAL DE ENTREGA: 21/06/2024**

## **2. a. Diagrama de Classe UML**

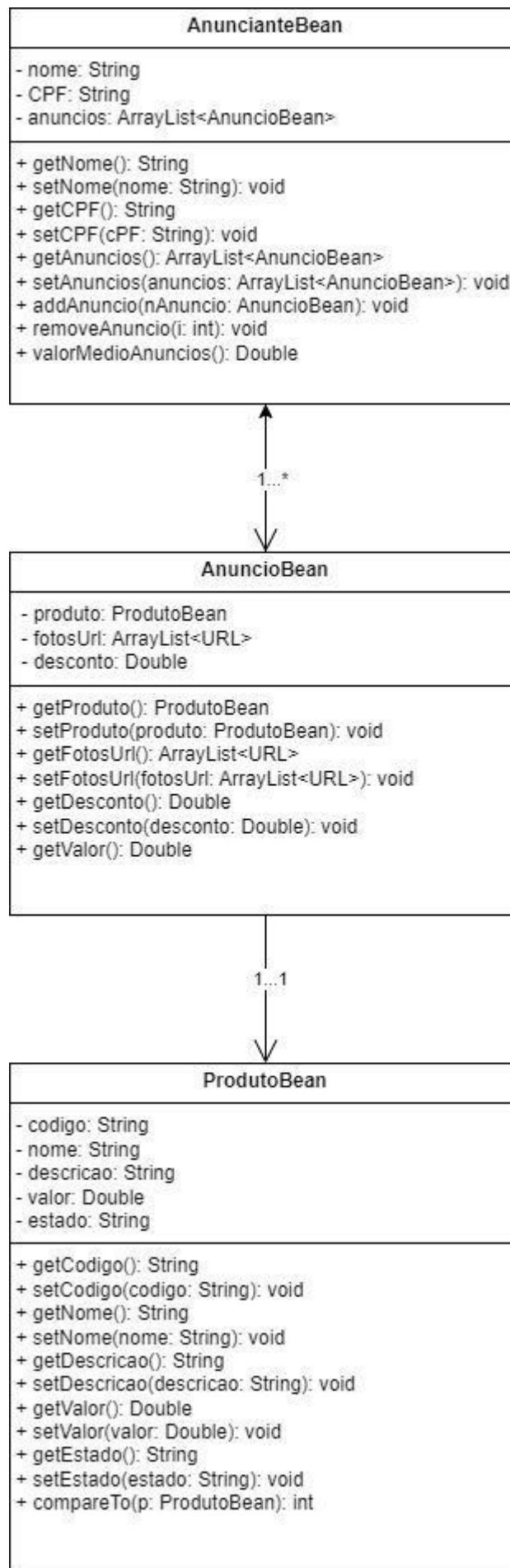
A classe AnuncianteBean possui os atributos nome, CPF e lista de anúncios. Os métodos da classe são os getters/setters para os atributos, adicionar anuncio, remover anuncio e calcular valor médio do anúncio.

A classe AnuncioBean possui os atributos produto, fotos e desconto. Os métodos da classe são os getters/setters para os atributos e cálculo do valor do anúncio considerando o desconto.

A classe ProdutoBean possui os atributos código, nome, descrição, valor e estado. Os métodos da classe são os getters/setters para os atributos e implementa a interface Comparable<produtoBean> para comparação de produtos com base no valor (compareTo).

A classe AnuncianteBean possui associação com a classe AnuncioBean por meio do atributo “anuncio” que é um ArrayList de AnuncioBean. Portanto, um anunciante pode ter vários anúncios.

A classe AnuncioBean possui associação com a classe ProdutoBean por meio do atributo “produto”. Portanto, um anúncio está associado a um único produto.

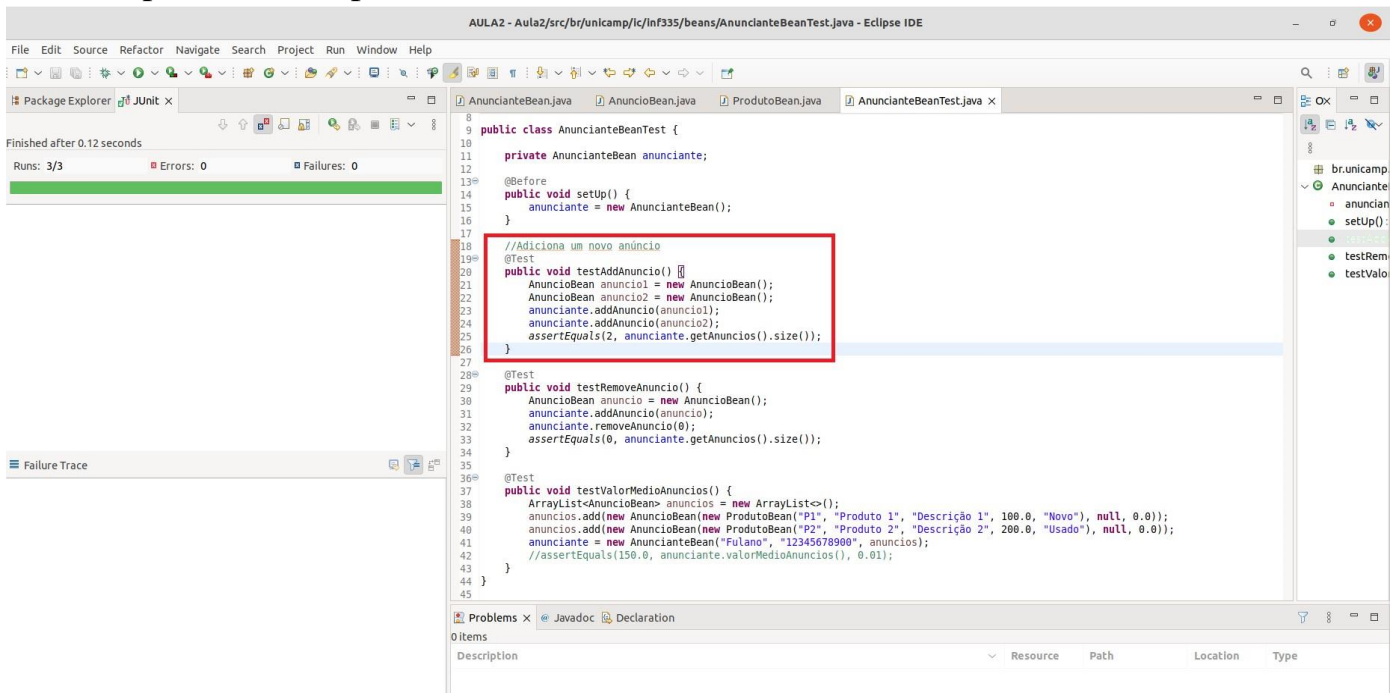


## 2. b. Cada tela (*print*) da execução dos testes unitários com JUnit e *bugs* detectados

Falha é a falha da execução do software o erro é quando ocorre erro na execução do teste.

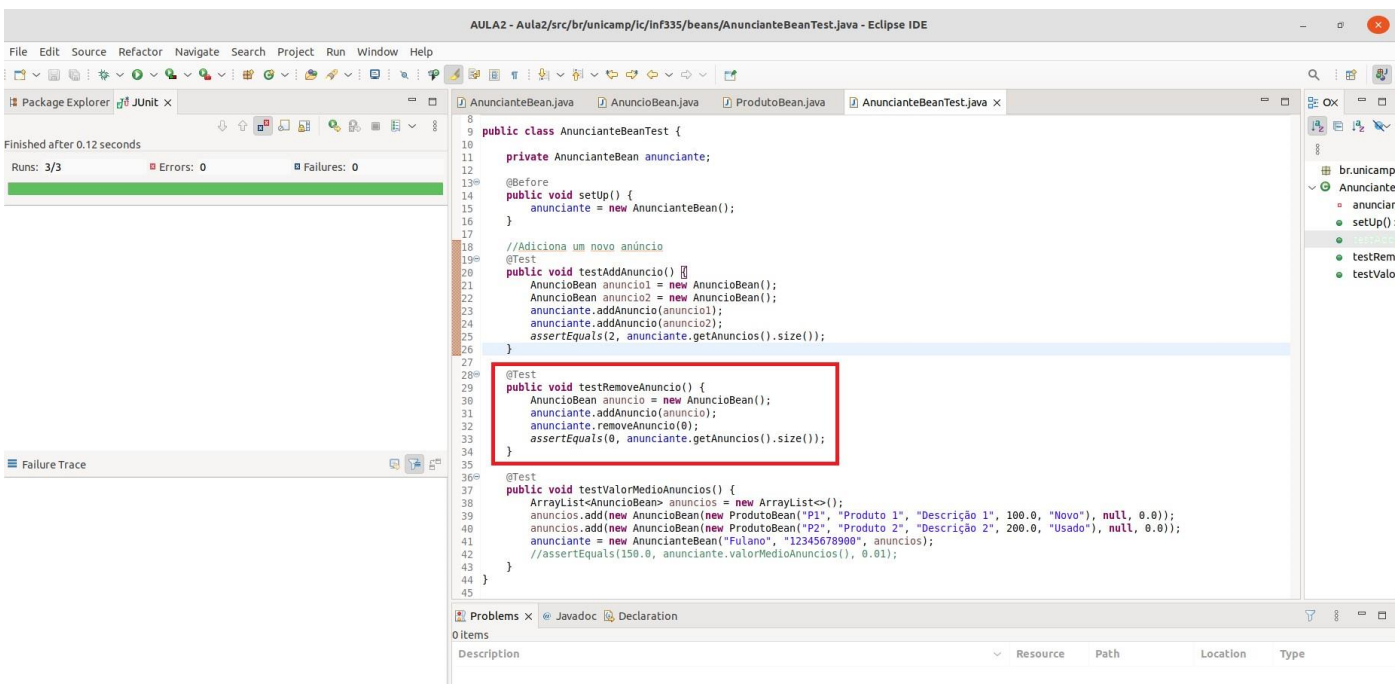
### JUnit para AnuncianteBeanTest:

Para verificar se os anúncios estão sendo inseridos de forma correta na lista, foram adicionados dois anúncios anuncio1 e anuncio2. Após inserir os dois anúncios foi utilizando a função assertEquals. Essa função compara dois valores, o primeiro é o valor esperado e o segundo o retorno esperado. Não apresentou erro ou falha.



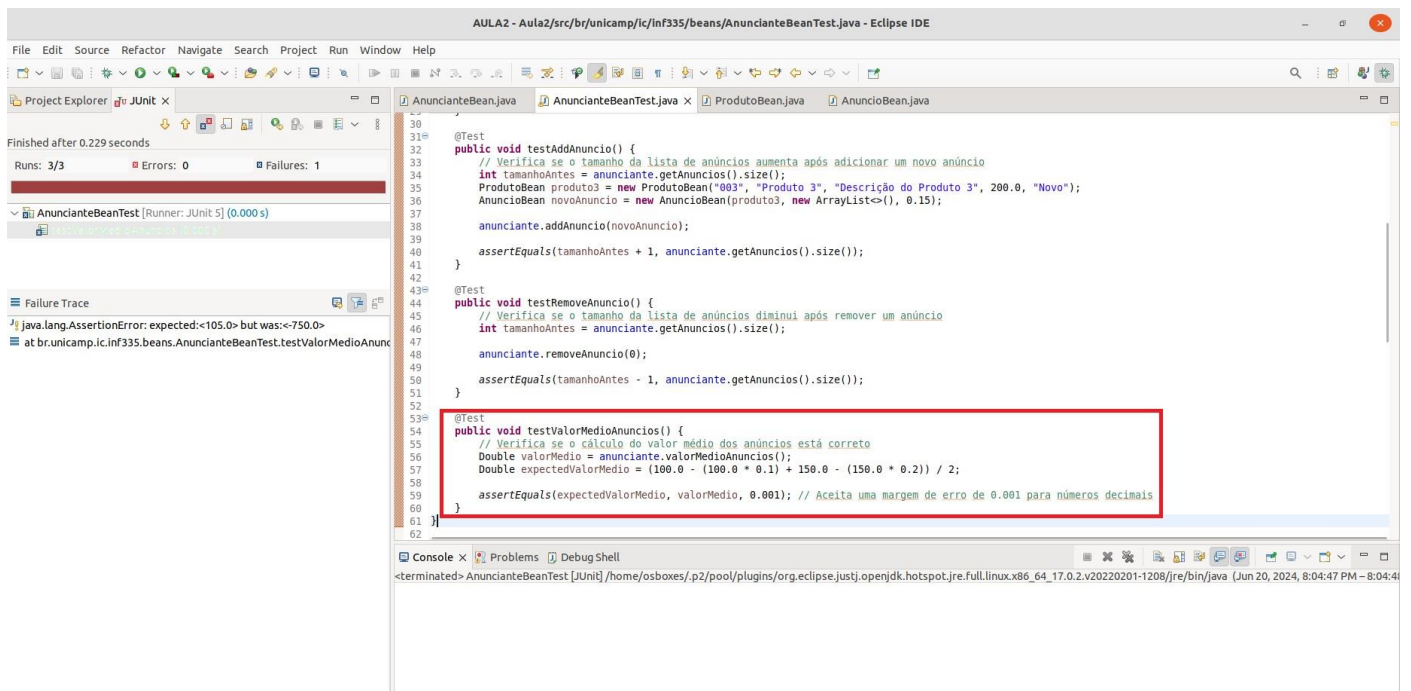
```
8 public class AnuncianteBeanTest {
9
10     private AnuncianteBean anunciante;
11
12     @Before
13     public void setUp() {
14         anunciante = new AnuncianteBean();
15     }
16
17     //Adiciona um novo anúncio
18     @Test
19     public void testAddAnuncio() {
20         AnuncioBean anuncio1 = new AnuncioBean();
21         AnuncioBean anuncio2 = new AnuncioBean();
22         anunciante.addAnuncio(anuncio1);
23         anunciante.addAnuncio(anuncio2);
24         assertEquals(2, anunciante.getAnuncios().size());
25     }
26
27     @Test
28     public void testRemoveAnuncio() {
29         AnuncioBean anuncio = new AnuncioBean();
30         anunciante.addAnuncio(anuncio);
31         anunciante.removeAnuncio(0);
32         assertEquals(0, anunciante.getAnuncios().size());
33     }
34
35     @Test
36     public void testValorMedioAnuncios() {
37         ArrayList<AnuncioBean> anuncios = new ArrayList<>();
38         anuncios.add(new AnuncioBean(new ProdutoBean("P1", "Produto 1", "Descrição 1", 100.0, "Novo"), null, 0.0));
39         anuncios.add(new AnuncioBean(new ProdutoBean("P2", "Produto 2", "Descrição 2", 200.0, "Usado"), null, 0.0));
40         anunciante = new AnuncianteBean("Fulano", "12345678900", anuncios);
41         //assertEquals(150.0, anunciante.valorMedioAnuncios(), 0.01);
42     }
43 }
44
45
```

O mesmo é feito para testar se os anúncios estão sendo removidos de forma correta na lista. Não apresentou erro ou falha.



```
8 public class AnuncianteBeanTest {
9
10     private AnuncianteBean anunciante;
11
12     @Before
13     public void setUp() {
14         anunciante = new AnuncianteBean();
15     }
16
17     //Adiciona um novo anúncio
18     @Test
19     public void testAddAnuncio() {
20         AnuncioBean anuncio1 = new AnuncioBean();
21         AnuncioBean anuncio2 = new AnuncioBean();
22         anunciante.addAnuncio(anuncio1);
23         anunciante.addAnuncio(anuncio2);
24         assertEquals(2, anunciante.getAnuncios().size());
25     }
26
27     @Test
28     public void testRemoveAnuncio() {
29         AnuncioBean anuncio = new AnuncioBean();
30         anunciante.addAnuncio(anuncio);
31         anunciante.removeAnuncio(0);
32         assertEquals(0, anunciante.getAnuncios().size());
33     }
34
35     @Test
36     public void testValorMedioAnuncios() {
37         ArrayList<AnuncioBean> anuncios = new ArrayList<>();
38         anuncios.add(new AnuncioBean(new ProdutoBean("P1", "Produto 1", "Descrição 1", 100.0, "Novo"), null, 0.0));
39         anuncios.add(new AnuncioBean(new ProdutoBean("P2", "Produto 2", "Descrição 2", 200.0, "Usado"), null, 0.0));
40         anunciante = new AnuncianteBean("Fulano", "12345678900", anuncios);
41         //assertEquals(150.0, anunciante.valorMedioAnuncios(), 0.01);
42     }
43 }
44
45
```

Já para testValorMedioAnuncio a função assertEquals apresentou falha. O valor esperado era 105.0, porém o valor médio calculado foi -750.0. Para correção, é importante garantir que o método getValor() na classe AnuncioBean esteja calculando corretamente o valor final do anúncio, levando em conta o desconto aplicado.



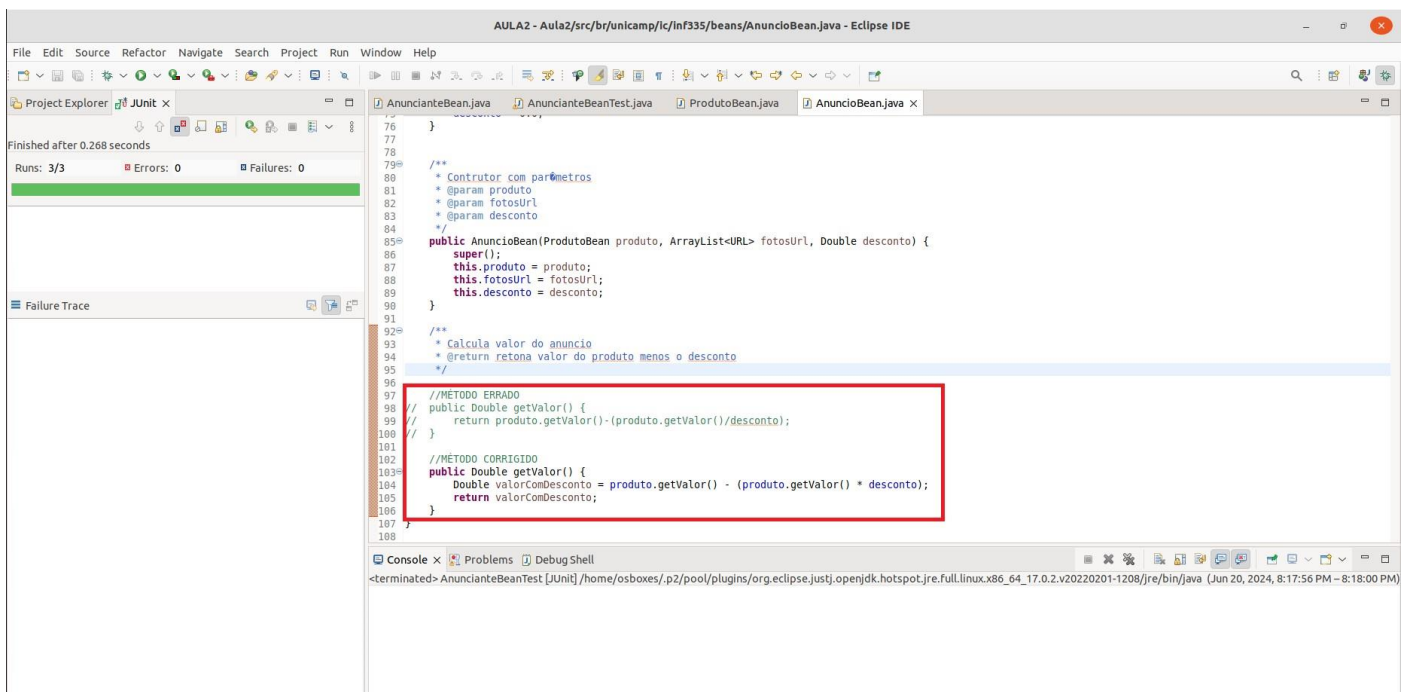
```
31 @Test
32 public void testAddAnuncio() {
33     // Verifica se o tamanho da lista de anúncios aumenta após adicionar um novo anúncio
34     int tamanhoAntes = anunciante.getAnuncios().size();
35     ProdutoBean produto3 = new ProdutoBean("003", "Produto 3", "Descrição do Produto 3", 200.0, "Novo");
36     AnuncioBean novoAnuncio = new AnuncioBean(produto3, new ArrayList<>(), 0.15);
37     anunciante.addAnuncio(novoAnuncio);
38     assertEquals(tamanhoAntes + 1, anunciante.getAnuncios().size());
39 }
40
41
42
43 @Test
44 public void testRemoveAnuncio() {
45     // Verifica se o tamanho da lista de anúncios diminui após remover um anúncio
46     int tamanhoAntes = anunciante.getAnuncios().size();
47     anunciante.removeAnuncio(0);
48     assertEquals(tamanhoAntes - 1, anunciante.getAnuncios().size());
49 }
50
51
52
53 @Test
54 public void testValorMedioAnuncios() {
55     // Verifica se o cálculo do valor médio dos anúncios está correto
56     Double valorMedio = anunciante.valorMedioAnuncios();
57     Double expectedValorMedio = (100.0 - (100.0 * 0.1) + 150.0 - (150.0 * 0.2)) / 2;
58     assertEquals(expectedValorMedio, valorMedio, 0.001); // Aceita uma margem de erro de 0.001 para números decimais
59 }
60
61
62
```

Failure Trace

```
java.lang.AssertionError: expected:<105.0> but was:<-750.0>
at br.unicamp.ic.inf335.beans.AnuncianteBeanTest.testValorMedioAnuncios(AnuncianteBeanTest.java:58)
```

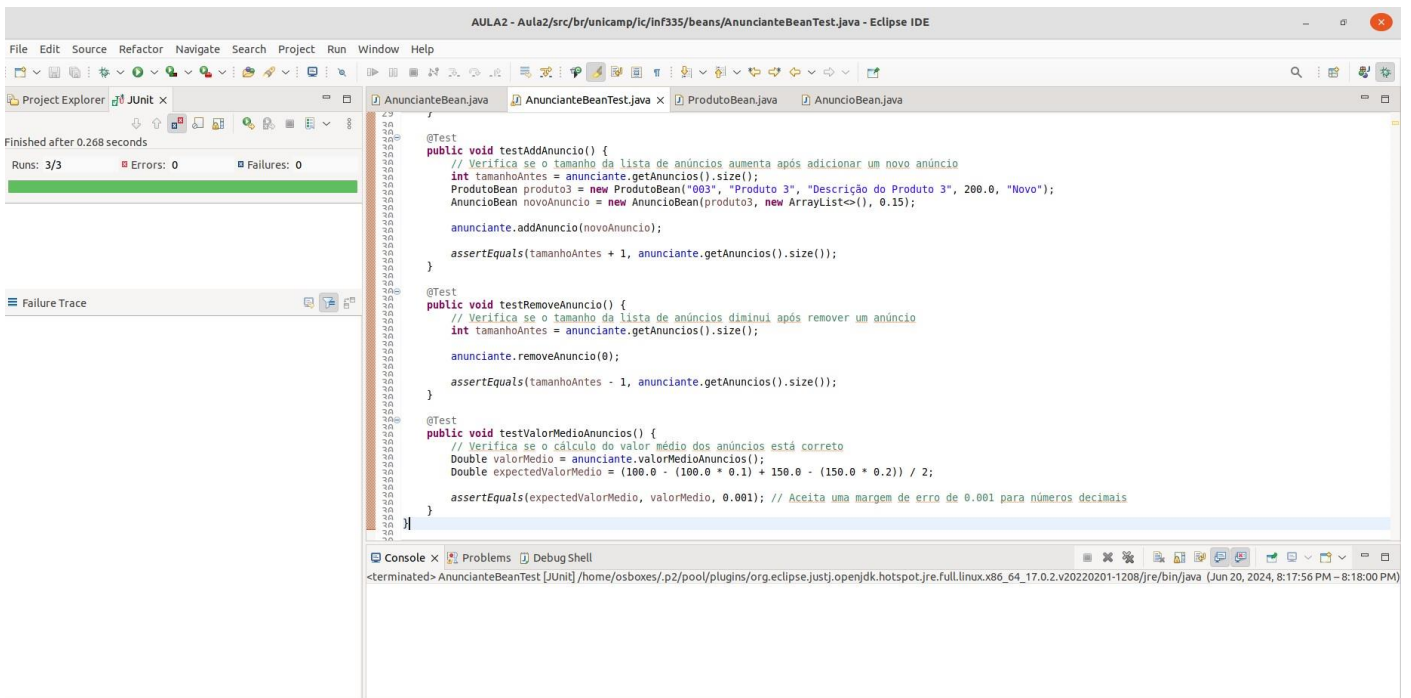
## JUnit para AnuncioBeanTeste:

Conforme apresentado no tópico anterior, o valor médio dos anúncios deve ser calculado corretamente, refletindo os descontos aplicados nos produtos.



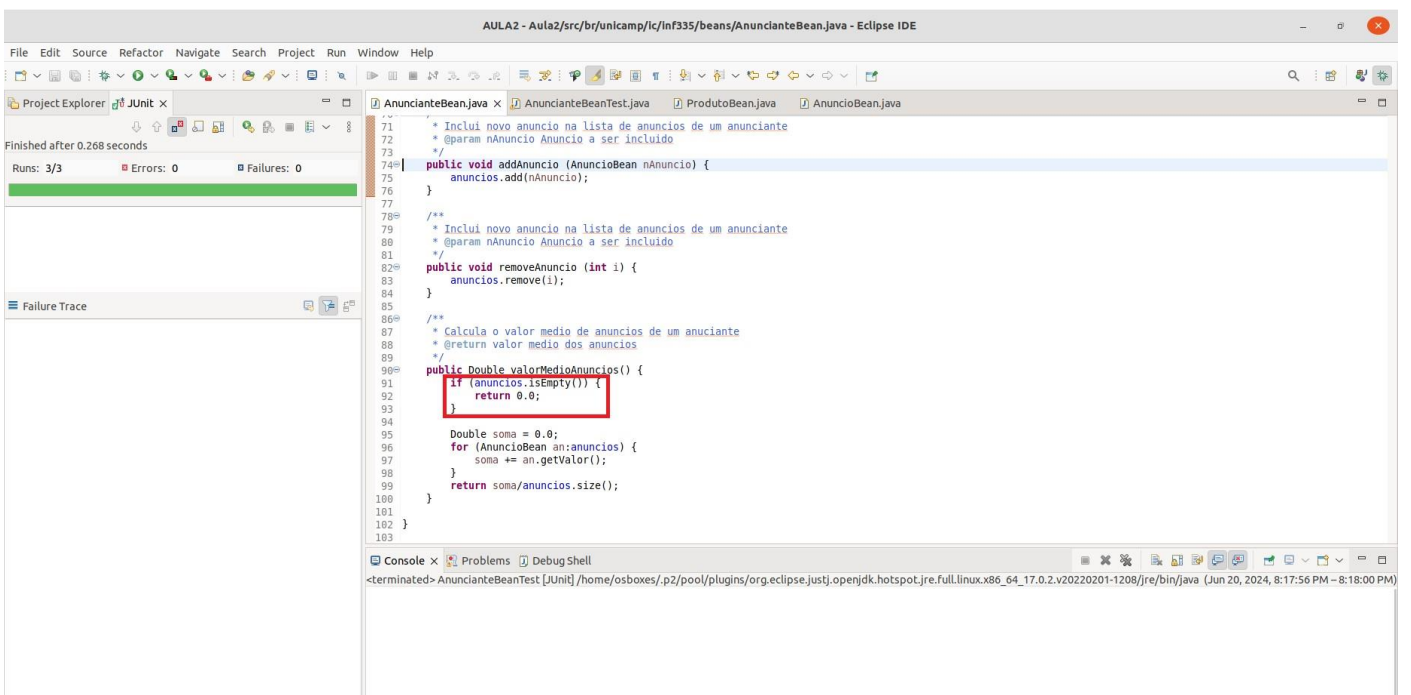
```
76 }
77
78
79 /**
80  * Contrutor com parâmetros
81  * @param produto
82  * @param fotosUrl
83  * @param desconto
84  */
85 public AnuncioBean(ProdutoBean produto, ArrayList<URL> fotosUrl, Double desconto) {
86     super();
87     this.produto = produto;
88     this.fotosUrl = fotosUrl;
89     this.desconto = desconto;
90 }
91
92 /**
93  * Calcula valor do anuncio
94  * @return retorna valor do produto menos o desconto
95  */
96
97 //MÉTODO ERRADO
98 // public Double getValor() {
99 //     return produto.getValor() - (produto.getValor() / desconto);
100 // }
101
102 //MÉTODO CORRIGIDO
103 public Double getValor() {
104     Double valorComDesconto = produto.getValor() - (produto.getValor() * desconto);
105     return valorComDesconto;
106 }
107
108
```

Após correção do método getValor() em AnuncioBean, O JUnit AnuncianteBeanTest não apresenta mais falha no assertEquals do testValorMedioAnuncios.



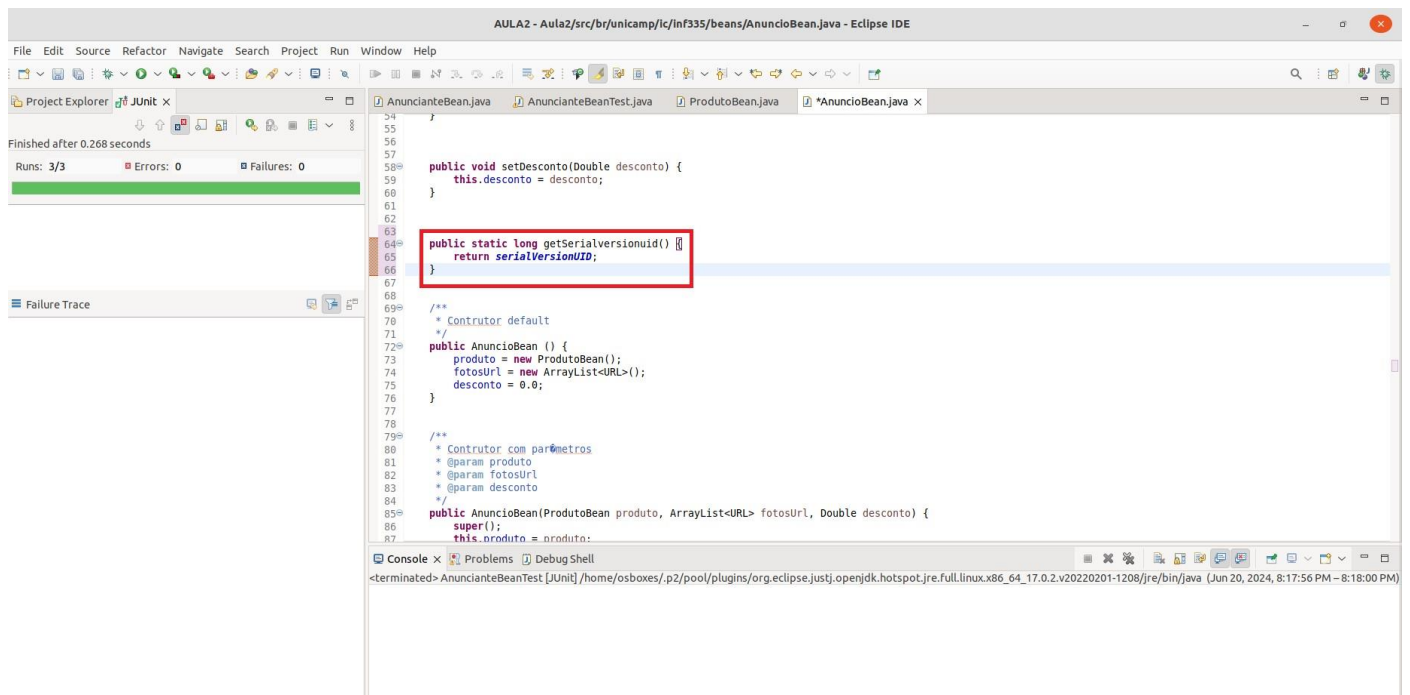
Além disso, o teste criado para a classe `AnuncioBeanTest` permite testar os métodos `testGetValor()` de forma individual. Para isso é instanciado um produto, um anúncio com esse produto e posteriormente testado o cálculo do valor do anúncio com desconto aplicado. Após correção descrita anteriormente, não apresentou erro ou falha. A função `assertEquals` compara valor entre retornado e valor esperado do método `testGetValor`.

Boas práticas de desenvolvimento de software devem ser aplicadas, por exemplo, tratar software para que seja possível ocorrer divisão por 0. A seguir é apresentada adição de `if` no método `valorMedioAnuncios` da classe `AnuncianteBean`.



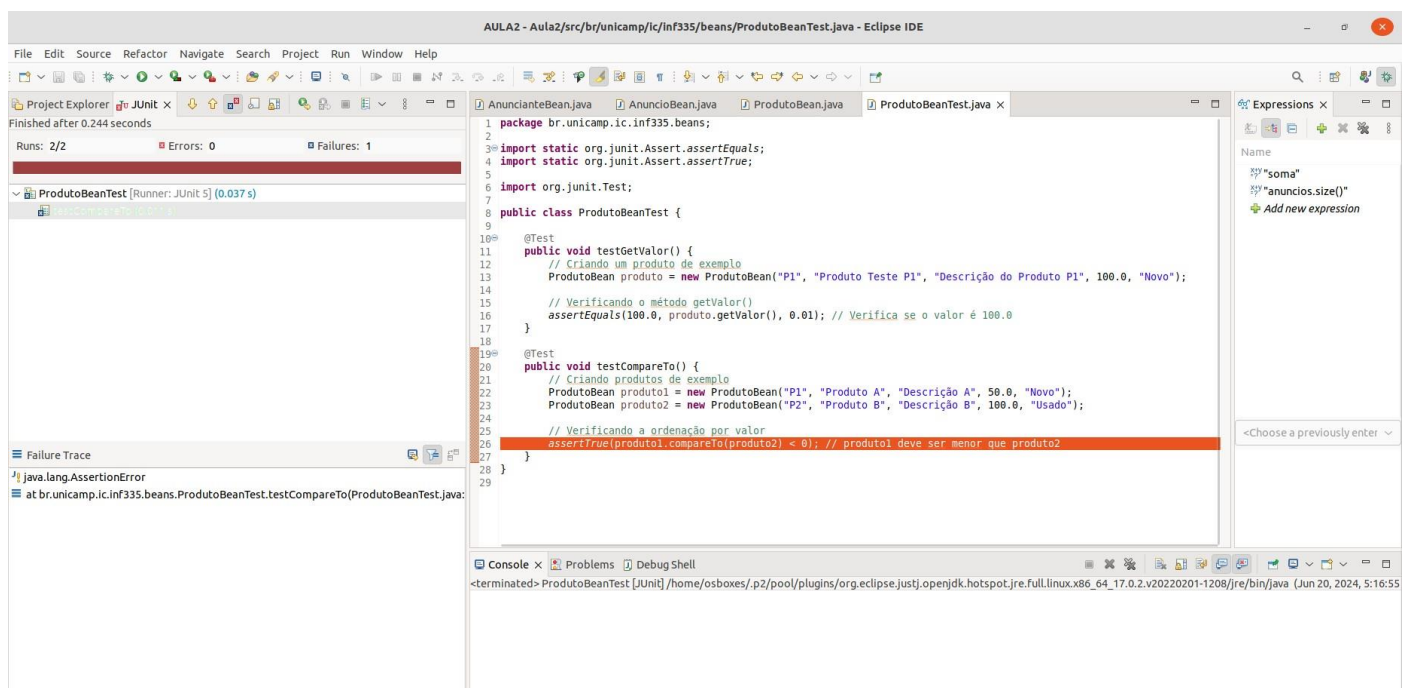


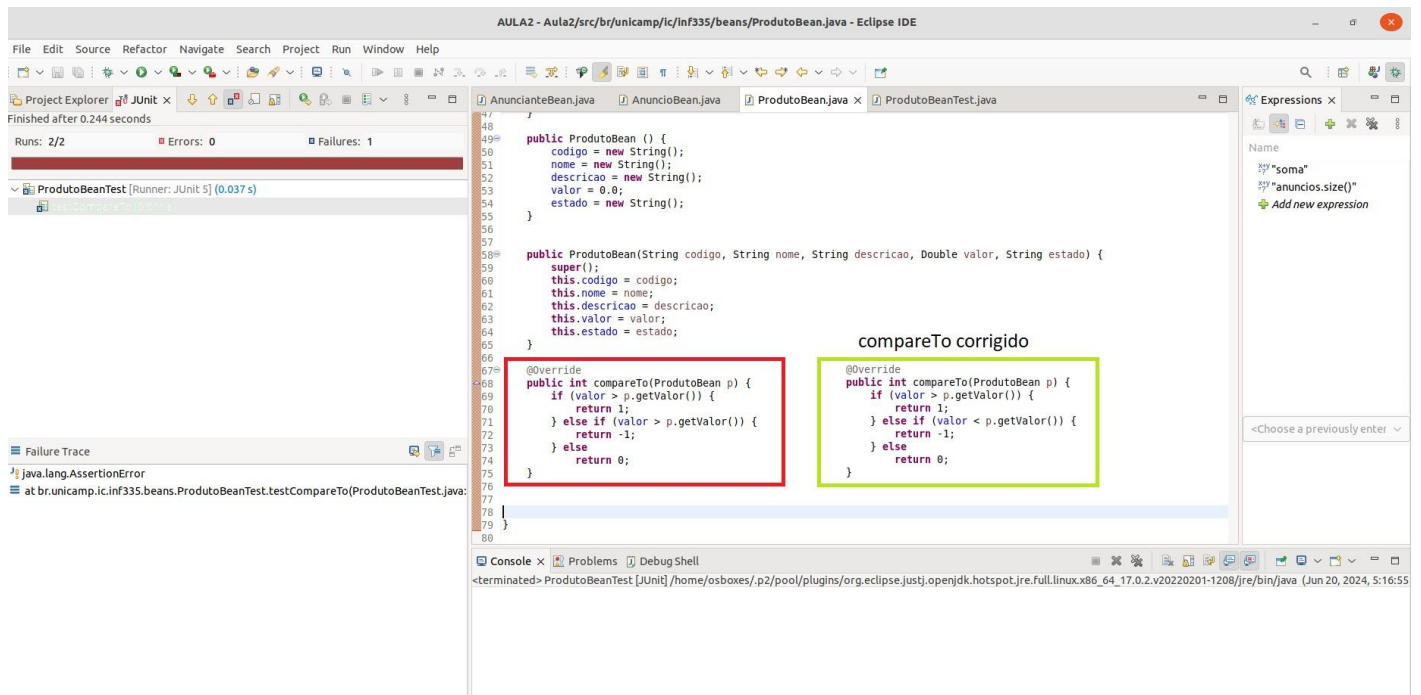
O método `getSerialVersionUID` é desnecessário e pode ser excluído sem nenhum prejuízo para o funcionamento do software.



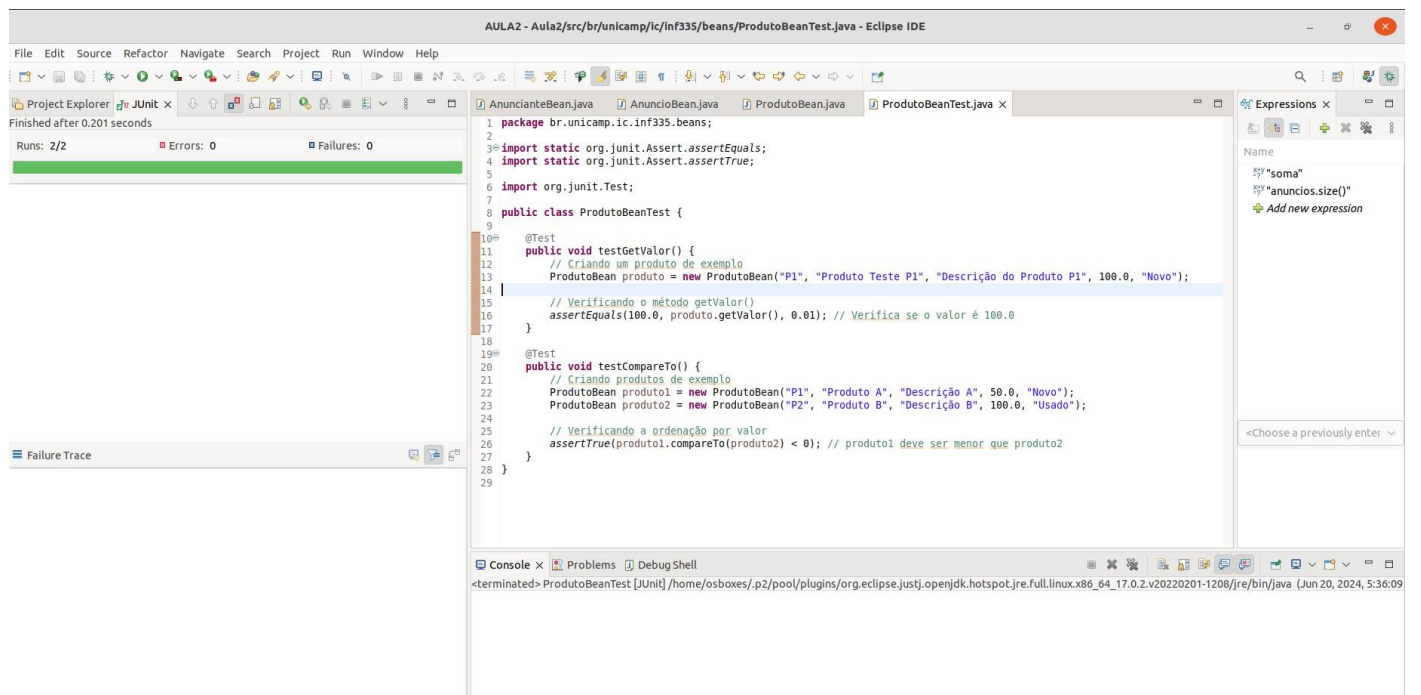
## Teste JUnit para ProdutoBeanTest:

O teste criado para a classe `ProdutoBeanTest` é para testar os métodos `testGetValor()` e `testCompareTo()`. Para isso foram instanciados objetos de exemplo. A função `assertEquals` compara valor entre retornado e valor esperado do método `testGetValor`, confirmando sucesso na inserção de valor. A função `assertTrue`, que verifica condição verdadeira, foi utilizada para verificar a ordenação por valor no método `testCompareTo`, apresentou falha e foi corrigido.





Após correção não apresentou mais falha.



O método `getSerialVersionUID` é desnecessário e pode ser excluído sem nenhum prejuízo para o funcionamento do software.



