

UNIP - Campinas

Ciência da Computação /
Sistemas de Informação

ALPOO

08 de Setembro

Interface Gráfica com o Usuário (1)

- A linguagem Java oferece, dentre as funcionalidades incorporadas à sua API padrão, um extenso conjunto de classes e interfaces para o desenvolvimento de aplicações gráficas. Esse conjunto facilita a criação de saídas na forma gráfica e de interfaces gráficas com usuários (GUIs), tanto na forma de aplicações autônomas como na forma de applets.

Interface Gráfica com o Usuário (2)

- Aplicações gráficas são criadas através da utilização de componentes gráficos, que estão agrupados em dois grandes pacotes: `java.awt` e `javax.swing`.
- AWT é o Abstract Windowing Toolkit, formador por que classes agrupam as funcionalidades gráficas que estão presentes desde a primeira versão de Java, que operam tendo por base as funcionalidades de alguma biblioteca gráfica do sistema onde a aplicação é executada.

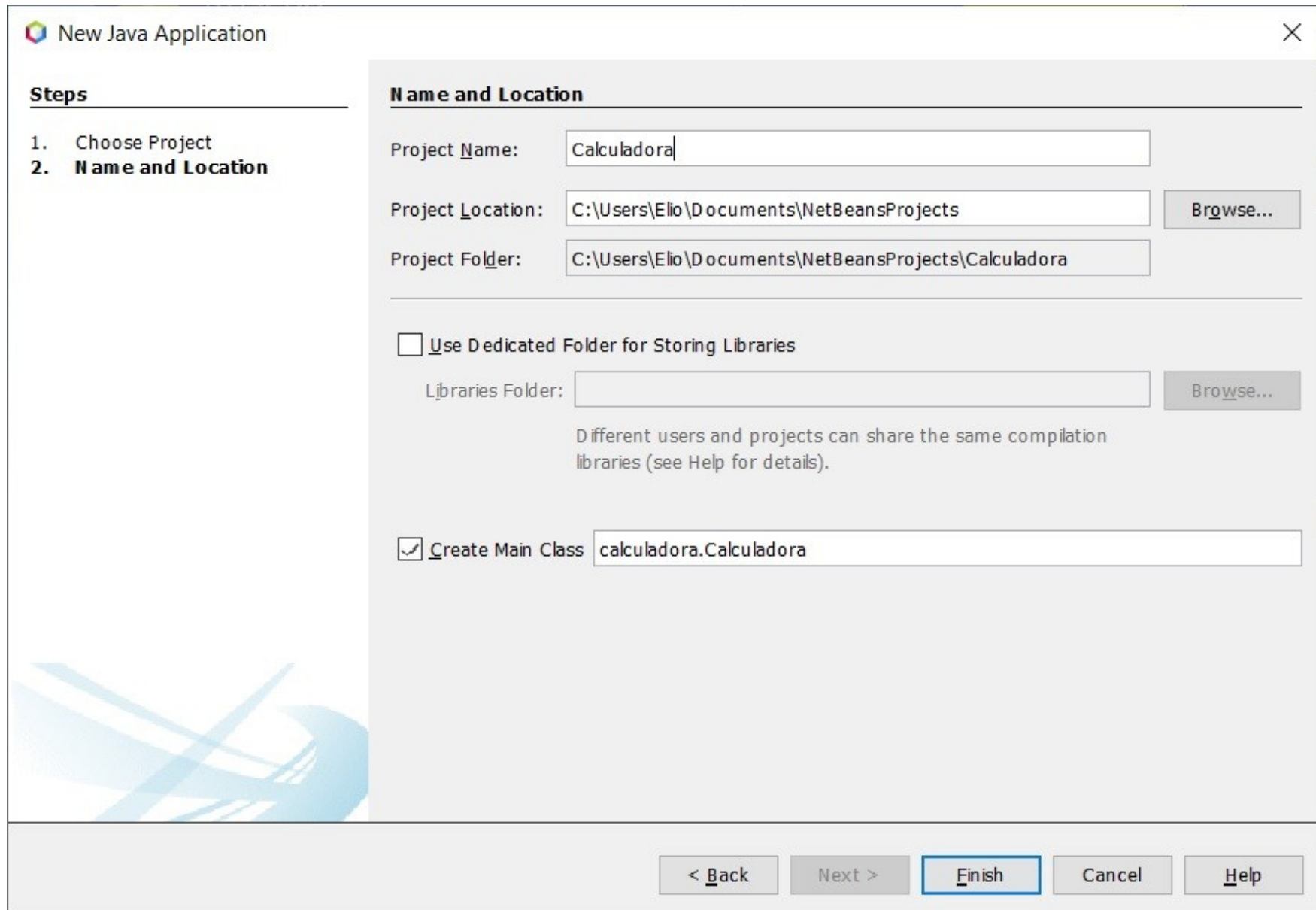
Interface Gráfica com o Usuário (3)

- Já o pacote `javax.swing` define uma extensão padronizada a partir de AWT que congrega componentes gráficos que utilizam exclusivamente Java (lightweight components), com funcionalidades e aparência independentes do sistema onde a aplicação é executada.

Exemplo (1)

- Nesta aula iremos desenvolver um exemplo para apresentar diferentes componentes de interação. Para tanto, crie um novo projeto chamado Calculadora.

Exemplo (2)



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

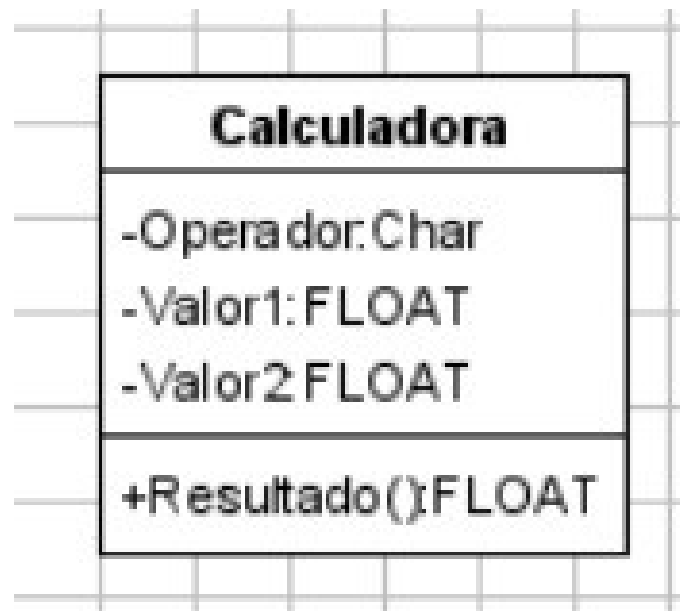
Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

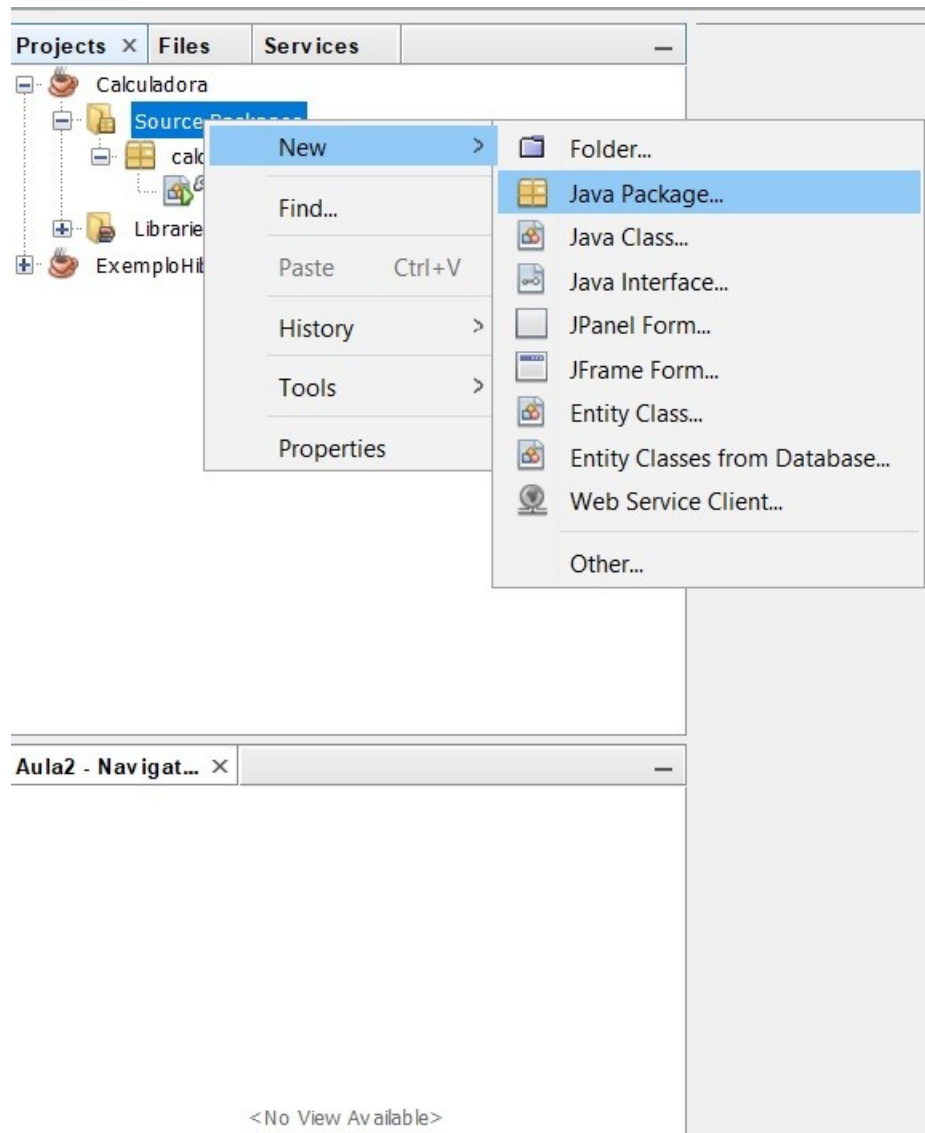
< Back Next > **Finish** Cancel Help

Exemplo (3)

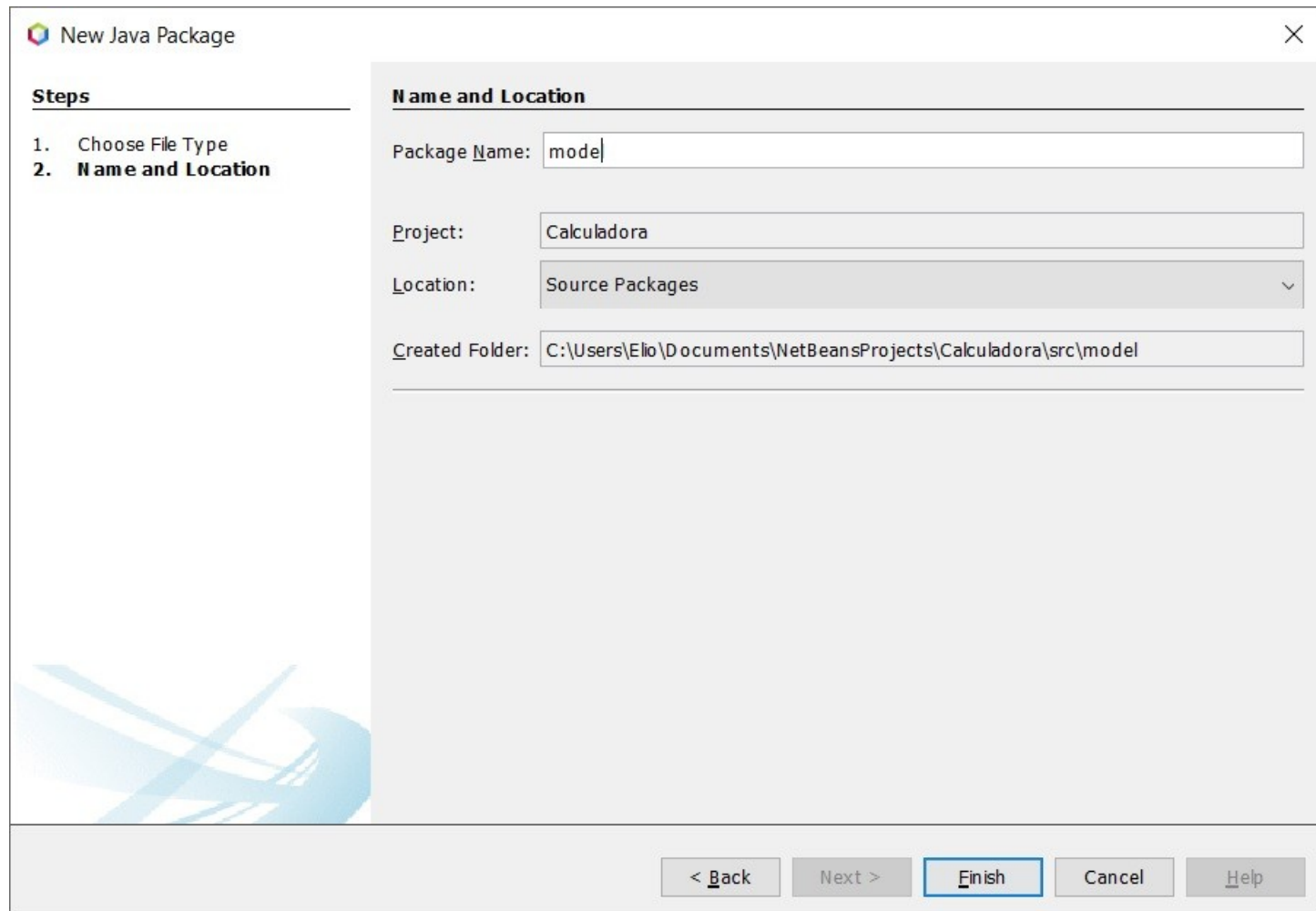
- No pacote do **model**, crie uma nova classe chamada Calculadora (Lembre-se que o Java diferencia maiúsculas de minúsculas). Os atributos e métodos da classe são apresentados abaixo.



Exemplo (4)



Exemplo (5)



The image shows a 'New Java Package' dialog box with a sidebar on the left and a main area on the right. The sidebar contains a 'Steps' section with two items: '1. Choose File Type' and '2. Name and Location', where the second item is currently selected. The main area is titled 'Name and Location' and contains four input fields: 'Package Name' with the text 'model', 'Project' with 'Calculadora', 'Location' with a dropdown menu showing 'Source Packages', and 'Created Folder' with the path 'C:\Users\Elio\Documents\NetBeansProjects\Calculadora\src\model'. At the bottom of the dialog are five buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), 'Cancel', and 'Help'.

New Java Package

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Package Name: model

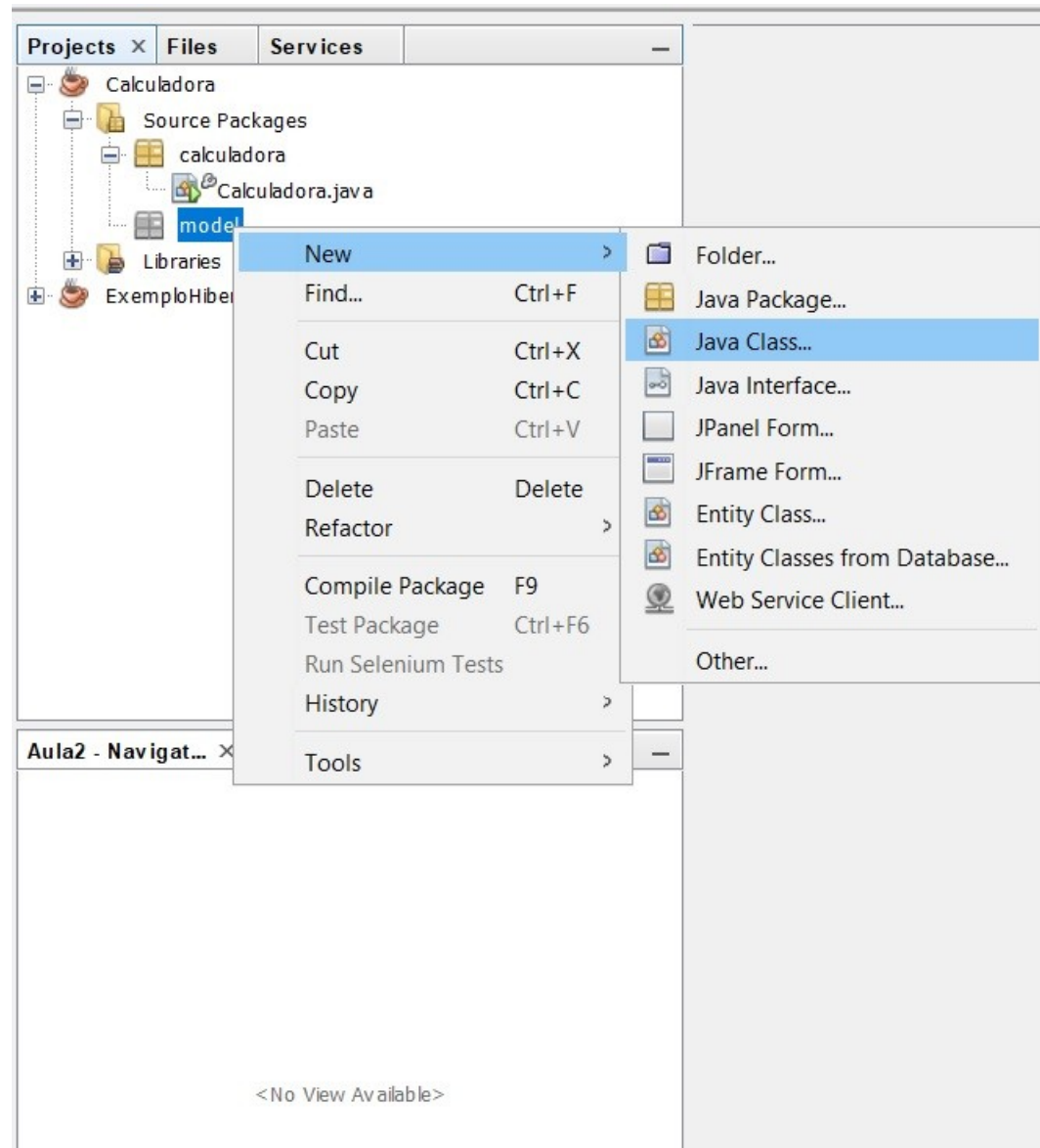
Project: Calculadora

Location: Source Packages

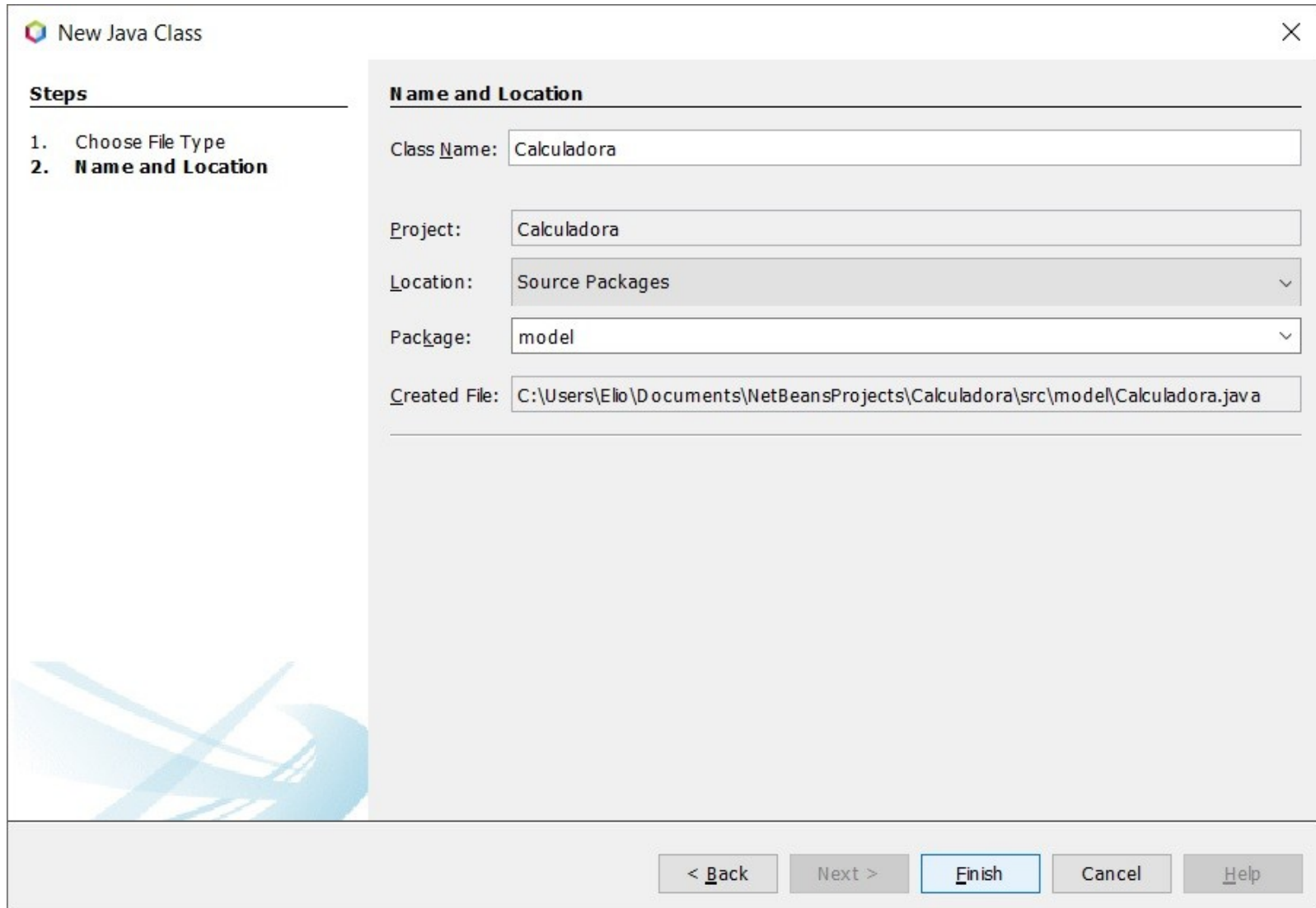
Created Folder: C:\Users\Elio\Documents\NetBeansProjects\Calculadora\src\model

< Back Next > Finish Cancel Help

Exemplo (6)



Exemplo (7)



The image shows a 'New Java Class' dialog box from an IDE. It has a title bar with a close button. On the left, a 'Steps' pane shows two steps: '1. Choose File Type' and '2. Name and Location', with the second step being active. The main area is titled 'Name and Location' and contains several input fields: 'Class Name' (text box with 'Calculadora'), 'Project' (text box with 'Calculadora'), 'Location' (dropdown menu with 'Source Packages'), 'Package' (dropdown menu with 'model'), and 'Created File' (text box showing the full file path). At the bottom, there are five buttons: '< Back', 'Next >', 'Finish' (highlighted in blue), 'Cancel', and 'Help'.

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

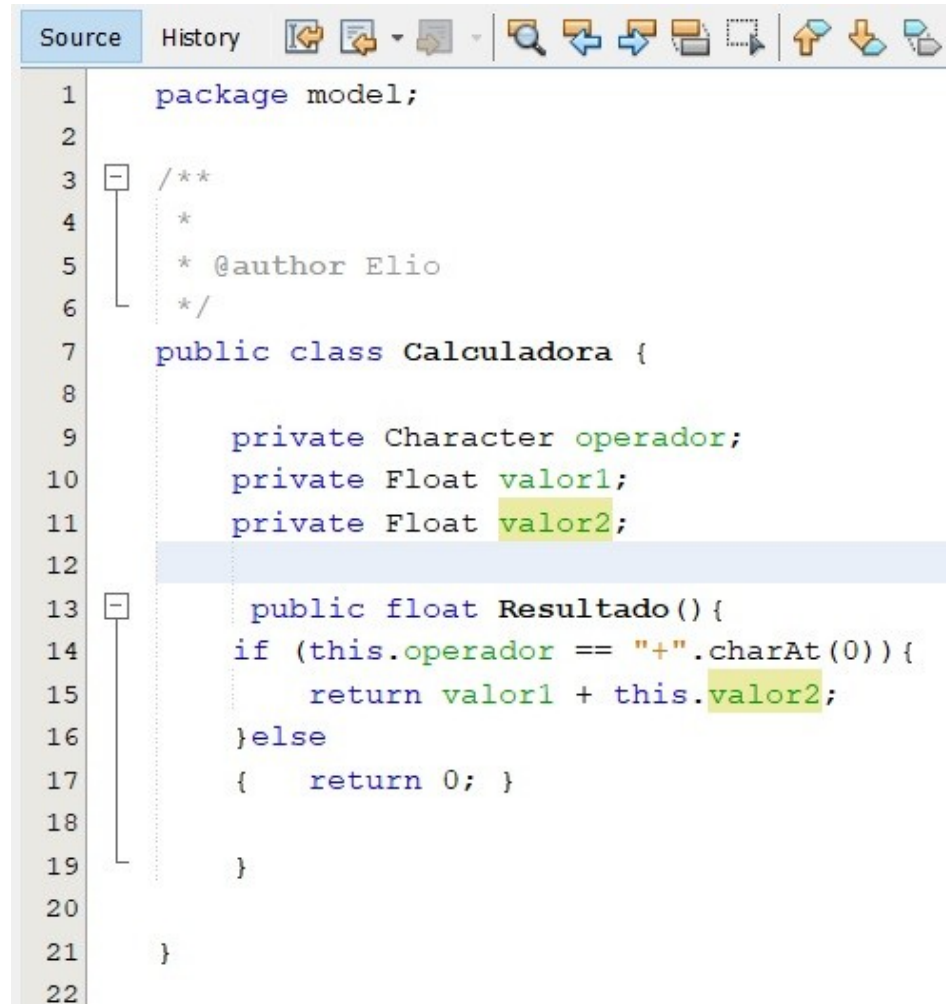
Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

Exemplo (8)



The screenshot shows an IDE window with a toolbar at the top containing icons for Source, History, and various editing actions. The code is written in Java and is as follows:

```
1 package model;
2
3 /**
4  *
5  * @author Elio
6  */
7 public class Calculadora {
8
9     private Character operador;
10    private Float valor1;
11    private Float valor2;
12
13    public float Resultado() {
14        if (this.operador == "+".charAt(0)) {
15            return valor1 + this.valor2;
16        } else {
17            return 0; }
18    }
19 }
20
21 }
22
```

The code defines a package `model` and a public class `Calculadora`. It includes a Javadoc comment, private fields `operador` (Character), `valor1` (Float), and `valor2` (Float), and a public method `Resultado()` that returns a float. The method checks if the operator is '+' and returns the sum of `valor1` and `valor2`, otherwise it returns 0. The IDE interface includes a toolbar with icons for Source, History, and various editing actions.

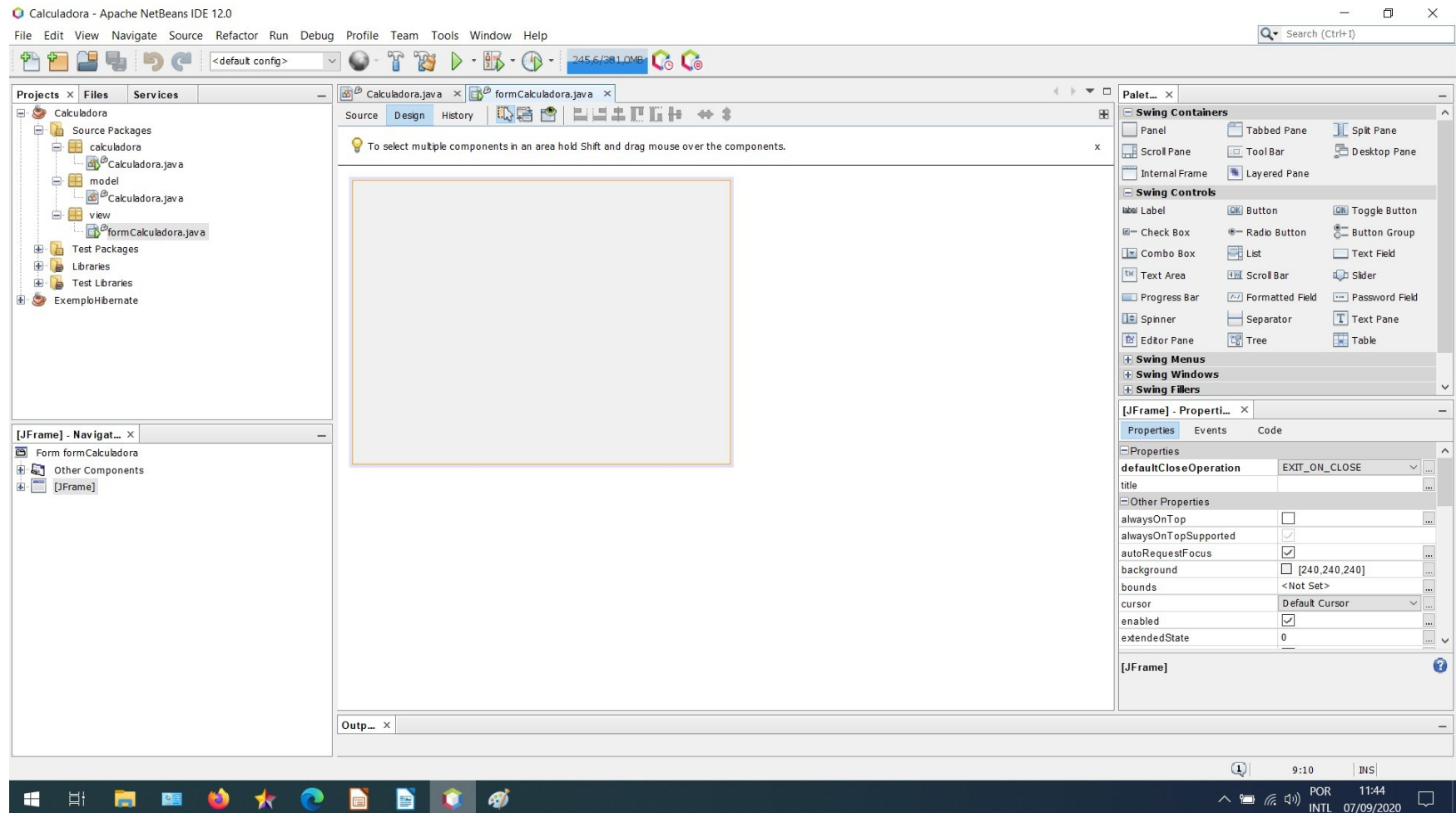
- Crie o construtor e os métodos de acesso...

Exemplo (9)

- O NetBeans possui funcionalidades para auxiliar no desenvolvimento da interface com os usuários, permitindo o emprego dos componentes visuais da linguagem Java mais facilmente.
- Agora, crie o pacote view e nele novo JFrameFrom, chamado formCalculadora.

Exemplo (10)

- Após finalizar a criação, o sistema irá exibir a página para edição, conforme a figura abaixo.



Exemplo (11)

- Observe os componentes: Source, Navigation e Properties
- Acrescente 3 Label's, 1 Button e 2 Text Field's (vazios – Propriedade Text) no formulário, criando uma tela similar à próxima figura.



Valor 1:

Valor 2:

Resultado:

Exemplo (12)

- Mude para a aba de código, e crie um objeto da classe Calculadora dentro do formulário. Será necessário importar do pacote model....

```
package view;
```

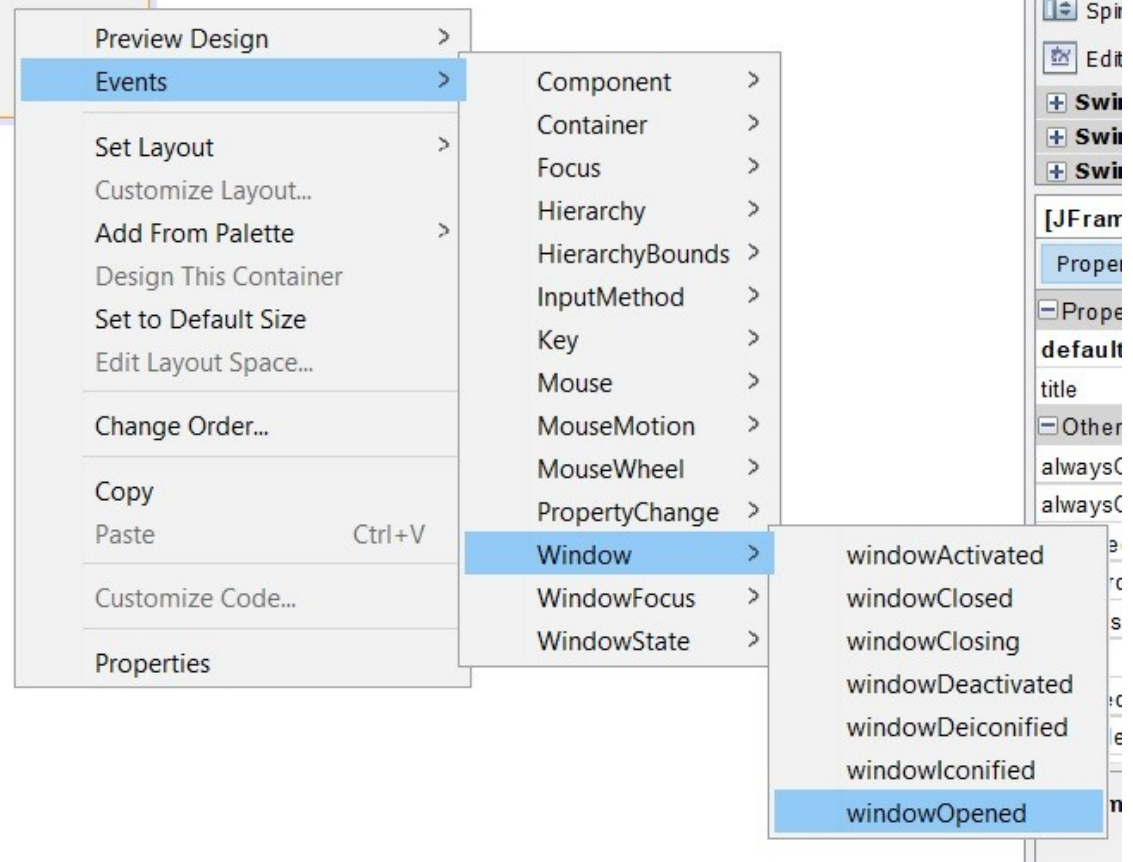
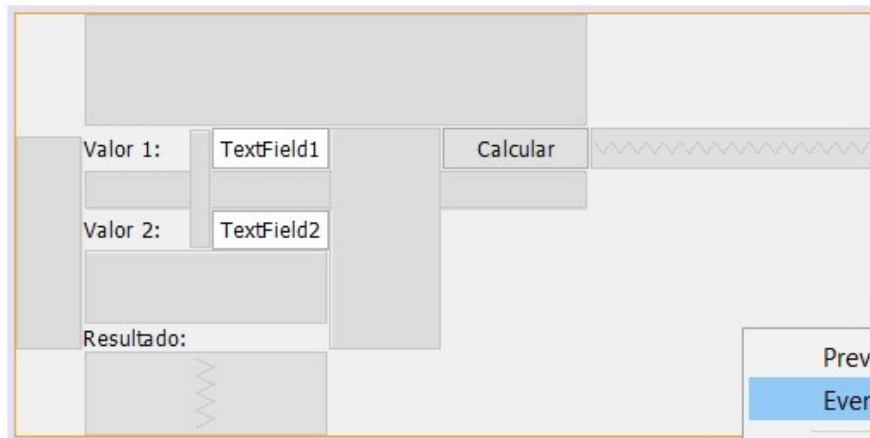
```
import model.Calculadora;
```

```
/**
 *
 * @author Elio
 */
public class formCalculadora extends javax.swing.JFrame {
    Calculadora calculadora;
    ...
}
```


Exemplo (13)

- Observe que o objeto ainda não foi instanciado. Para isso, clique com o botão direito no formulário e selecione conforme a próxima figura.

Exemplo (14)



Exemplo (15)

- Evento é uma ação que será executado quando uma atividade esperada ocorrer.
- No caso windowopened, ou seja, quando a janela for aberta, criaremos o objeto.

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {  
    calculadora = new Calculadora();  
}
```

Exemplo (16)

- Uma vez criando o objeto devemos usar o método Resultado().
- Para tanto, iremos realizar o cálculo quando o usuário clicar no botão, ou seja evento **mouseclicked** do botão.
- Implemente o método conforme a figura a seguir.

Exemplo (17)

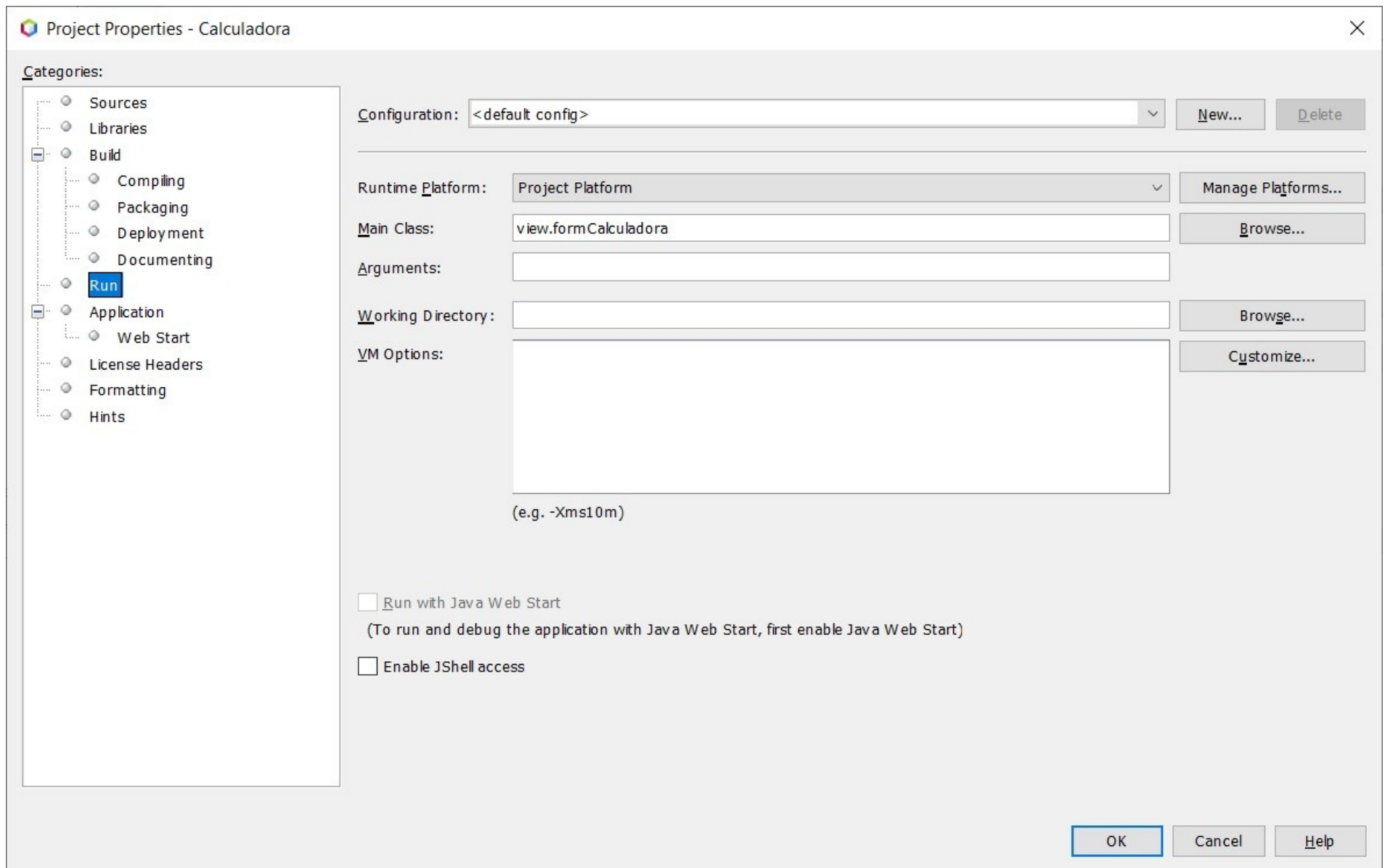
```
private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {  
    float resultado;  
    calculadora.setOperador("+".charAt(0));  
    calculadora.setValor1(Float.parseFloat(jTextField1.getText()));  
    calculadora.setValor2(Float.parseFloat(jTextField2.getText()));  
    resultado = calculadora.Resultado();  
    jLabel13.setText("Resultado:" + resultado);  
}
```

- Repare que o formCalculadora não será executado junto com o projeto, pois ele não é o main principal.

Exemplo (18)

Para alterar isso, selecione o projeto com o botão direito e escolha a opção properties. Na opção Run, mude a classe principal, conforme a figura seguir.

Exemplo (19)



Exercícios (1)

1) Implemente as 4 operações básicas na calculadora.

2) Faça as seguintes atividades:

(a) Criar a classe Produto (pacote model) com os atributos privados: nome (String), código (Integer), valor (Float) e preço (Float) e um construtor.

(b) Criar um JFrame (pacote view) para preencher os campos nome, código e valor de um objeto da classe produto.

(c) Calcular o preço do produto, sabendo-se que o preço é $1.2 * \text{valor}$.

(d) Criar um botão no JFrame para exibir os dados do produto utilizando showMessageDialog.

Exercícios (2)

3) Faça as seguintes atividades:

- (a) Criar a classe Perecível (model), que herda de Produto e possui um atributo adicional validade (Integer) e ainda um construtor próprio. Se necessário, modifique a classe produto.
- (b) Alterar o JFrame para permitir o preenchimento dos campos nome, código, valor e validade de um objeto da classe perecível.
- (c) Fazer a sobrecarga do método para calcular o preço, sabendo-se que o preço é $1.4 * \text{valor}$ do produto perecível.
- (d) Exibir os dados do perecível utilizando showMessageDialog.