



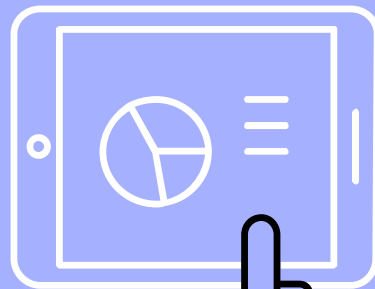
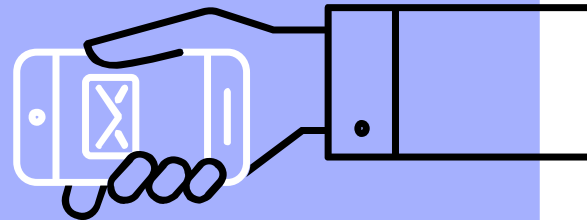
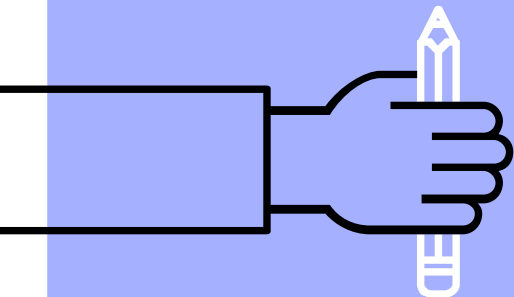
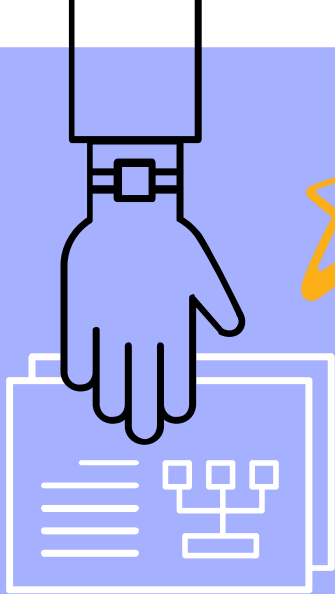
Universidade

Cruzeiro do Sul

Programação Orientada a Objetos

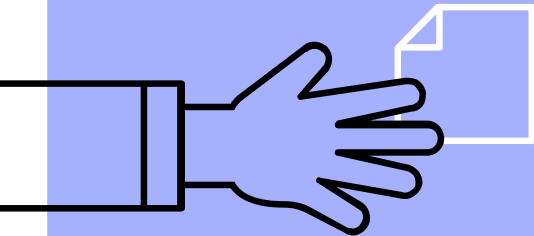
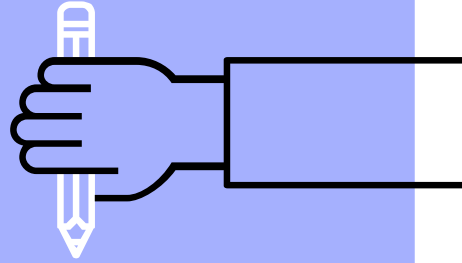
Aula 08

Prof.^a Erika Miranda



8. Unidade

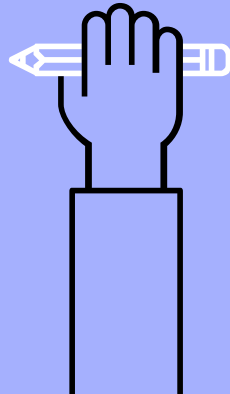
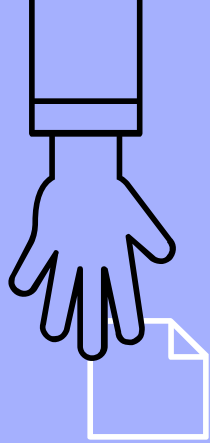
Herança



Herança

Como o próprio nome sugere, na orientação a objetos o termo herança **se refere a algo herdado**.

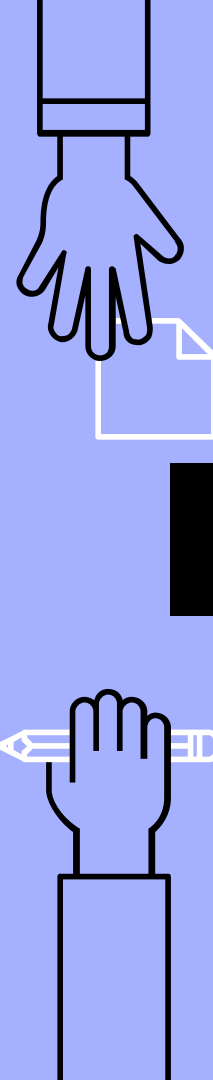
Em Java, a herança ocorre quando uma **classe passa a herdar características** (atributos e métodos) **definidas em uma outra classe**, especificada como sendo sua ancestral ou superclasse.



Herança

A técnica da herança **possibilita** o **compartilhamento** ou **reaproveitamento** de recursos definidos anteriormente em uma outra classe.

A classe **fornecedora** dos recursos recebe o nome de **superclasse** e a **receptora** dos recursos de **subclasse**.



Herança

Uma classe derivada **herda a estrutura de atributos** e métodos de sua classe “base”, mas pode seletivamente:

- adicionar novos métodos
- estender a estrutura de dados
- redefinir a implementação de métodos já existentes

► Exemplo:

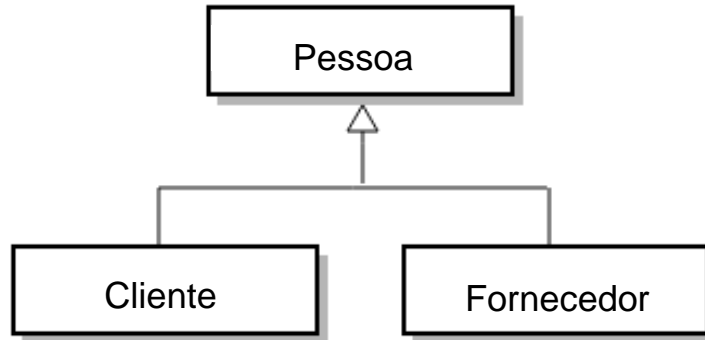
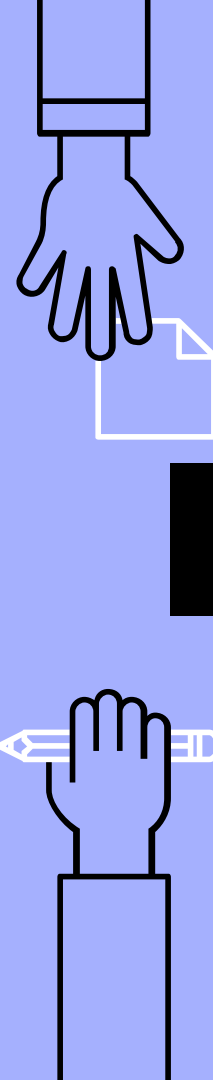
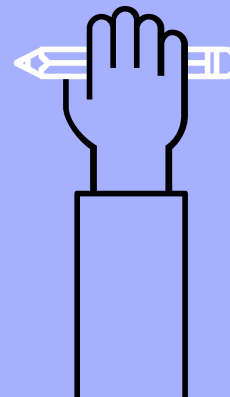
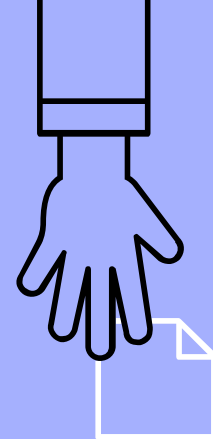
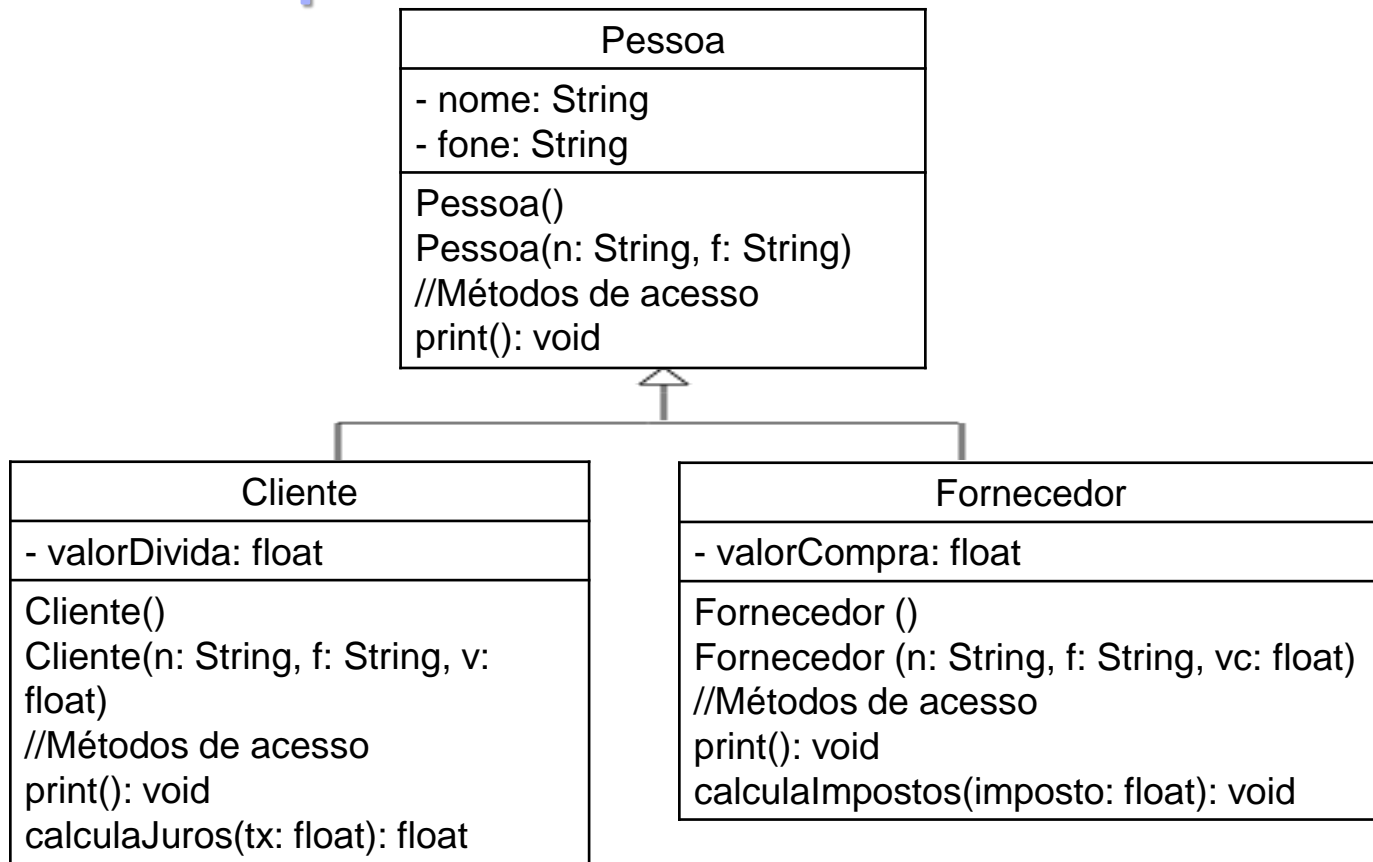


Diagrama UML simplificado (não mostra os métodos e atributos)



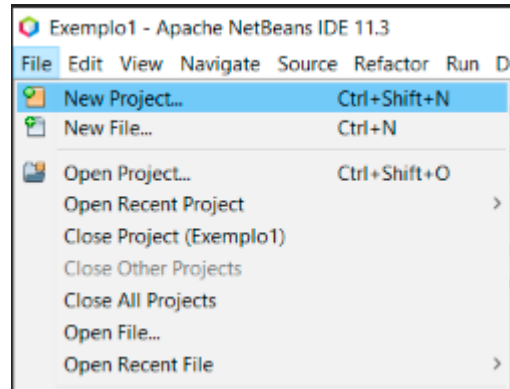
Exemplo



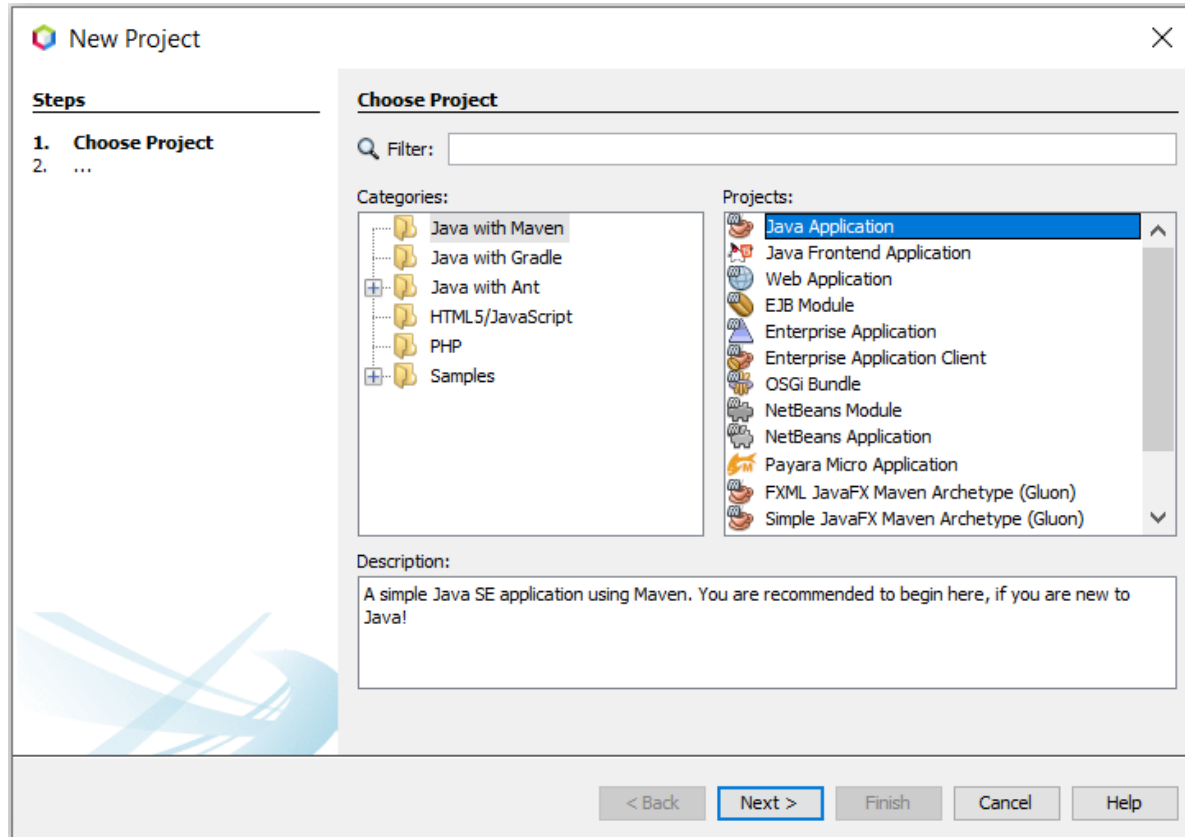
Exemplos

Vamos implementar usando o NetBeans.

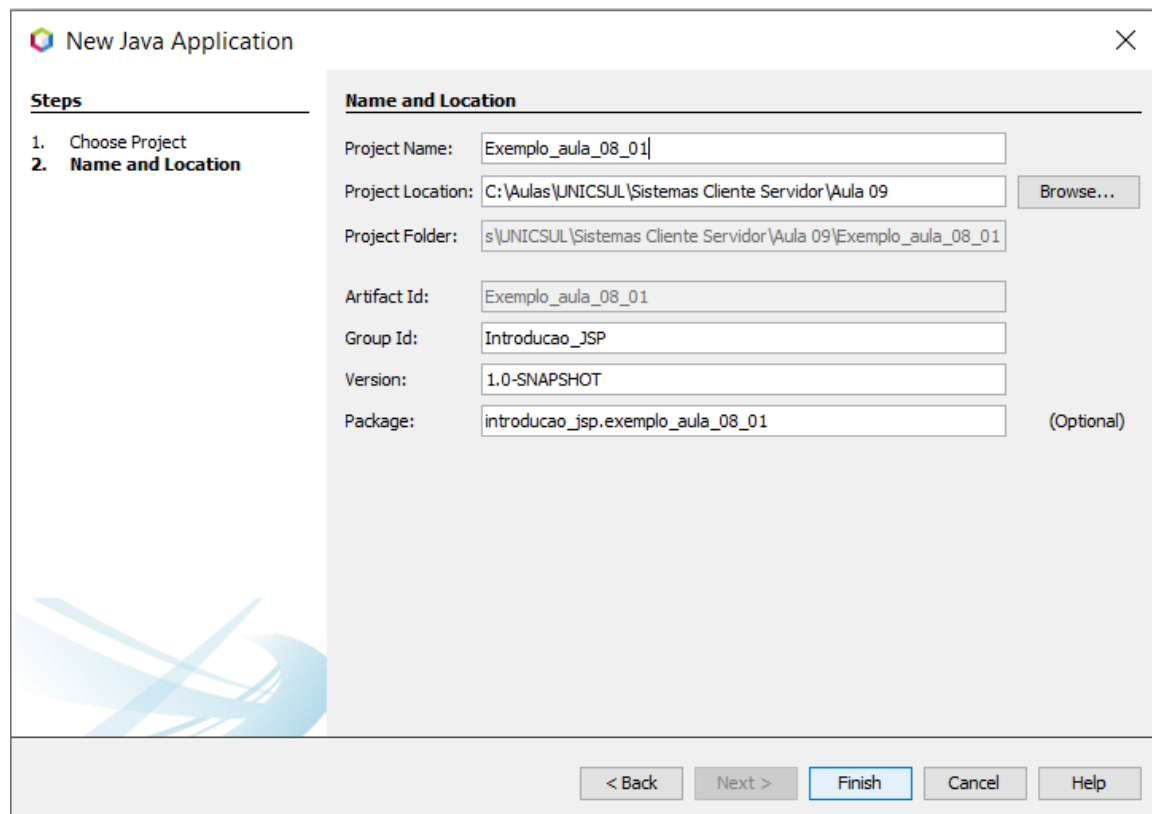
Vamos iniciar um novo projeto:



Exemplos



Exemplos



New Java Application

Steps

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

Name and Location

Project Name:

Project Location:

Project Folder:

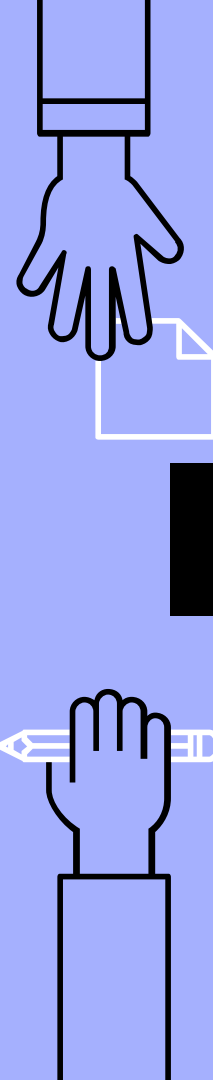
Artifact Id:

Group Id:

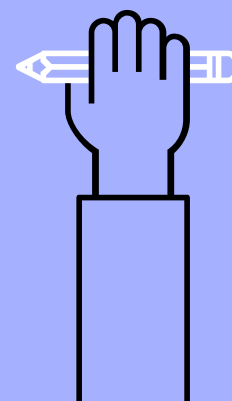
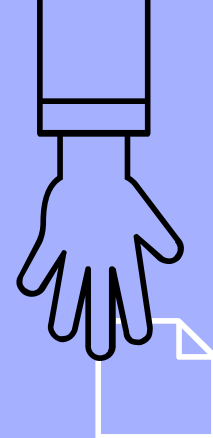
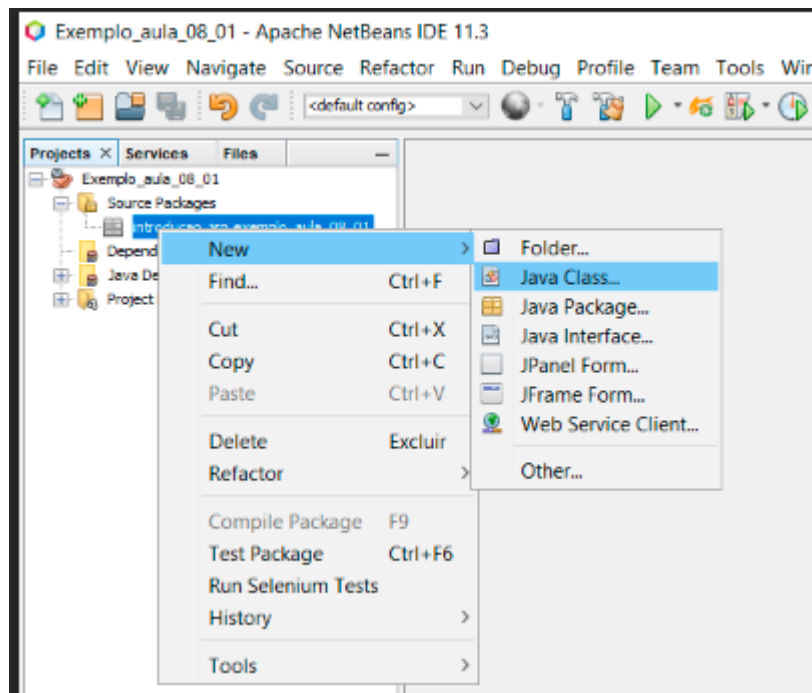
Version:

Package: (Optional)

< Back Next > **Finish** Cancel Help



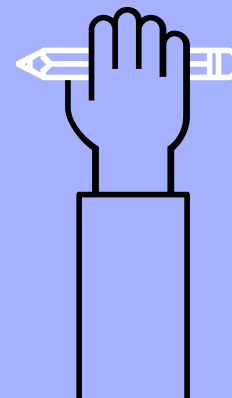
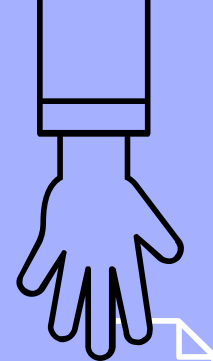
Interfaces - Exemplos



Interfaces - Exemplos

Vamos criar as seguintes classes:

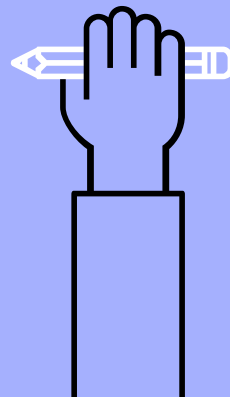
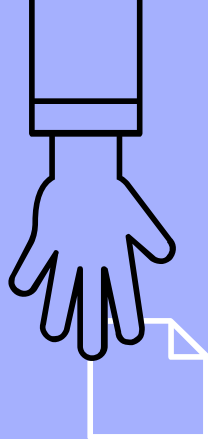
- ▶ Pessoa
- ▶ Fornecedor (extends Pessoa)
- ▶ Cliente (extends Pessoa)
- ▶ TestePessoa (public static void main(String[] args))



Pessoa.java x Fornecedor.java x TestePessoa.java x Cliente.java x

Source History

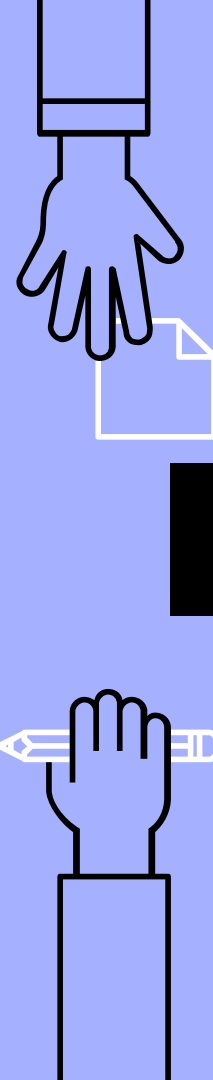
```
1  import javax.swing.JOptionPane;
2
3  @
4  public class Pessoa {
5      private String nome;
6      private String fone;
7
8      Pessoa() {}
9      Pessoa(String n, String f){
10         nome=n;
11         fone=f;
12     }
13
14     public String getNome() {
15         return nome;
16     }
17
18     public void setNome(String nome) {
19         this.nome = nome;
20     }
21
22     public String getFone() {
23         return fone;
24     }
25
26     public void setFone(String fone) {
27         this.fone = fone;
28     }
29
30     @
31     public void print(){
32         JOptionPane.showMessageDialog(null,"Dados \nNome: "
33                                     + nome + "\nTelefone: "+ fone);
34     }
35 }
```



Pessoa.java x Fornecedor.java x TestePessoa.java x Cliente.java x

Source History

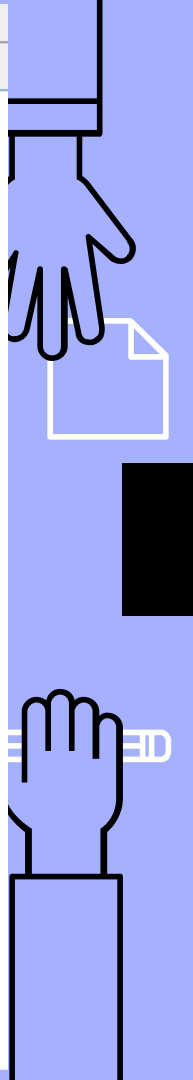
```
1  import javax.swing.JOptionPane;
2
3  public class Fornecedor extends Pessoa{
4      private float valorCompra;
5      Fornecedor() {}
6      Fornecedor(String n, String f, float vc){
7          super(n, f);
8          valorCompra=vc;
9      }
10     public float getValorCompra() {
11         return valorCompra;
12     }
13     public void setValorCompra(float valorCompra) {
14         this.valorCompra = valorCompra;
15     }
16     public void print() {
17         super.print();
18         JOptionPane.showMessageDialog(null, "\nValor da Compra: " + valorCompra);
19     }
20     public void calculaImpostos(float imposto){
21         valorCompra+=valorCompra*imposto/100;
22     }
23 }
```

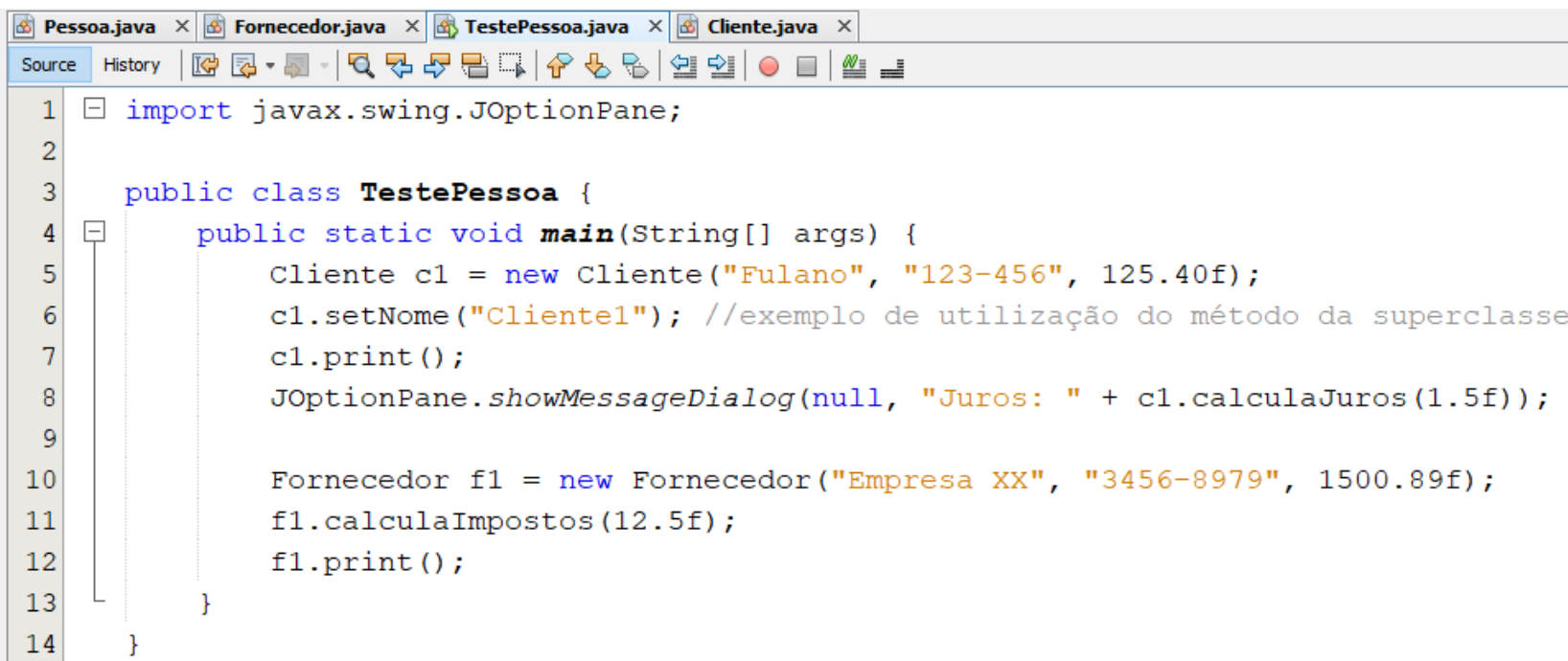


```

1  import javax.swing.JOptionPane;
2  public class Cliente extends Pessoa {
3      private float valorDivida;
4
5      Cliente() {}
6
7      Cliente(String n, String f, float vd) {
8          super(n, f);
9          valorDivida = vd;
10     }
11
12     public float getValorDivida() {
13         return valorDivida;
14     }
15
16     public void setValorDivida(float valorDivida) {
17         this.valorDivida = valorDivida;
18     }
19
20     public void print() {
21         super.print();
22         JOptionPane.showMessageDialog(null, "\nValor da Divida: " + valorDivida);
23     }
24
25     public float calculaJuros(float tx) {
26         return valorDivida * tx / 100;
27     }
28 }

```





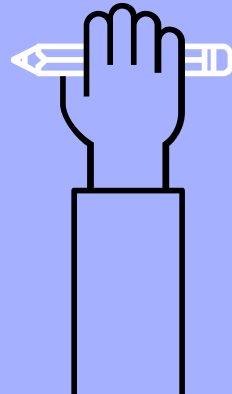
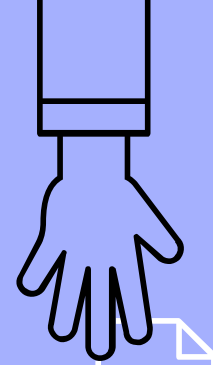
The image shows a screenshot of a Java IDE with four tabs: Pessoa.java, Fornecedor.java, TestePessoa.java, and Cliente.java. The TestePessoa.java tab is active, displaying the following code:

```
1 import javax.swing.JOptionPane;
2
3 public class TestePessoa {
4     public static void main(String[] args) {
5         Cliente c1 = new Cliente("Fulano", "123-456", 125.40f);
6         c1.setNome("Cliente1"); //exemplo de utilização do método da superclasse
7         c1.print();
8         JOptionPane.showMessageDialog(null, "Juros: " + c1.calculaJuros(1.5f));
9
10        Fornecedor f1 = new Fornecedor("Empresa XX", "3456-8979", 1500.89f);
11        f1.calculaImpostos(12.5f);
12        f1.print();
13    }
14 }
```

On the right side of the image, there is a vertical blue bar with two hand icons. The top hand is a white outline of a hand with fingers spread, holding a white document icon. The bottom hand is a white outline of a hand holding a white pencil icon.

Interfaces - Exemplos

Vamos testar a aplicação!



Exercício – Completar o Diagrama com os atributos e pelo menos um métodos em cada Classe e Implementar o diagrama de classe utilizando o conceito de Herança

