TECNOLOGIA EM DESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA

TÉCNICA DE PROGRAMAÇÃO I

Aula 01

Técnica de Programação

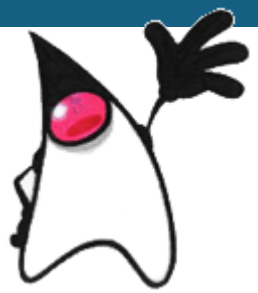
Profº Luiz Cláudio

Fatec Registro

CPS Centro Paula Souza

SÃO PAULO GOVERNO DO ESTADO

Java – Orientação a objetos



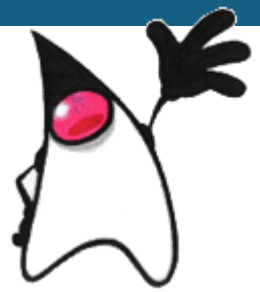
- É uma técnica de desenvolvimento de softwares que consiste em representar os elementos do mundo real (que pertencem ao escopo da aplicação) dentro do software.

- Em tese, é uma forma mais natural de informatização, já que leva em consideração os elementos como eles realmente existem.

Java –Conceitos de Orientação a Objetos

- Vale ressaltar que a orientação a objetos não é exclusividade da linguagem Java. Outras linguagem C#, VB.net, PHP , Python entre outras, fazem uso de tais recursos.

Orientação a Objetos
Abstração



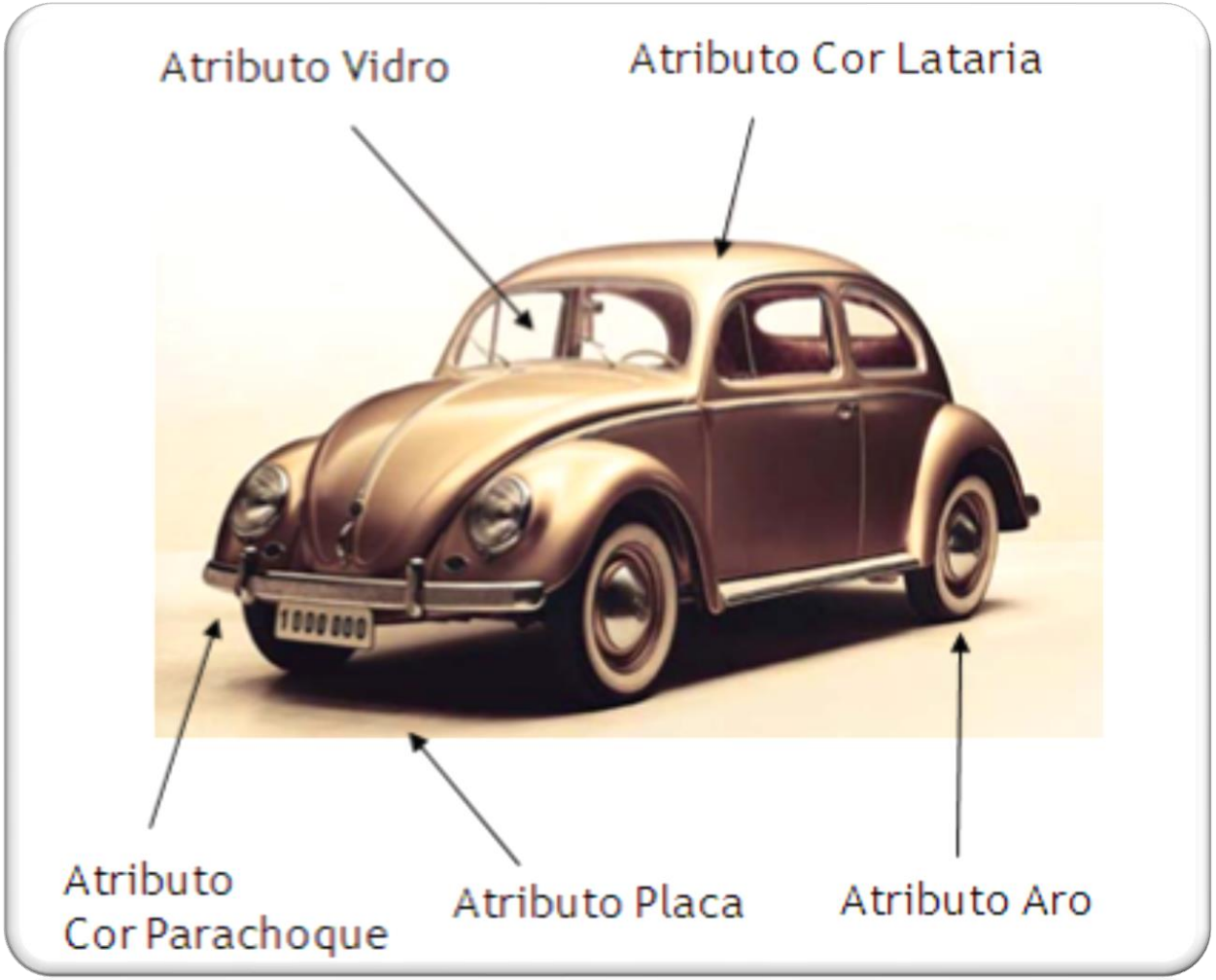
Os carros apresentarem diferentes formatos, cores e estilos.

Cada objeto-carro tem característica semelhantes entre si, por exemplo todos tem quatro rodas, e no mínimo, duas portas, além de luzes de farol e de freio, bem como vidros frontais e laterais, porém cada carro possui sua característica própria.

No momento compreendemos este conceito, percebemos que “fusca” e a denominação de um grupo, ou seja, é a abstração de uma classe: a classe carro.

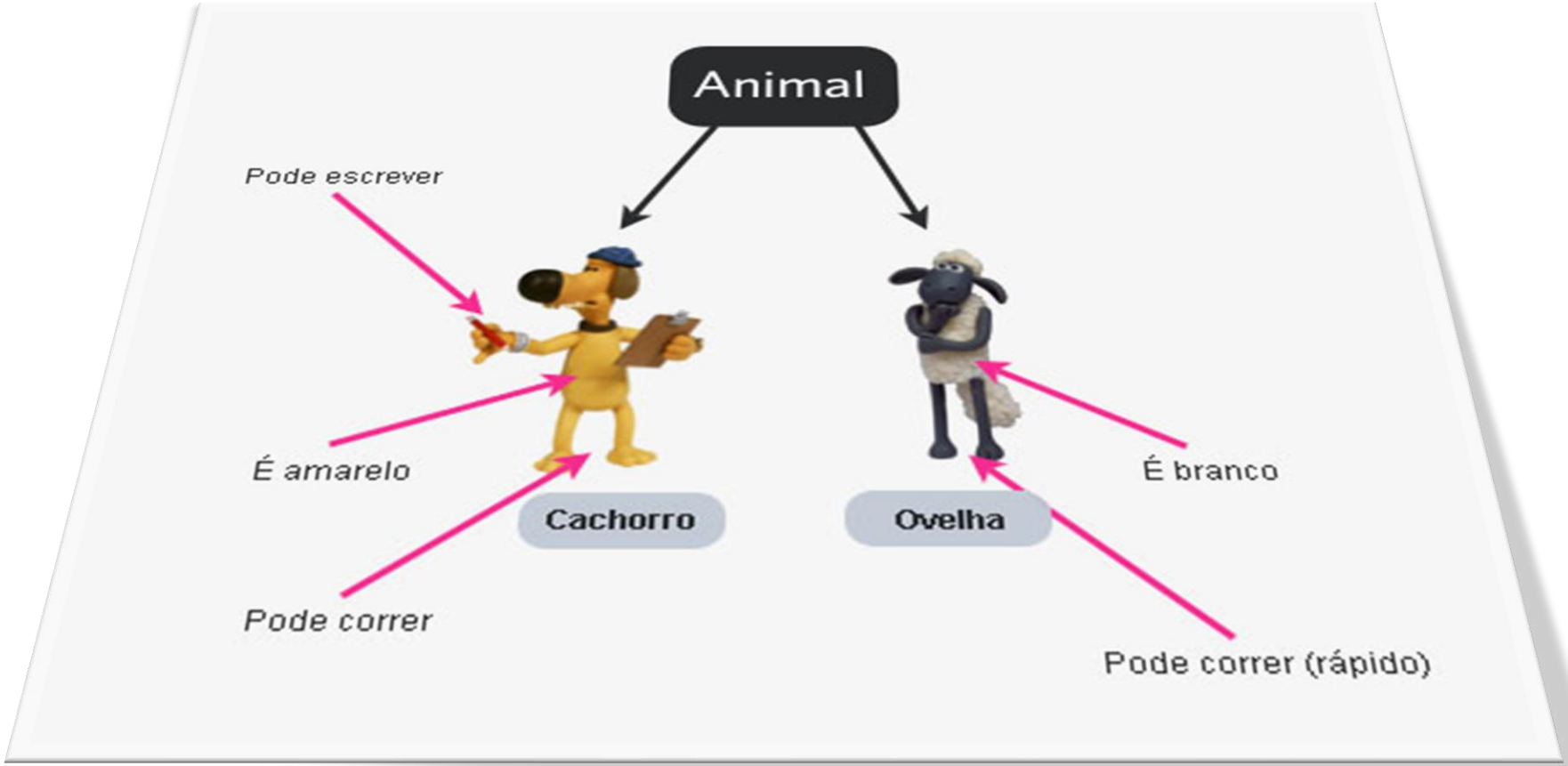
Orientação a Objetos

Abstração



Orientação a Objetos

Abstração



Orientação a Objetos

Classe

- Na verdade, na linguagem Java, **tudo** esta definido em classes.
- A estrutura da própria linguagem é organizada em classes.
- Representações básicas de classes em orientação a objetos:
  - A classe de modelagem (ou definição de tipo).
  - A classe que contém a inicialização da aplicação.

Orientação Objetos

Ideias básicas da OO

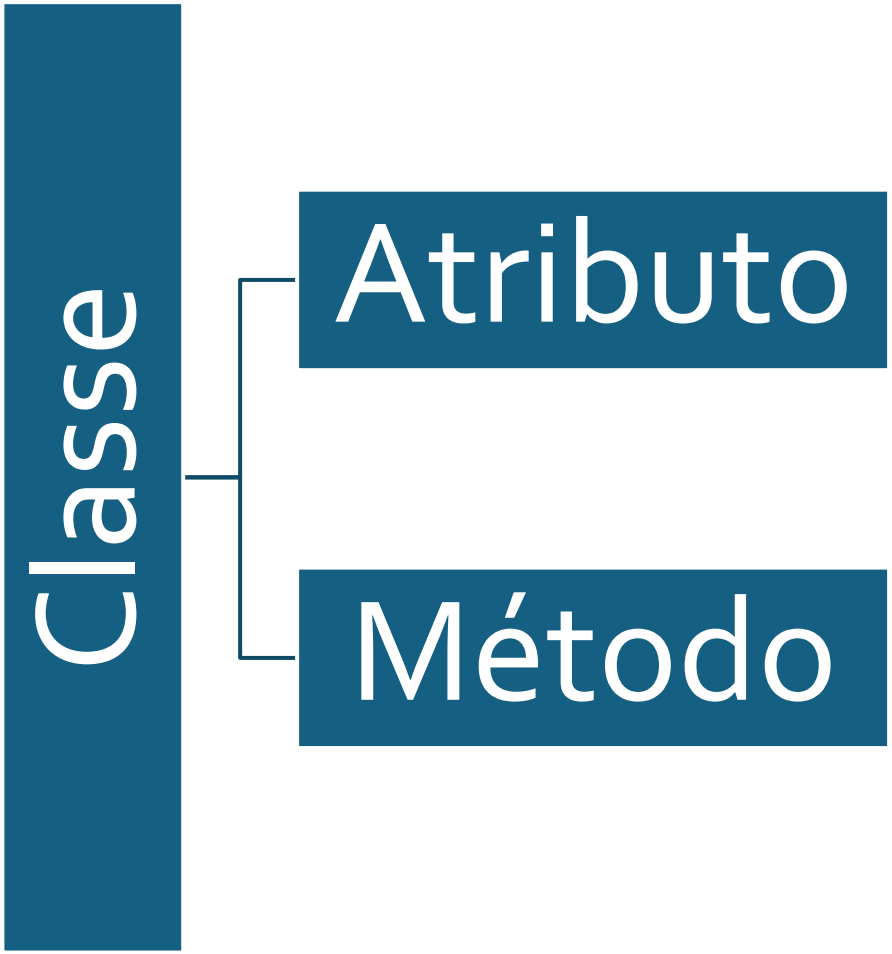
Na POO o programador é responsável por moldar o mundo dos objetos, e explicar para estes objetos como eles devem interagir entre si.

CLASSE

Uma classe funciona como uma "receita" para criar objetos. Inclusive, vários objetos podem ser criados a partir de uma única classe. Assim como várias casas ou prédios poderiam ser construídos a partir de uma única planta; ou vários bolos poderiam ser preparados a partir de uma única receita; ou vários carros poderiam ser construídos a partir de um único projeto.

Casas construídas a partir da mesma planta podem possuir características diferentes. Por exemplo, a cor das paredes.

CLASSE



ATRIBUTOS

Atributo

É um dado para o qual cada objeto tem seu próprio valor.

Definem características de objetos.

Exemplo de atributos:

Marca
Velocidade
Cor
Portas

MÉTODOS

Método

Definem as habilidades dos objetos.

Ações que os objetos podem executar.

Métodos são declarados dentro de uma classe para representar as operações que os objetos pertencentes a esta classe podem executar, ou seja, as ações que objeto pode fazer , um automóvel pode ter seus atributos cadastrado, os dados podem ser mostrados, alterados, podemos aumentar a velocidade.

OBJETO

Uma aplicação orientada a objetos é composta por objetos.

É uma Instância(criação) de uma classe

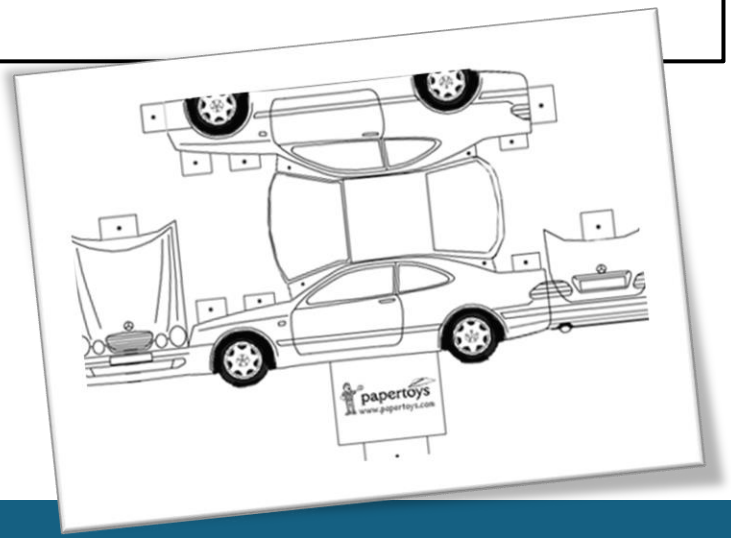
Podemos definir os objetos como sendo os objetos físicos do mundo real (carro, avião, cachorro, casa, telefone, computador).

Por exemplo, quando um cliente requisita um saque através de um caixa eletrônico do banco, o objeto que representa o caixa eletrônico deve interagir com o objeto que representa a conta do cliente.

Objeto

Um objeto é como se fosse uma casa ou um prédio. Para ser construído, precisa de um espaço físico. Esse espaço físico é algum trecho vago da memória do computador que executa a aplicação. No caso das casas e dos prédios, o espaço físico é algum terreno vazio.

Um prédio é construído a partir de uma planta criada por um engenheiro ou arquiteto. Para criar um objeto, é necessário algo semelhante a uma planta para que sejam "desenhados" os atributos e métodos que o objeto deve ter. Em orientação a objetos, a "planta" de um objeto é o que chamamos de classe.



Criar Classe em Java

Classes

A classe Conta poderia ser escrita utilizando a linguagem Java. A classe Conta é declarada utilizando a palavra reservada **class.**

```
class Conta
{
    public double saldo;
    public double limite;
    public int numero;
}
```

classe

atributos

Orientação Objetos em Java

- Assinatura de método
  - **Formado por:**
    - **nome do método.**
    - **argumentos** (parâmetros)
    - **tipo de resposta** (retorno)

Parâmetros: São valores(como variáveis) atribuídos
aos métodos para alterar o seu comportamento em tempo de execução.

```
public String calculaSalario(int valor)
```

Retorno — Nome — Parâmetro entrada

Orientação Objetos em Java

- Assinatura de método sem parâmetros e sem retorno
  - **Formado por:**
    - **nome do método.**
    - **argumentos** (parâmetros)
    - **tipo de resposta** (retorno)

```
public void calculaSalario()
```

Retorno vazio

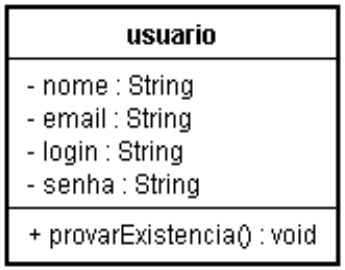
Nome

Sem parâmetros

UML

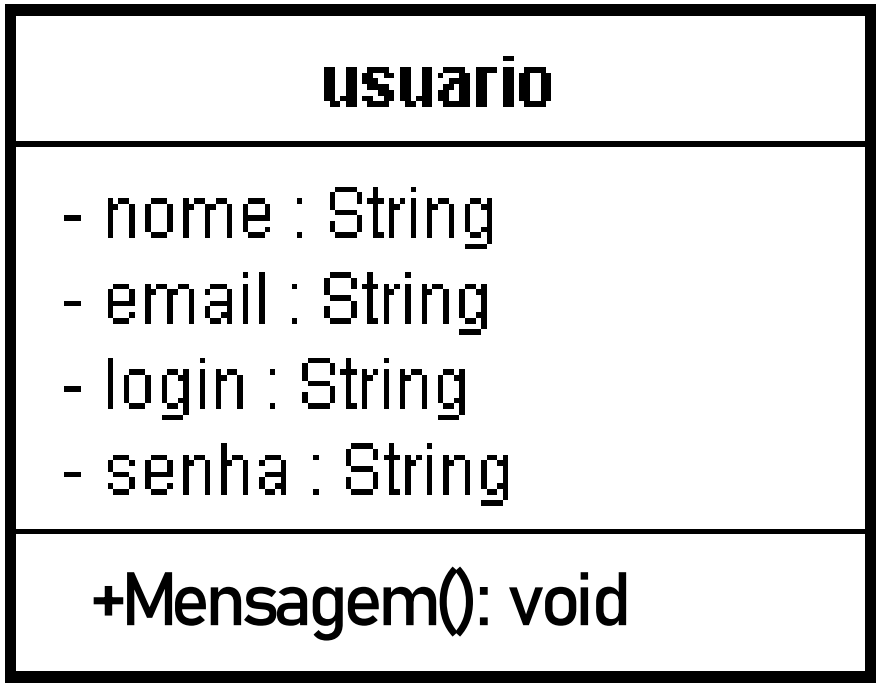
- A partir do momento em que os elementos básicos da orientação a objetos são assimilados, podemos modelar classes nas especificações corretas utilizando a principal ferramenta de modelagem e documentação de aplicações orientadas a objeto existente no mercado:

- A UML (Unified Modeling Language ou Linguagem unificada de modelagem).

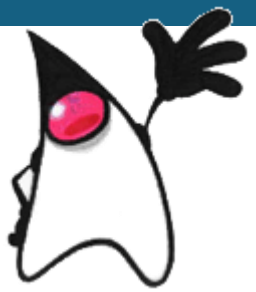


Classe de modelagem: Usuario

- Nome da classe
  - Usuario
- Atributos
  - nome, email, login e senha
- Métodos
  - Mensagem



CLASSE CARRO



Carro
- marca : String - velocidade : double - cor : String - porta: int
+cadastrarDadosCarro(): void +mostrarDadosCarro(): void +alterarDadosCarro() +aumentarVelocidade(int v):void

Tipos de Dados em JAVA

Classificação	Tipo	Descrição
Lógico	boolean	Pode possuir os valores true (verdadeiro) ou false (falso)
Inteiro	byte	Abrange de -128 a 127 (8 bits)
	short	Abrange de -32768 a 32767 (16 bits)
	int	Abrange de -2147483648 a 2147483647 (32 bits)
	long	Abrange de -2^{63} a $(2^{63})-1$ (64 bits)
Ponto Flutuante	float	Abrange de 1.40239846^{-46} a 3.40282347^{+38} com precisão simples (32 bits)
	double	Abrange de $4.9406564584124654^{-324}$ a $1.7976931348623157^{+308}$ com precisão dupla (64 bits)
Caracter	char	Pode armazenar um caracteres unicode (16 bits) ou um inteiro entre 0 e 65535

Modificadores de acesso

- Modificadores de acesso
  - Private
    - Só fica visível dentro da classe em que foi implementado
  - Public
    - Fica visível em toda a aplicação
  - Protected
    - Fica visível na classe em que foi implementado e em suas sub-classe

ENCAPSULAMENTO

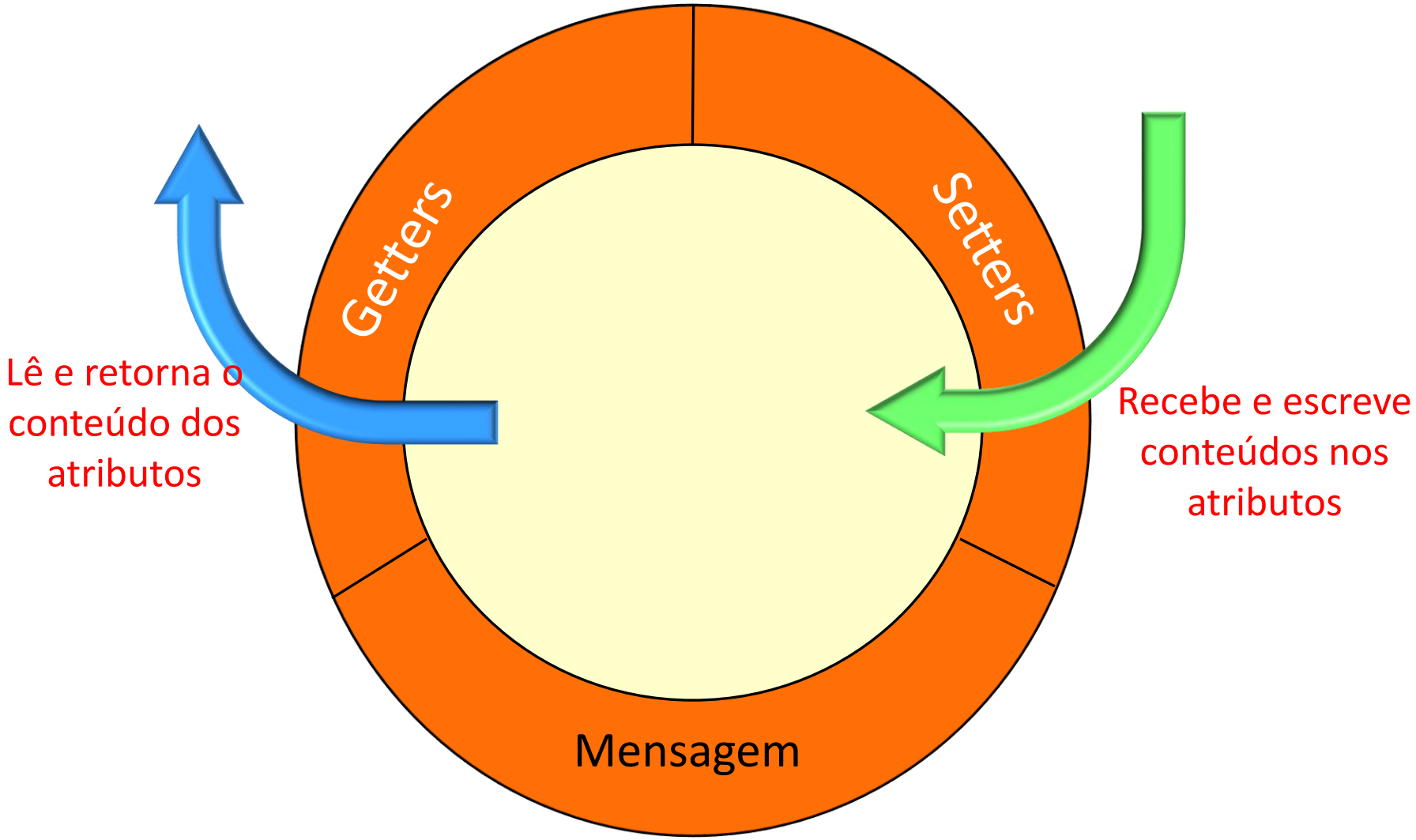
- Consiste na proteção dos atributos de uma classe (e posteriormente dos objetos) de acessos externos.

- Considerando que todas as regras referentes a classe estão contidas na própria classe (e nunca em outra parte da aplicação)

ENCAPSULAMENTO

- Métodos getters e setters.
- Cada atributo tem seus próprios métodos públicos getter e setter.
  - Getter: Lê o conteúdo de um atributo e retorna seu valor.
  - Setter: Recebe um valor por parâmetro e altera (escreve) tal valor no referente atributo.

Objeto usuario1



Codificação da classe Usuario

Atributos

```
public class Usuario {

    // Atributos
    private String nome;
    private String email;
    private String login;
    private String senha;
```

Construtores

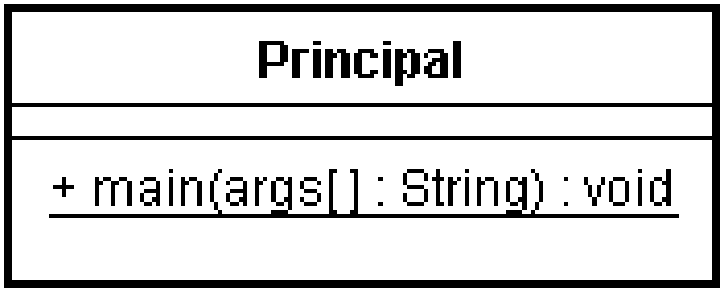
```
    // Construtores
    //  Inicializa os atributos vazios
    public Usuario() {
        this("","","","");
    }
    //  Inicializa os atributos com valores passados por parametro
    public Usuario(String email, String login, String nome, String senha) {
        this.email = email;
        this.login = login;
        this.nome = nome;
        this.senha = senha;
    }
```

Getters e Setters
Encapsulamento

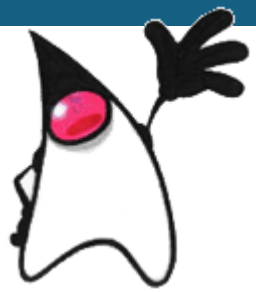
```
    // Getters e Setters
    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
```

CLASSE MAIN

- A classe principal será usada somente para conter o método main



CLASSE USUÁRIO E PRINCIPAL



Usuario

- nome : String
- email : String
- login: String
- senha : String

+getNome(): String
+setNome(nome: String): void
+getEmail(): String
+setEmail(email: String): void
+getLogin(): String
+setLogin(nome: String): void
+getSenha(): String
+setSenha(senha: String): void
+Mensagem(): void

Principal

+ main(args{}: String): void

COMO INSTANCIAR UM OBJETO

```
public class Principal {

    public static void main(String[] args) {

        Usuario usuario1 = new Usuario();
```

Tipo

Comando

Nome do novo objeto

Construtor

COMO REALIZAR A CHAMADA DE UM MÉTODO

```
public class Principal {

    public static void main(String[] args) {

        Usuario usuario1 = new Usuario();

        usuario1.provarExistencia();

    }

}
```

Nome do novo objeto

Método

Codificação da classe Usuario

```
public class Usuario {

    // Atributos
    private String nome;
    private String email;
    private String login;
    private String senha;

    // Construtores
    //  Inicializa os atributos vazios
    public Usuario() {
        this("","","","");
    }
    //  Inicializa os atributos com valores passados por parametro
    public Usuario(String email, String login, String nome, String senha) {
        this.email = email;
        this.login = login;
        this.nome = nome;
        this.senha = senha;
    }

    // Getters e Setters
    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    // Implementação dos demais getters e setters

    // Métodos especificos da classe
    public void provarExistencia(){
        System.out.println("Oi, eu existo!");
    }
}
```

Codificação da classe Principal

```
public class Principal {

    public static void main(String[] args) {

        Usuario usuario1 = new Usuario();

        usuario1.provarExistencia();



    }

}
```

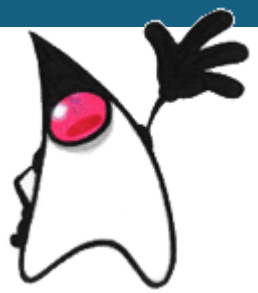
ESTRUTURA BÁSICA DO PROJETO JAVA

- Classe de modelagem (Definição de tipo)
  - Atributo
  - Contrutores
  - Getters e setters
  - Métodos especificos da classe
- Classe Principal
  - Método main()

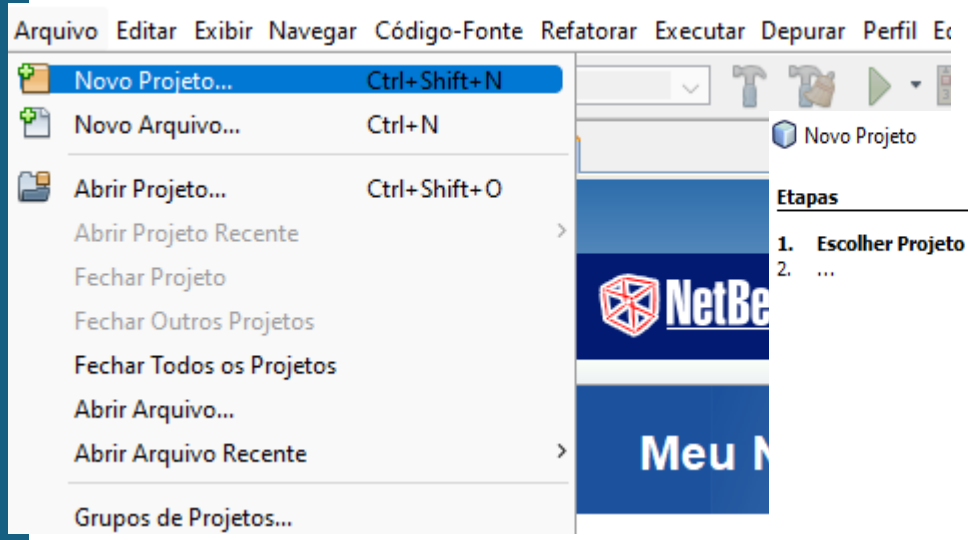
Desenvolvimento da aplicação PrimeiroPrograma

- Passo a passo para a criação da aplicação exemplo utilizando o Netbeans
- Definição do workspace
  - Antes mesmo de começar devemos definir nossa área de trabalho onde salvará os arquivos gerados.
  - Crie uma pasta com qualquer nome e em qualquer unidade de disco e considere-a como sua workspace.

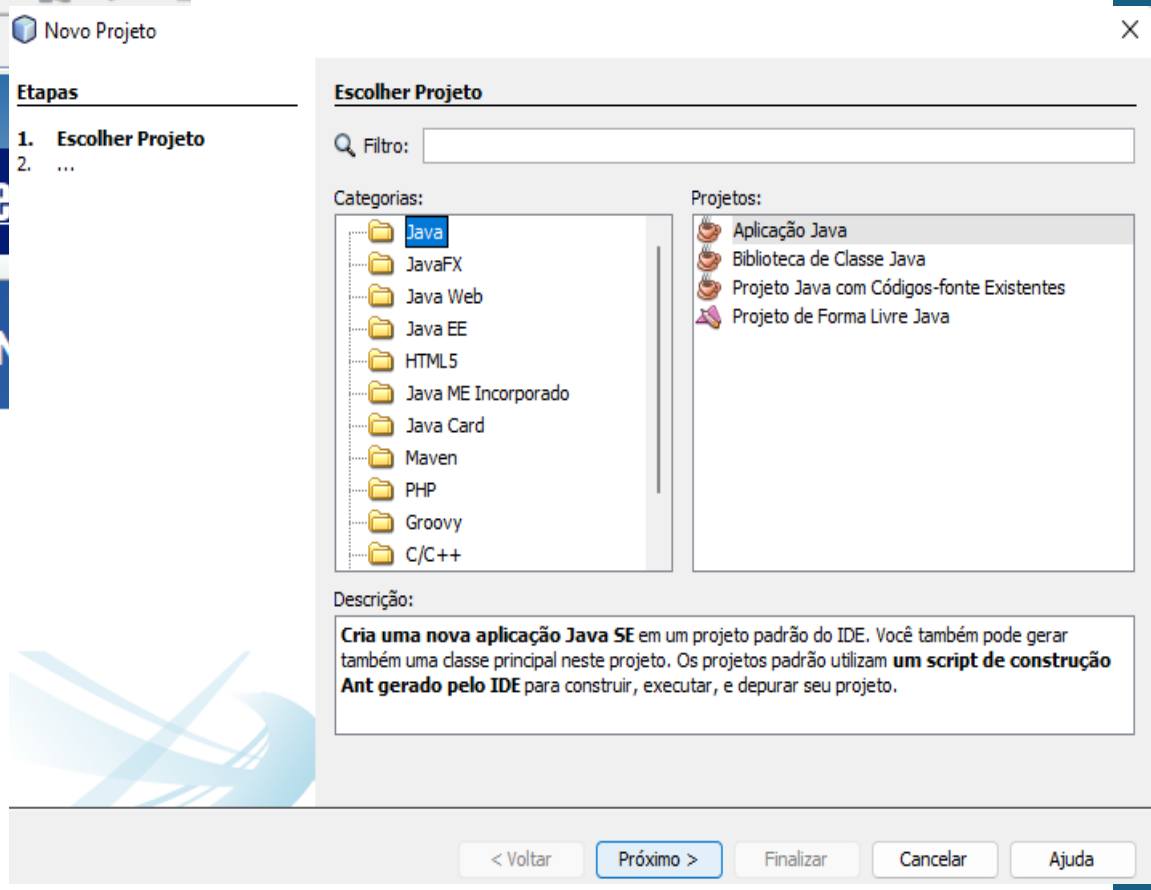
Criando o PrimeiroPrograma



Clique em Arquivo Novo Projeto...

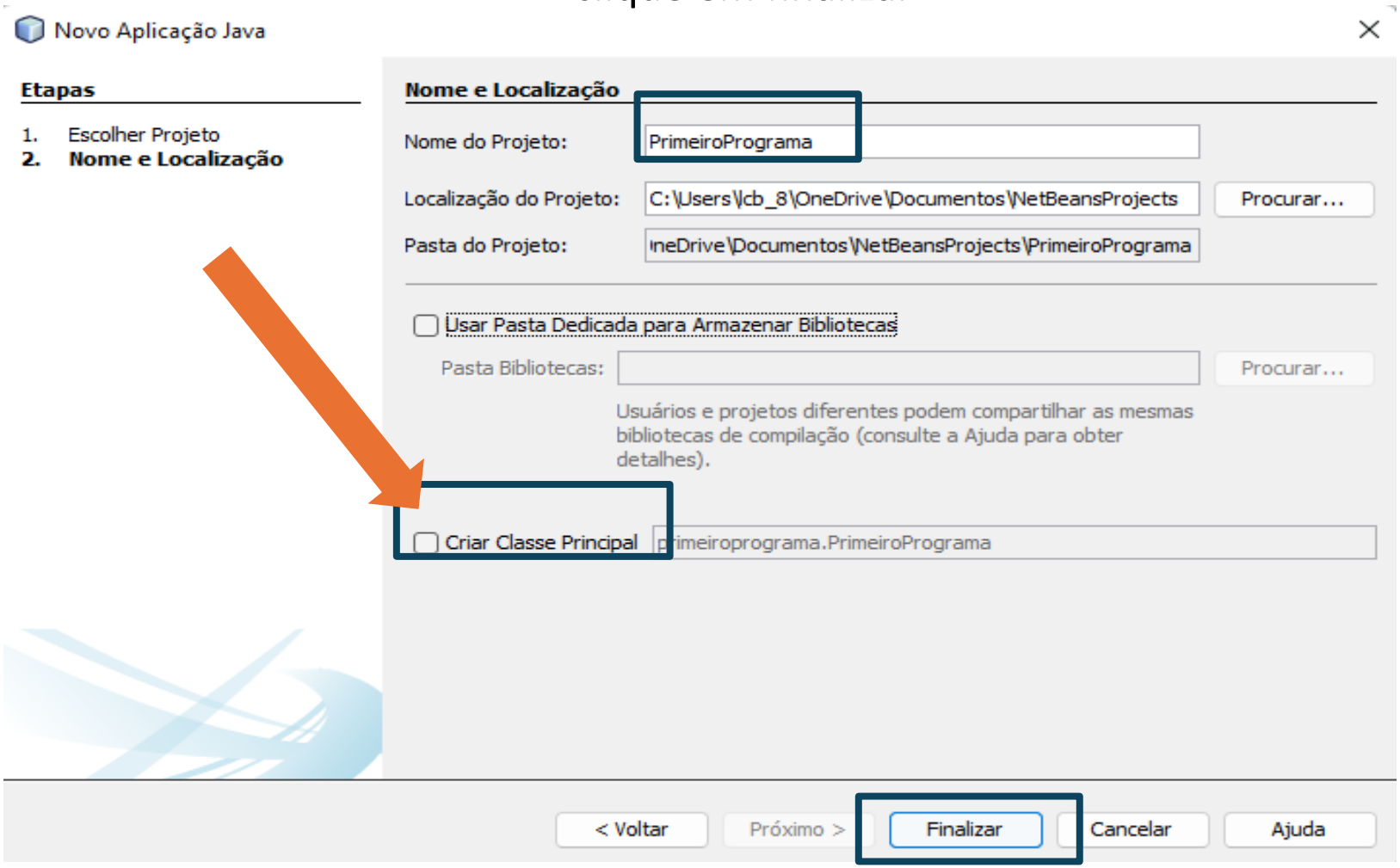


Selecione Java – Aplicação Java

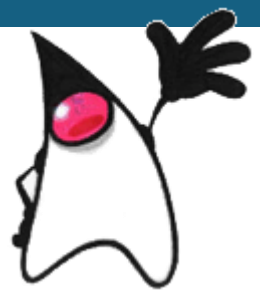


Criando o PrimeiroPrograma

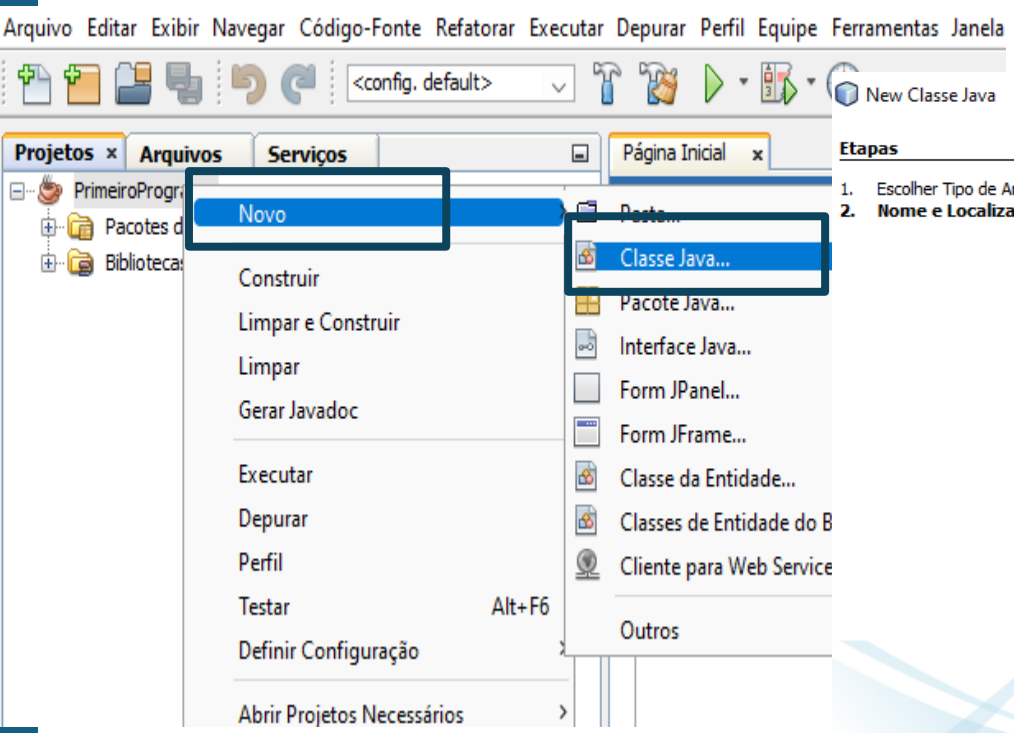
Defina o Nome do Projeto e Desmarque a caixa para criar a Classe Principal, depois clique em finalizar



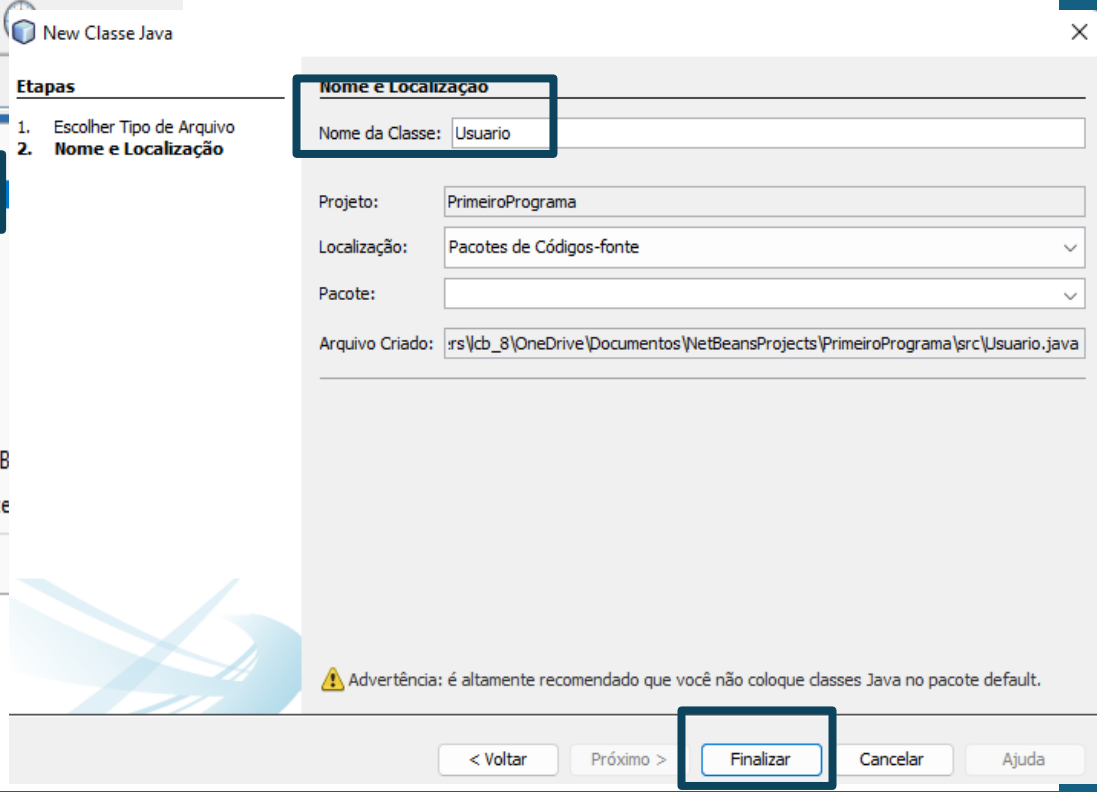
Criação de uma classe



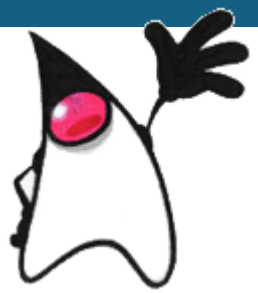
Para Criar uma nova classe, clique com o botão direito no nome do projeto , Novo-> Classe Java...



Defina um classe com nome Usuario, depois clique em Finalizar



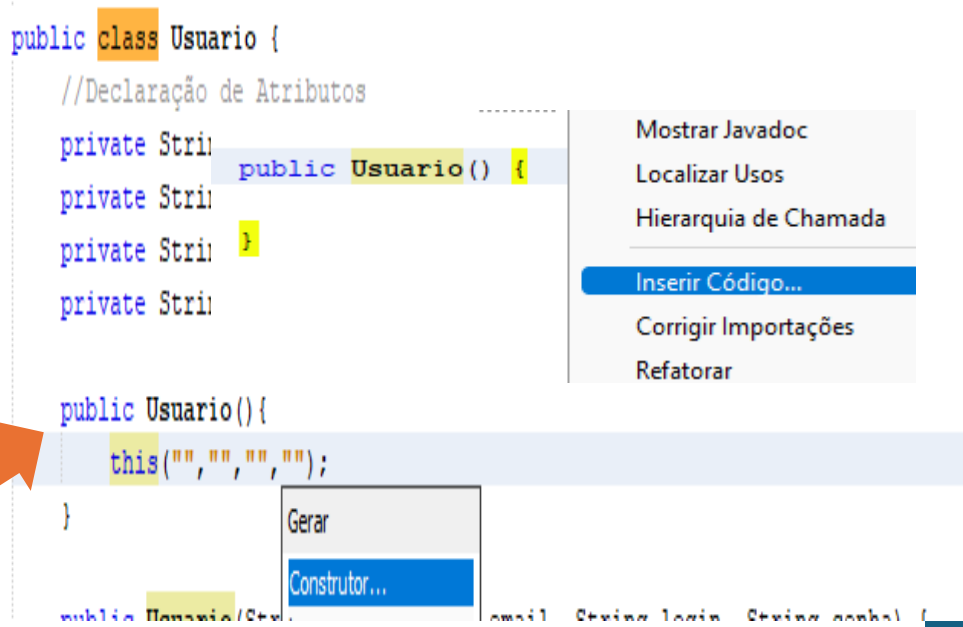
Criação Criação da estrutura da classe de modelagem-Construtores



Declaração dos atributos

```
public class Usuario {
    //Declaração de Atributos
    private String nome;
    private String email;
    private String login;
    private String senha;
}
```

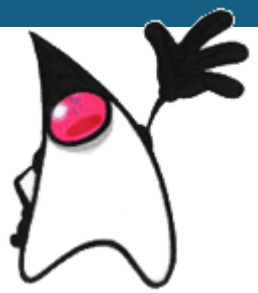
Geração de construtores



Escreva o construtor Usuario , digite this(“”,””,””,””), que define a quantidade de atributos do construtor iniciando os atributos com o valor vazio

Para gerar o construtor automático , clique botão direito inserir Código -> Construtor...

Criação Criação da estrutura da classe de modelagem-Construtores



```
public class Usuario {
    //Declaração de Atributos
    private Stri
    private Stri
    private Stri
    private Stri

    public Usuario() {

    }

    public Usuario(){
        this("","","","");
    }
```

Mostrar Javadoc
Localizar Usos
Hierarquia de Chamada
Inserir Código...
Corrigir Importações
Refatorar

Gerar
Construtor...

Gerar Construtor

Selecione os campos a serem inicializados pelo construtor:

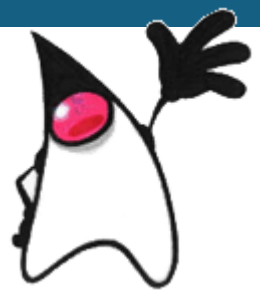
- nome : String
- email : String
- login : String
- senha : String

```
public class Usuario {
    //Declaração de Atributos
    private String nome;
    private String email;
    private String login;
    private String senha;

    public Usuario() {
        this("","","","");
    }

    public Usuario(String nome, String email, String login, String senha) {
        this.nome = nome;
        this.email = email;
        this.login = login;
        this.senha = senha;
    }
```

Gerar | Cancelar

Para gerar o construtor automático , clique botão direito inserir Código -> Construtor... -> Na tela que abrir selecione os campos que serão iniciados pelo construtor e clique em gerar

Criação Criação da estrutura da classe de modelagem

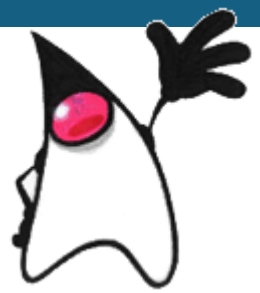


Construtores gerados

```
public class Usuario {
    //Declaração de Atributos
    private String nome;
    private String email;
    private String login;
    private String senha;

    //Construtores

    //Inicializa os atributos vazios
    public Usuario() {
        this("","","","");
    }

    //Construtores com passagem de parâmetros
    public Usuario(String nome, String email, String login, String senha) {
        this.nome = nome;
        this.email = email;
        this.login = login;
        this.senha = senha;
    }
```

Acrescentar

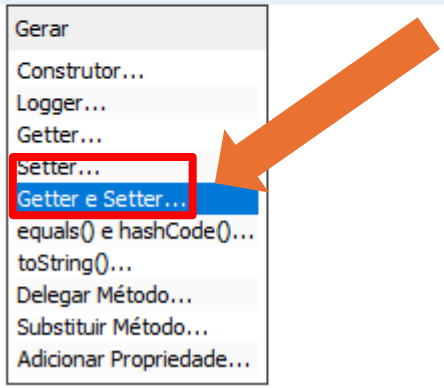
Criação Criação da estrutura da classe de modelagem – Getters e Setters



Geração de getters e setters

Clique com o botão direito em em inserir código-Getter e Setter

```
14      //Construtores com passagem de parâmetros
15      public Usuario(String nome, String email, Stri
16          this.nome = nome;
17          this.email = email;
18          this.login = login;
19          this.senha = senha;
20      }
21
22
23
24
25  }
26
27
28
29
```

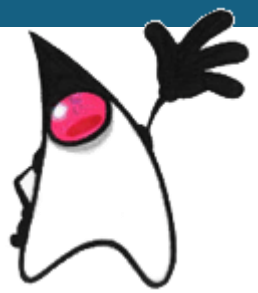


getters e setters gerados

Usuario.java

```
public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getLogin() {
    return login;
}

public void setLogin(String login) {
    this.login = login;
}

public String getSenha() {
    return senha;
}

public void setSenha(String senha) {
    this.senha = senha;
}
```

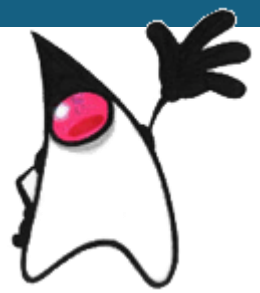
Criando Um método Mensagem-Classe Usuario



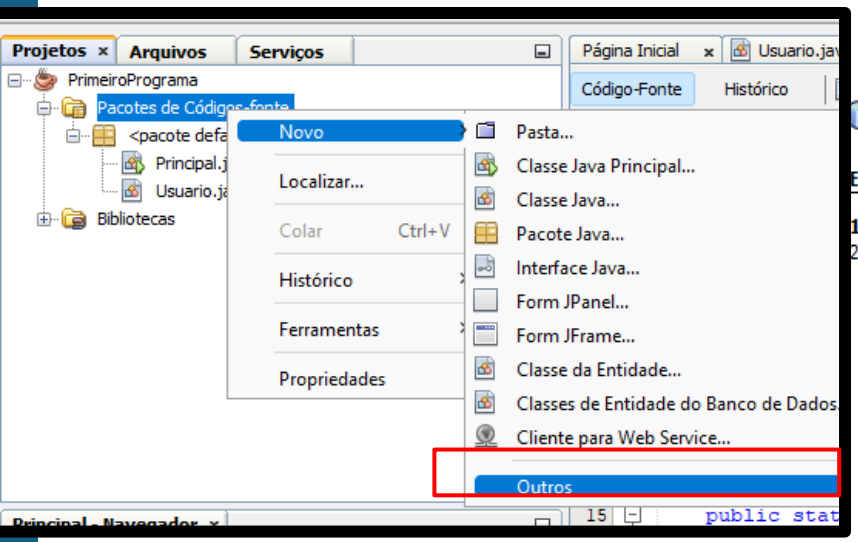
Implementar o método Mensagem

```
,

public void Mensagem(){
    System.out.println("Bem Vindo ao Java Usuário");
}
```

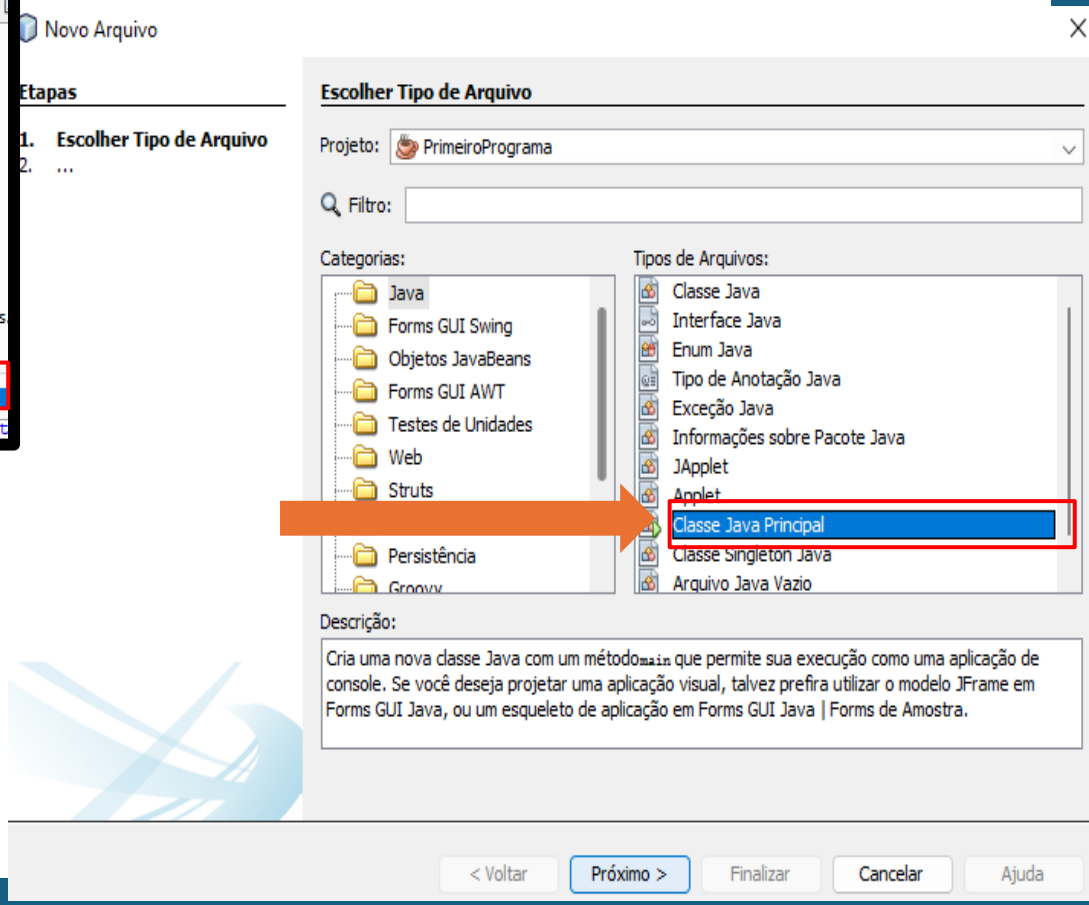
Criando a Classe Principal



Clique com o botão direito em novo-> Outros...

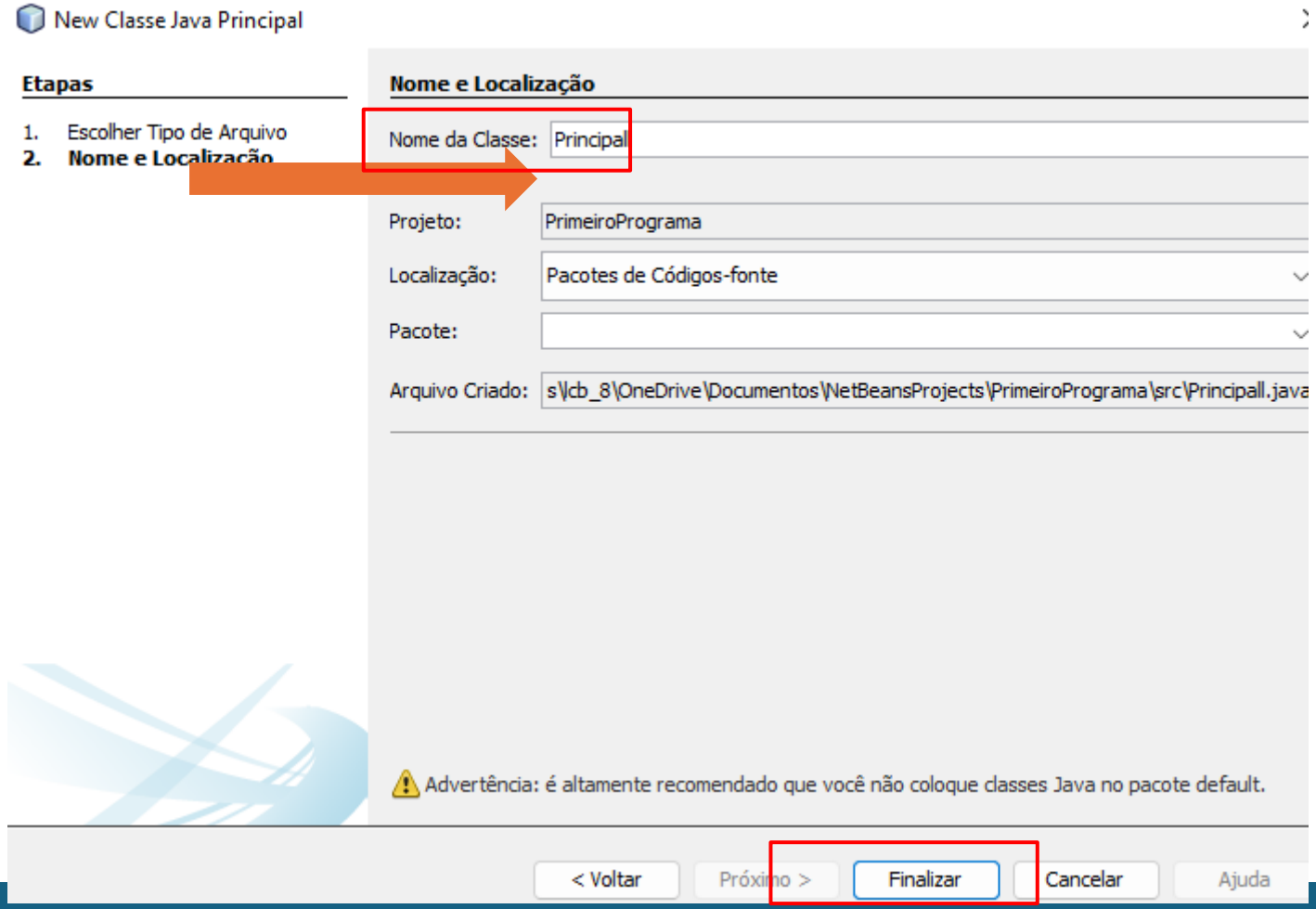


Selecione a Categoria Java o Arquivo Classe Java Principal

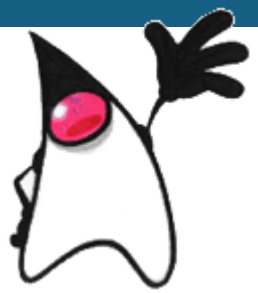


Criando a Classe Principal

Coloque o nome da Classe como Principal e clique em Finalizar



Criando a Classe Principal-Instanciando um Objeto



```
public static void main(String[] args) {
    // TODO code application logic here

    //Instaciando um objeto
    Usuario usuariol = new Usuario();
}
```

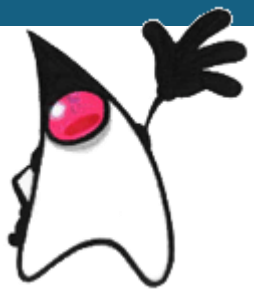
Criamos objeto usuario1 , quepossui todos os métodos definidos na classe Usuario , este é objeto é instanciado dentro do método Principal(main)

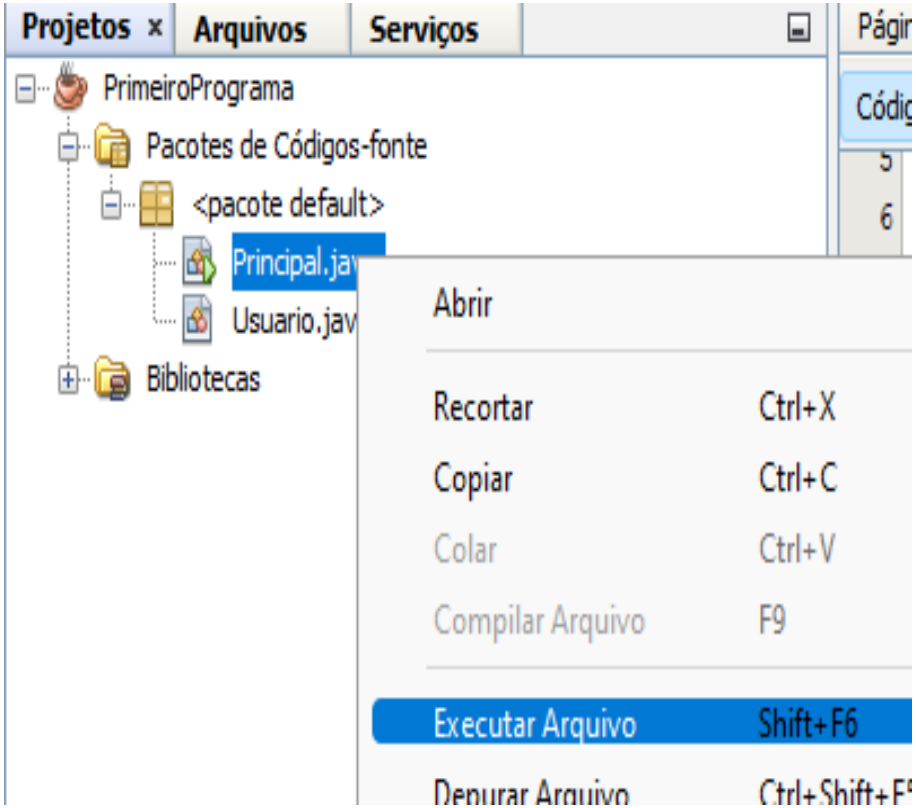
Para chamar um método de uma classe , colocamos o nome do objeto , seguido do nome do método

Usuario1.Mensagem

```
public static void main(String[] args) {
    // TODO code application logic here

    //Instaciando um objeto
    Usuario usuariol = new Usuario();
    usuariol.Mensagem();
}
```

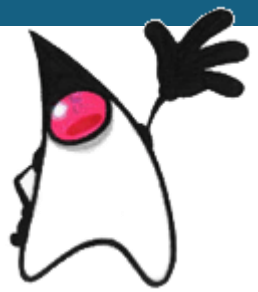
Executando um Projeto em Java



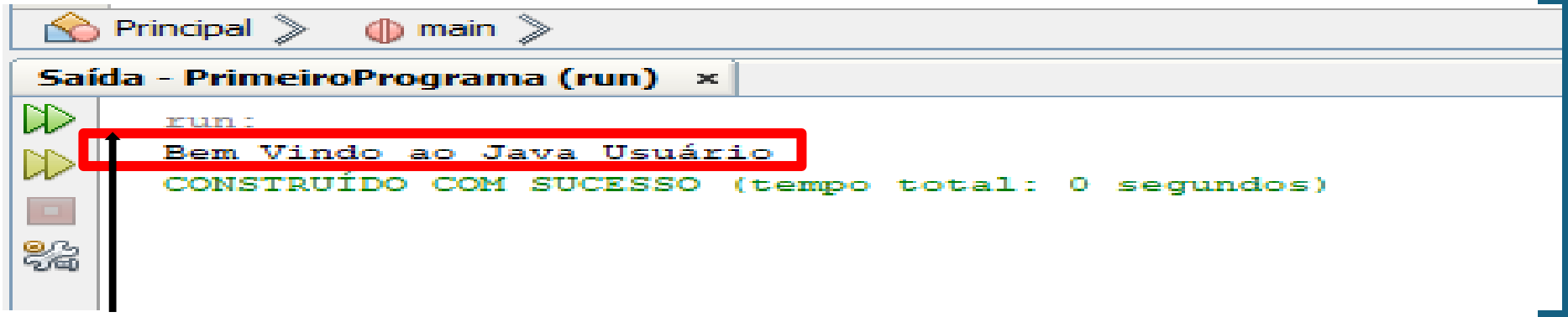


Clicar com o botão direito do mouse sobre a classe que contém o método main, clique dm Executar Arquivo , ou Pressione F6

Executando um Projeto em Java

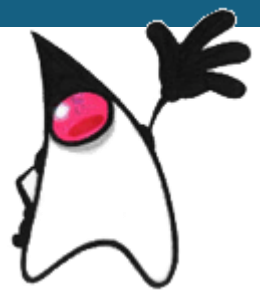


O painel console simula um prompt em ambiente texto



Enfim, a emocionante manifestação de vida do objeto usuario1

Incrementando a Aplicação

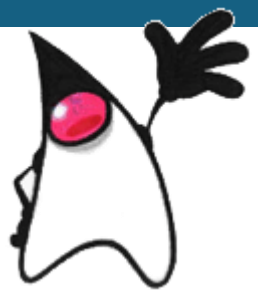


Vamos acrescentar na classe Principal, armazenando valores nos atributos da classe Usuário, para guardar valores nos atributos , utilizamos a função set.

Classe Principal: Armazenando dados nos atributos, para enviar a classe Usuário

```
public static void main(String[] args) {
    // TODO code application logic here

    //Instaciando um objeto
    Usuario usuario1 = new Usuario();
    usuario1.setNome("Luiz");
    usuario1.setEmail("luiz@claudio.com");
    usuario1.setLogin("luizinho");
    usuario1.setSenha("123456");
    usuario1.Mensagem();
}
```

Incrementando a Aplicação

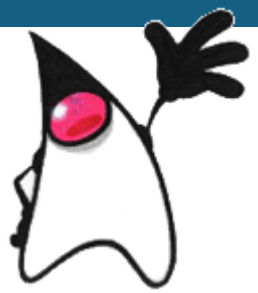


Inserindo o conteúdo dos atributos na classe Usuario, dentro do método Mensagem, para resgatar os valores dos atributos utilizamos a função get

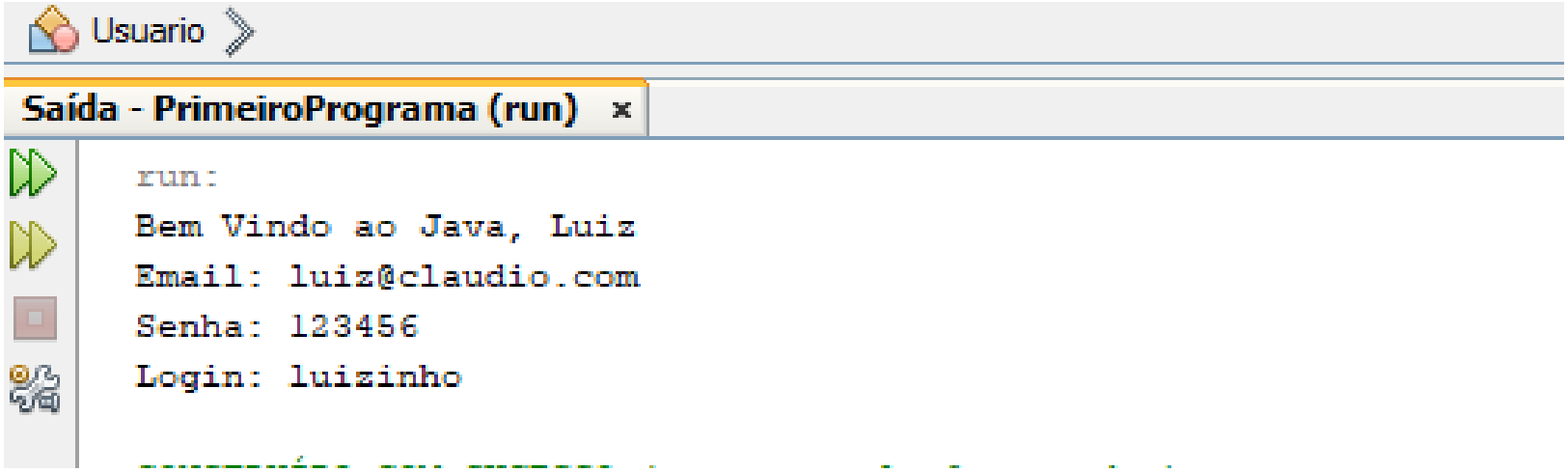
Classe Usuario: Alterar o método Mensagem

```
public void Mensagem(){
    System.out.println("Bem Vindo ao Java, " + getNome() + "\n" +
            "Email: " + getEmail() + "\n" +
            "Senha: " + getSenha() + "\n" +
            "Login: " + getLogin() + "\n" );
}
```

Executando novamente Projeto em Java



Pressione F6 e veja o resultado:



Exercício 1

Contatos
- Nome : String - Telefone : String - Endereco: String - Cidade : string
+ cadastrarDados:void() + mostrarDados:void()

Principal
+ main(args{}: String): void