



*SEU FUTURO ESTÁ AQUI!*

**APTECH**  
**COMPUTER EDUCATION**

# Introdução à Programação Orientada a Objetos



# Introdução à Programação Orientada a Objeto

- Duração: 8 horas-aula
- Conteúdo:
  - ✓ Classe
  - ✓ Objeto
  - ✓ Atributo
  - ✓ Método

# Programação Orientada a Objeto

# POO

- Algumas linguagens populares Orientada a Objeto:
  - ✓ C#
  - ✓ C++
  - ✓ Java



# C#

- C# (C sharp) é uma linguagem de programação orientada à objetos de propósitos gerais.
- Foi criada pela Microsoft e faz parte do framework .NET.
- Não roda diretamente no hardware, como em C, C++ ou Assembly.

# C#

- A sintaxe é inspirada na linguagem C.
- C# foi bastante baseada em Java.
- A linguagem C# foi criada junto com a arquitetura .NET.



# C#

- Foi criada praticamente do zero para funcionar na nova plataforma, sem preocupações de compatibilidade com código de legado.
- A maior parte das classes do .NET Framework foram desenvolvidas em C#.

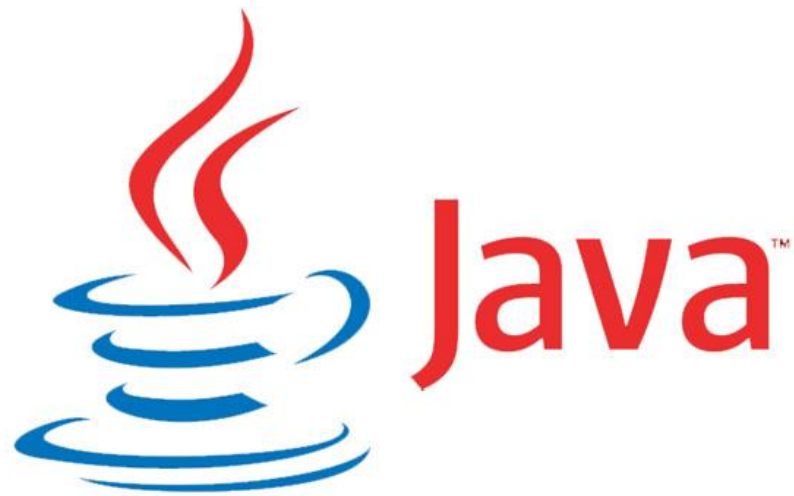


# C++

- C++ é uma linguagem de programação orientada a objetos, fundamental para diversos sistemas operacionais e programas, com interface gráfica.
- Por ser mais eficiente, C++ é utilizado em vários ambientes.

# C++

- C++ pode ser utilizado em diversos sistemas operacionais como: jogos, vídeo games, softwares embarcados (para celulares, eletrodomésticos, firewall e routers) e terminais de pagamento como o Cielo.



# Java

- Java foi criado em 1995 pela empresa Sun.
- Programação Orientada a Objeto.
- Multiplataforma: Conceito “write once, run everywhere”.

# Vida do Java

- 1995:
  - Lançamento da tecnologia Java
- 1996:
  - Lançamento do JDK
  - 83.000 páginas web utilizando Java
- 1997:
  - Lançamento Servlets e Java Web Server (Java EE)
- 1998:
  - Projeto Swing (Java cliente-servidor)
  - Formalizado Java Community Process (JCP)

# Vida do Java

- 1999:
  - Lançamento da tecnologia JavaServer Pages (JSPs)
  - Sun anuncia as três edições da plataforma Java: J2SE, J2EE e J2ME
  - Java SE versão 1.3
- 2001:
  - Java SE versão 1.4
- 2004:
  - Java SE versão 5



# Vida do Java

- 2007:
  - Java SE versão 6
  - Torna-se open source
- 2009:
  - A Oracle compra a Sun
- 2011:
  - Java SE versão 7
- 2014:
  - Java SE versão 8

# Java

- **Java SE: soluções para Desktops**
  - Aplicações autônomas (stand alone); Applets
- **Java EE: soluções corporativas**
  - eCommerce; eBusiness
- **Java ME: soluções para o consumidor**
  - Celulares; PDAs; set-top boxes TV; GPS; robôs

# Por que Java?

# Java

- Java é a base da grande maioria de aplicativos em rede.
- Padrão global para desenvolvimento e fornecimento de aplicativos para celular, jogos, conteúdo on-line e software corporativo.

# Java

- 1,1 bilhão de desktops executam Java.
- 930 milhões de download do Java Runtime Environment a cada ano.
- 3 bilhões de telefones celulares executam Java.
- 100% de Blu-ray players executam Java.

# **Programação Estruturada X Programação Orientada a Objeto**

# O que é Programação Estruturada?

# Programação

- Na Programação Estruturada é necessário fazer a programação linha por linha (passo a passo).
- Permite que o programador pegue trechos de maior uso do seu programa e transforme-os em sub-rotinas(procedimentos e funções) que serão reutilizados sempre que necessário.



# O que é Programação Orientada a Objeto?

# Programação

- A Programação estruturada fica limitada quando um projeto fica grande demais.
- A Programação Orientada a Objeto pegou as melhores ideias da programação estruturada e combinou-as com vários conceitos novos.

# Programação

- A Programação Orientada a Objeto foi criada para tentar simular o mundo real dentro do computador, a partir de Objetos.
- Para se criar os Objetos, utiliza-se as Classes.

# O que é Classe?

# Classe

- A Classe é um modelo (molde, gabarito, esqueleto, rótulo, etc...) criado para caracterizar um grupo.
- Exemplo de Classes:
  - Animais
  - Veículos
  - Eletrodomésticos
  - Estudantes

# Classe

- Na Classe são definidos quais são os atributos e os métodos.
- Os atributos são as características dos Objetos e os métodos são as ações que os Objetos realizam.

# Classe

- Exemplos de atributos:
  - nome
  - idade
  - data de nascimento
  - telefone
- Exemplos de métodos:
  - somar dois números
  - verificar CPF
  - calcular o troco

# Classe

- O tamanho/abrangência da Classe depende da qualidade e quantidade dos atributos e métodos.
- Mais exemplos de Classes:
  - Animais
  - Mamíferos
  - Cachorros



# O que é Objeto?

# Objeto

- O Objeto é uma representação em software de entidades (coisas) do mundo real.
- No Objeto, define-se o conteúdo/valores de cada um dos atributos.

# Classe x Objeto

Classe (Estudantes)	Objeto1 (aluno1)	Objeto2 (aluno2)
nome	Marcelo	Luísa
idade	30	10
data de nascimento	27/05/1984	12/10/2003
telefone	3242-2821	3242-2821



# Classe x Objeto

Classe	Objeto
É um modelo conceitual.	É uma coisa real.
Descreve uma entidade.	É a entidade.
Consiste em campos (membros de dados) e funções.	É uma instância de uma classe.

**Classe**



**Objeto**



# Objeto

- A partir da Classe cria-se o Objeto.
- Todo o Objeto nasce de uma Classe.
- Pode-se criar diversos Objetos a partir de uma única Classe.
- Criar um Objeto = Instância de uma Classe.
- Criar um Objeto = Instanciar um Objeto.

**IDE**

# IDE

- IDE é um programa que reúne ferramentas para desenvolvimento de software, ou seja, é o ambiente onde você desenvolverá seu aplicativo.
- IDE do inglês *Integrated Development Environment*.
- É possível editar código, depurá-lo (analisar e procurar erros) e manter todos os arquivos de um projeto atualizados.

# IDE para Java

- As duas IDE mais utilizadas para Java são:
- NetBeans
- Eclipse



# NetBeans



**NetBeans**

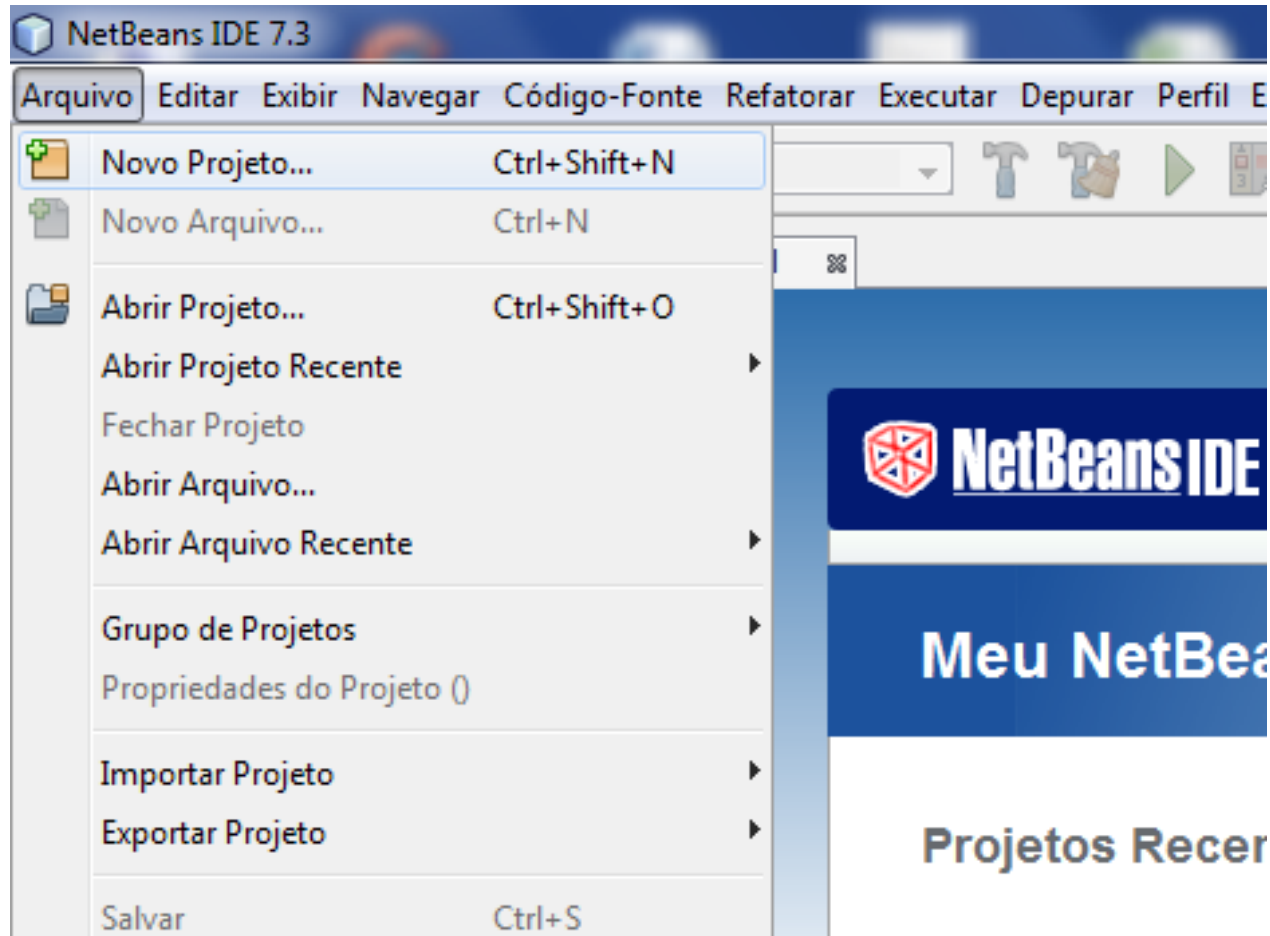
# NetBeans

- O NetBeans é um ambiente de desenvolvimento integrado (IDE) gratuito e de código aberto para desenvolvedores de software na linguagem Java.
- Para usar o NetBeans junto com o JDK(Java Development Kit), basta baixar no site:  
<http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html>

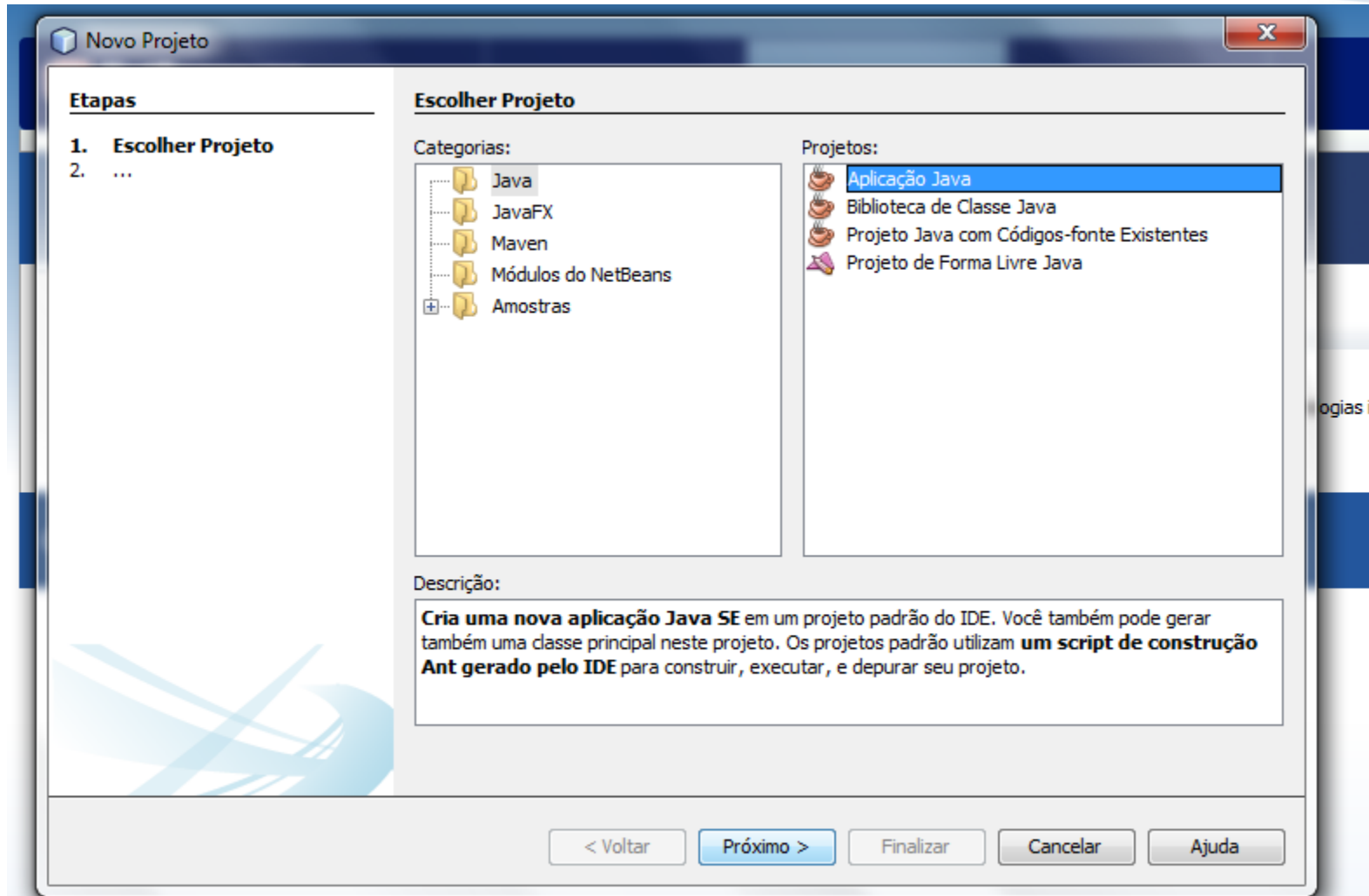
# Primeiro programa

- Criar uma pasta na Área de trabalho.
- Abrir o NetBeans e criar um novo projeto chamado “Primeiro programa”.

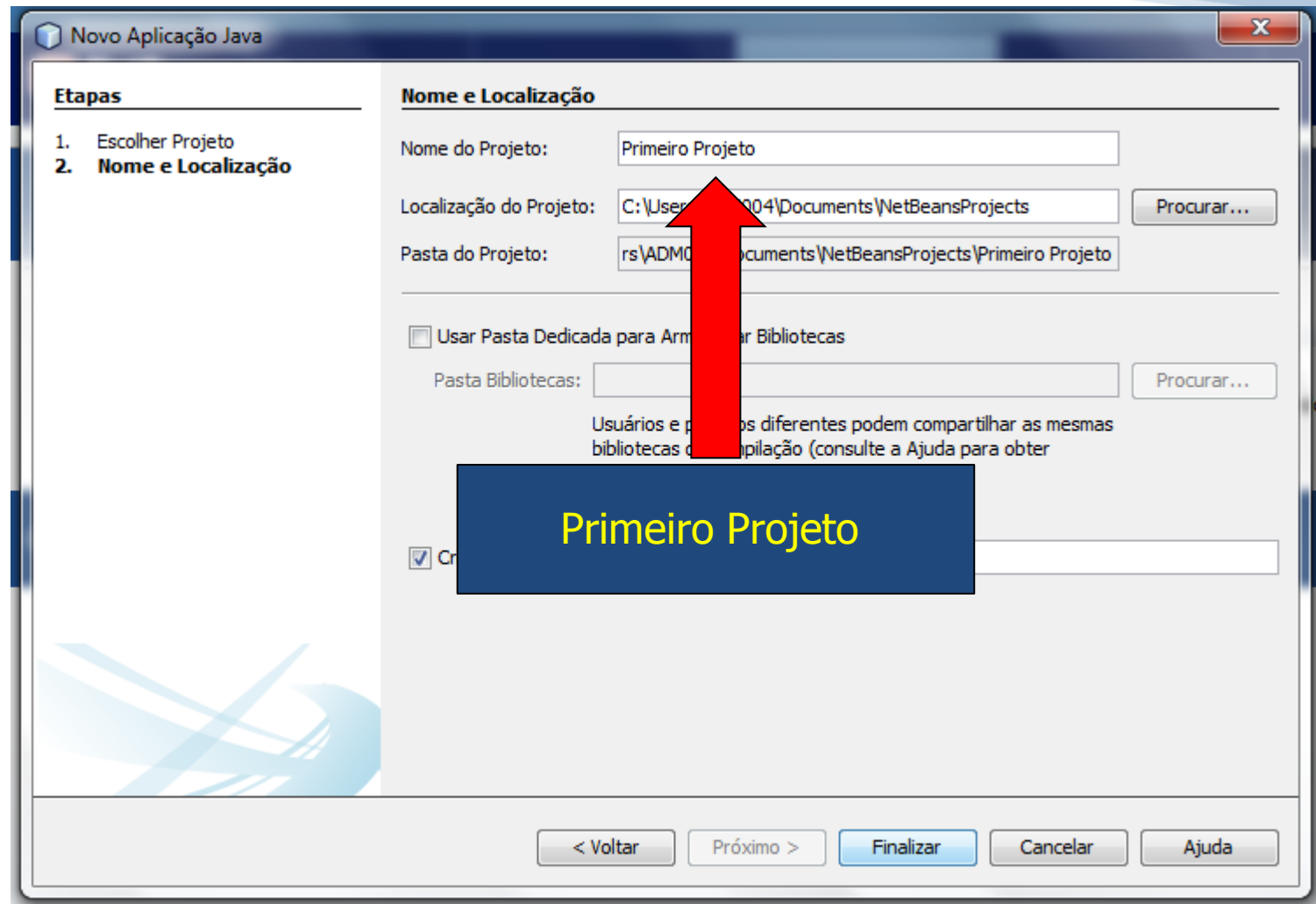
# Primeiro programa



# Primeiro programa



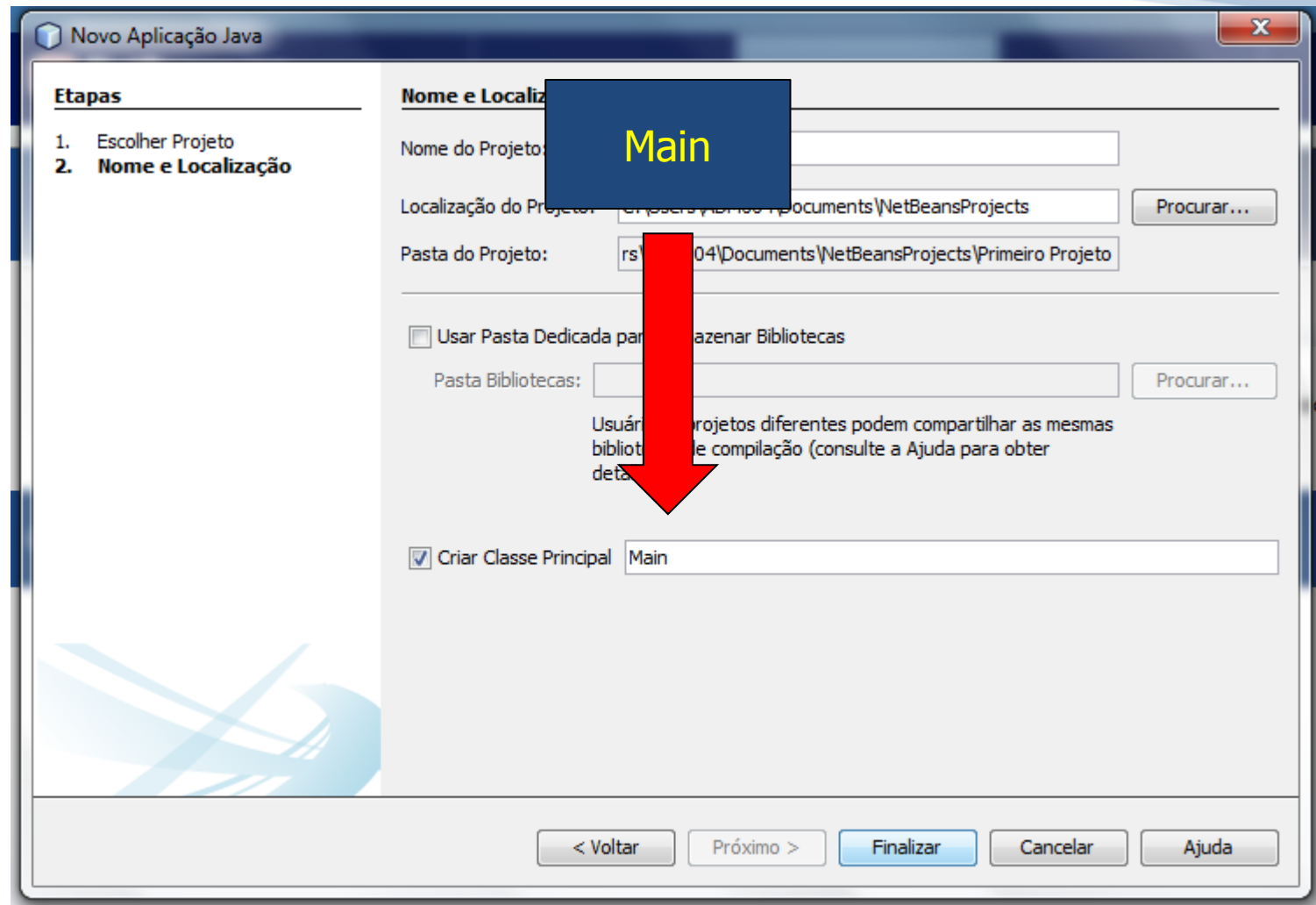
# Primeiro programa



# Primeiro programa

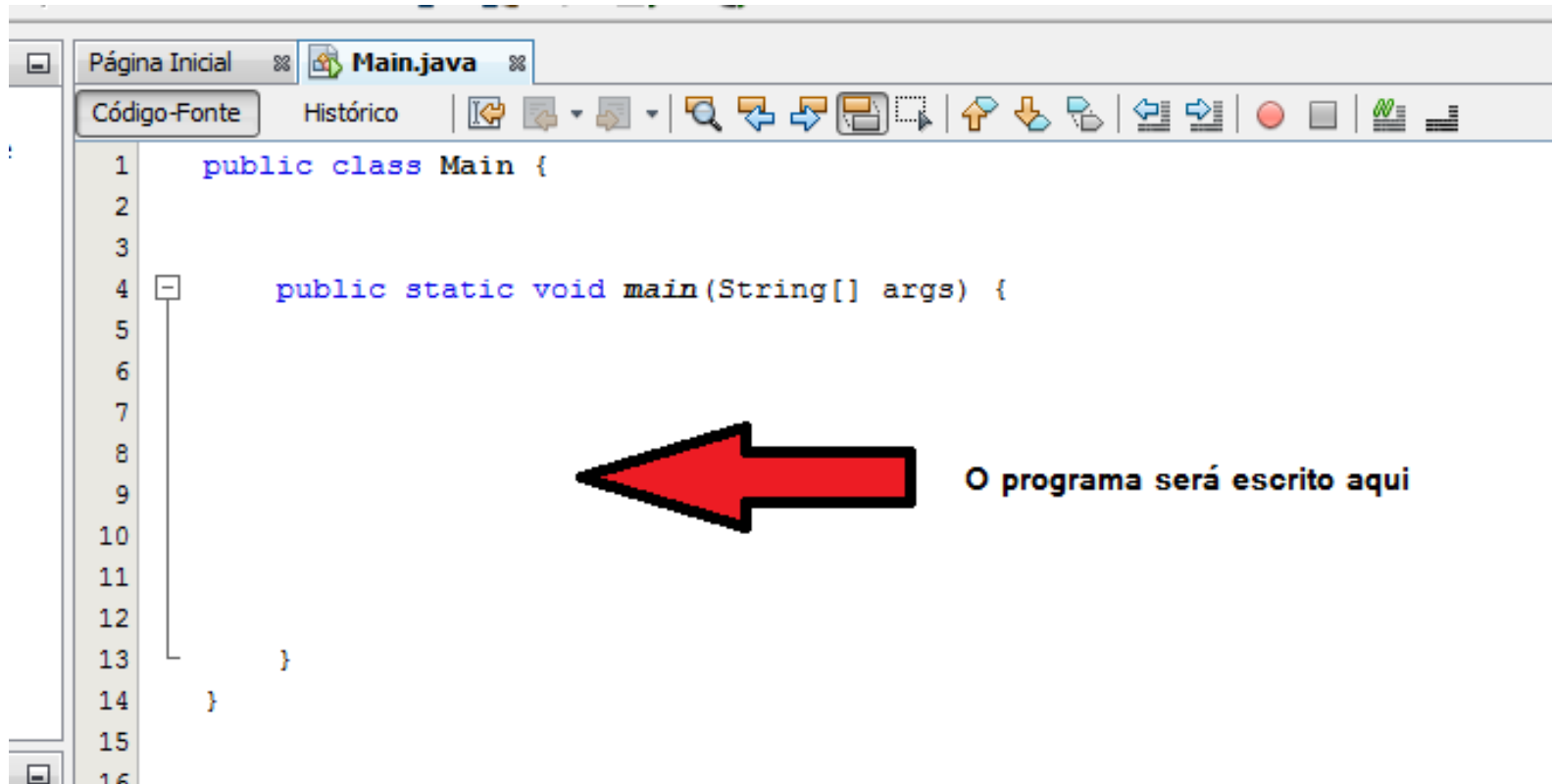
- Chamar a classe principal de “Main”.
- Dentro da Classe “Main”, será feito o nosso programa.

# Primeiro programa





# Primeiro programa



The screenshot shows an IDE window titled 'Main.java'. The code editor contains the following Java code:

```
1 public class Main {  
2  
3  
4     public static void main(String[] args) {  
5  
6  
7  
8  
9  
10  
11  
12  
13     }  
14 }  
15  
16
```

A red arrow points to the space between lines 4 and 13, indicating where the program code should be written.

O programa será escrito aqui

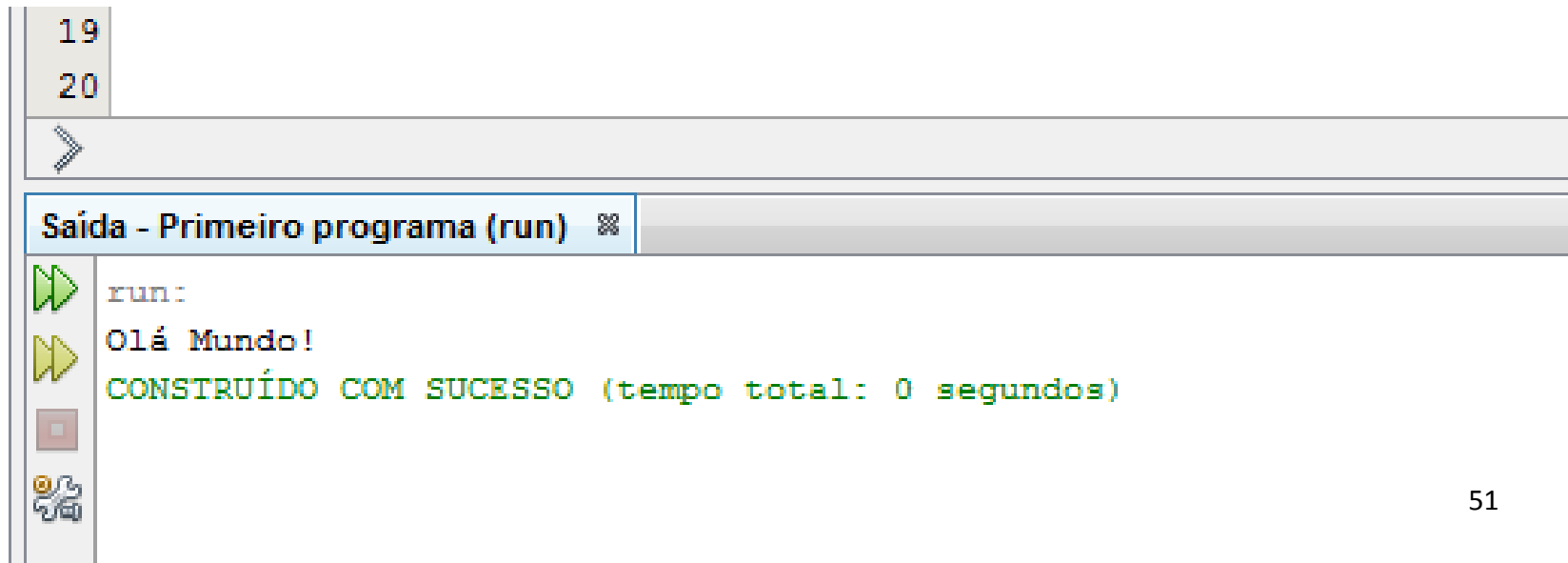
# Primeiro programa

- Dentro da Classe “Main”, o primeiro programa será mostrar na tela “Olá mundo!”.
- Usar o comando “sout” + tecla “TAB” e escrever “Olá Mundo !”.
- Ficará escrito da seguinte forma:  
`System.out.println("Olá Mundo!");`

# Primeiro programa

- Agora é só executar o programa, apertando “F6”.
- Na parte inferior do NetBeans aparecerá o resultado do programa.

s)



# Exercícios

- 1) Faça um programa que mostre o texto “Exemplo de texto.”
  
- 2) Faça um programa que mostre na primeira linha o seu nome e na segunda a linha a sua data de nascimento.

# Tipos de variáveis

- Na programação orientada a Objeto, os atributos possuem tipos, assim como as variáveis.
- As mais comuns são:
  - ✓ int -> números inteiros
  - ✓ double -> números reais
  - ✓ char -> caracteres
  - ✓ String -> palavras ou frases

# Tipos de variáveis

- Exemplos:
  - ✓ `int num = 54;`
  - ✓ `double troco = 65.80;`
  - ✓ `char letra = 'R';` //usar apóstrofos
  - ✓ `String palavra = "exemplo";` //usar aspas

# Tipos de variáveis

- Para mostrarmos o valor de uma variável, utilizamos o comando:

```
System.out.println("O valor da variável é "+variavel);
```

# Exemplo

- Criar um programa que mostre o valor da variável nome.

```
- */  
    public class Main {  
  
        /**  
         * @param args the command line arguments  
         */  
        public static void main(String[] args) {  
            String nome;  
            nome = "Roberto";  
  
            System.out.println("O nome é "+nome);  
        }  
    }  
}
```



# Outro exemplo

- Criar outro projeto, “Soma de dois números”.
- Criar novamente a Classe principal “Main”.
- Dentro da Classe principal será feito a soma de dois números.

# Outro exemplo

- Criar as variáveis reais N1, N2 e soma.
  - ✓ double N1=10;
  - ✓ double N2=20;
  - ✓ double soma=0;
- Colocar a fórmula da soma:  
soma = N1+N2;

# Outro exemplo

- Mostrar o resultado da soma:  
`System.out.println("O resultado é "+soma);`
- Para rodar o programa apertar o "F6".

# Classes

# Exemplo

## Criando uma Classe

**Novo Aplicação Java**

**Etapas**

1. Escolher Projeto
2. **Nome e Localização**

**Nome e Localização**

Nome do Projeto:

Localização do Projeto:

Pasta do Projeto:

☐ Usar Pasta Dedicada para Armazenar Bibliotecas

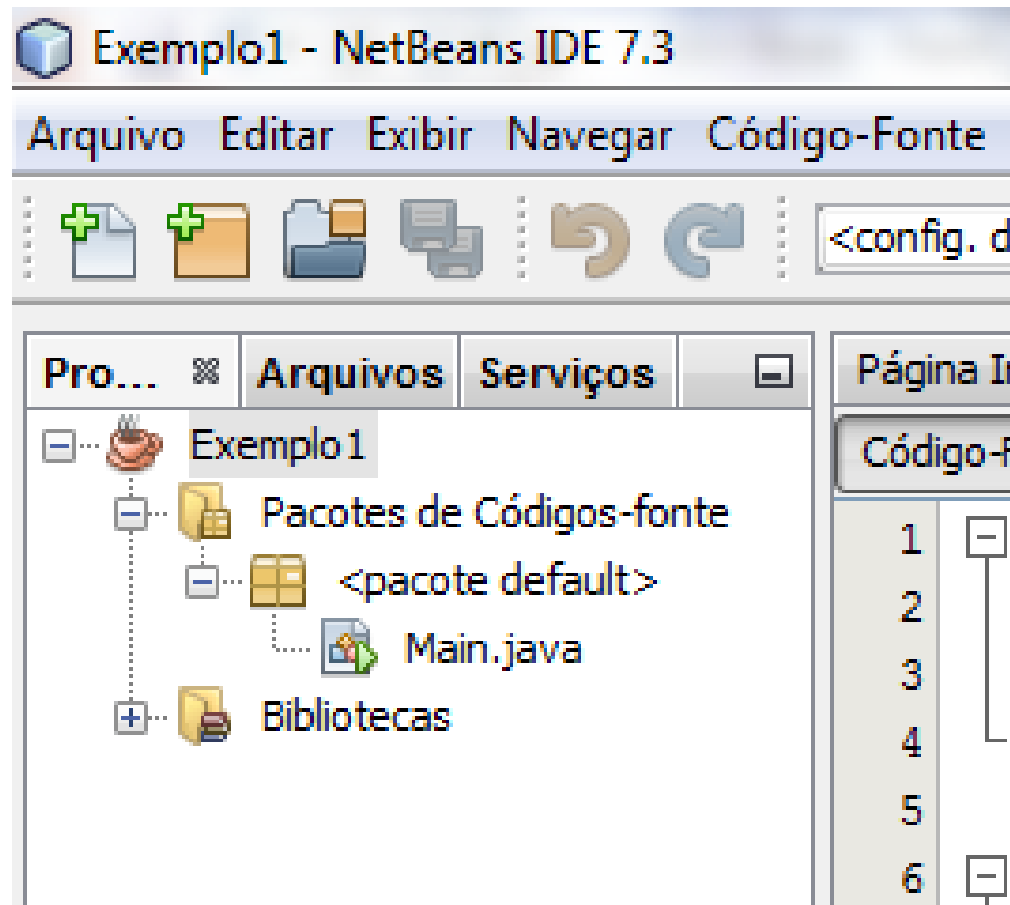
Pasta Bibliotecas:

Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).

☒ Criar Classe Principal

# Exemplo

## Criando uma Classe



# Classe

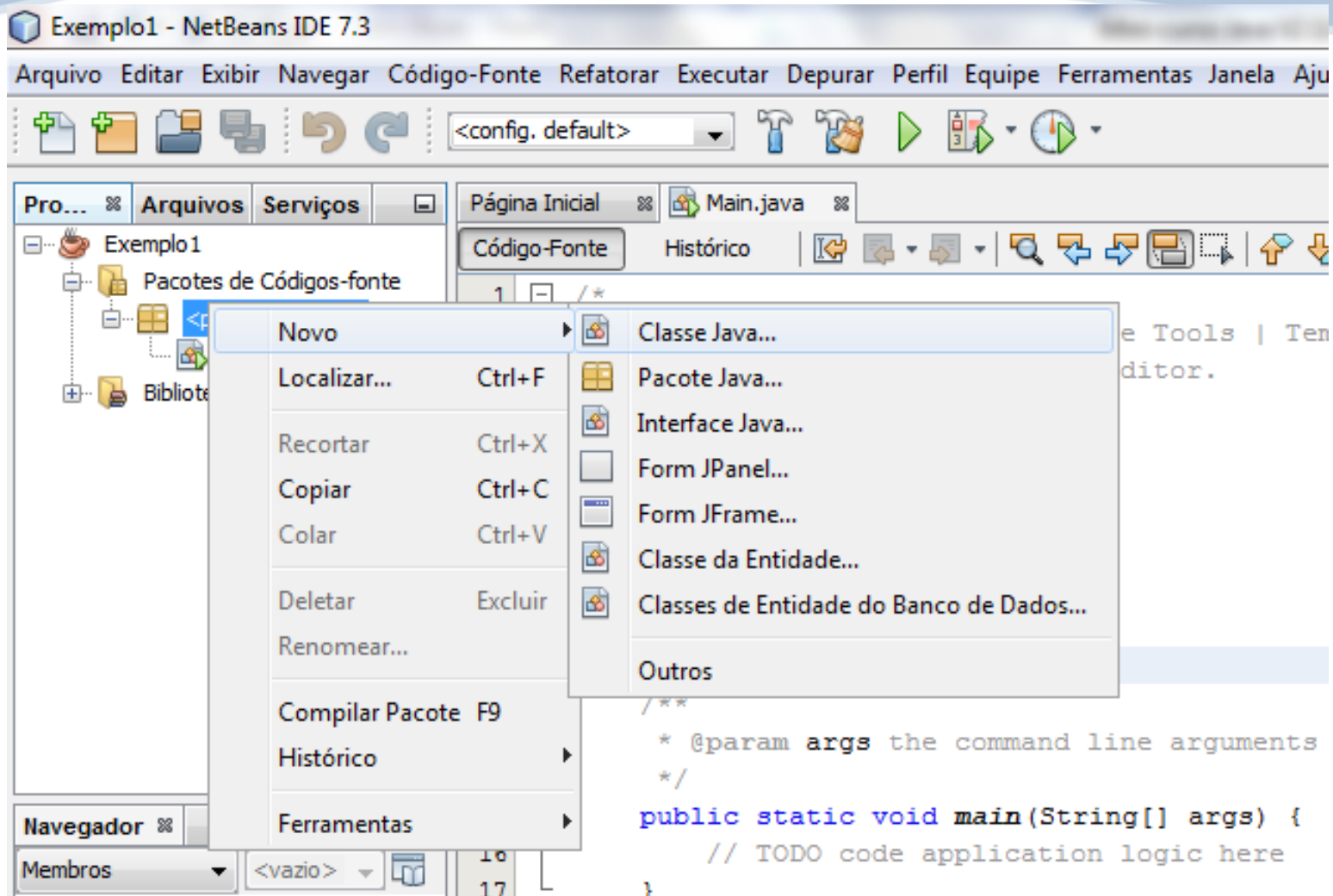
- Para criar uma nova Classe:

No lado esquerdo da tela, estão os projetos salvos. Criar a classe desejada lá. Clicar botão direito -> "Novo" -> "Classe Java..." -> Nomeie a Classe.

- O nome da classe deve **sempre** começar com letra **Maiúscula**.

# Exemplo

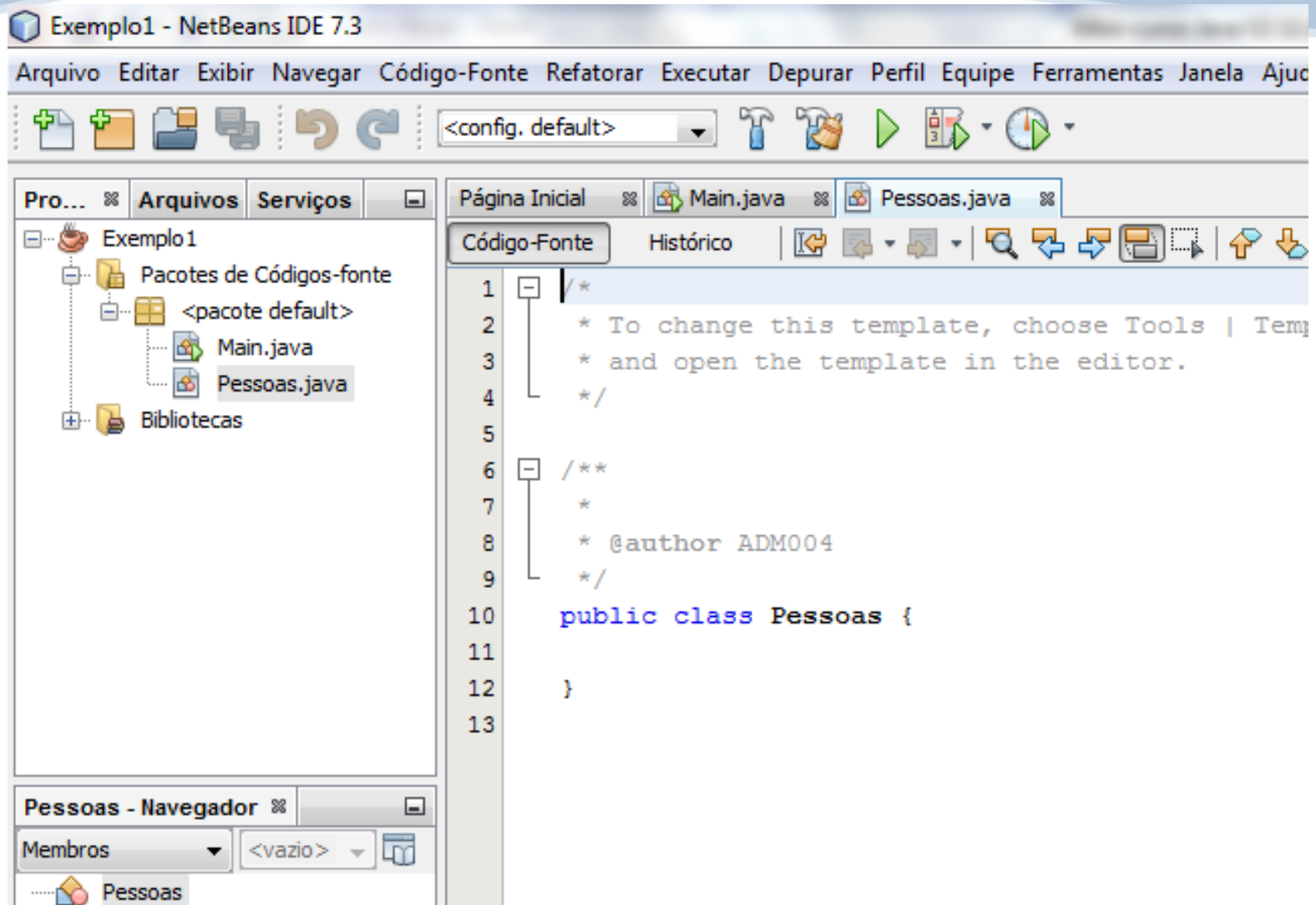
## Criando uma Classe





# Exemplo

## Criando uma Classe



# Classe

- Criar a Classe Estudantes com os atributos:

nome

idade

data de nascimento

telefone

# Exemplo

## Criando uma Classe

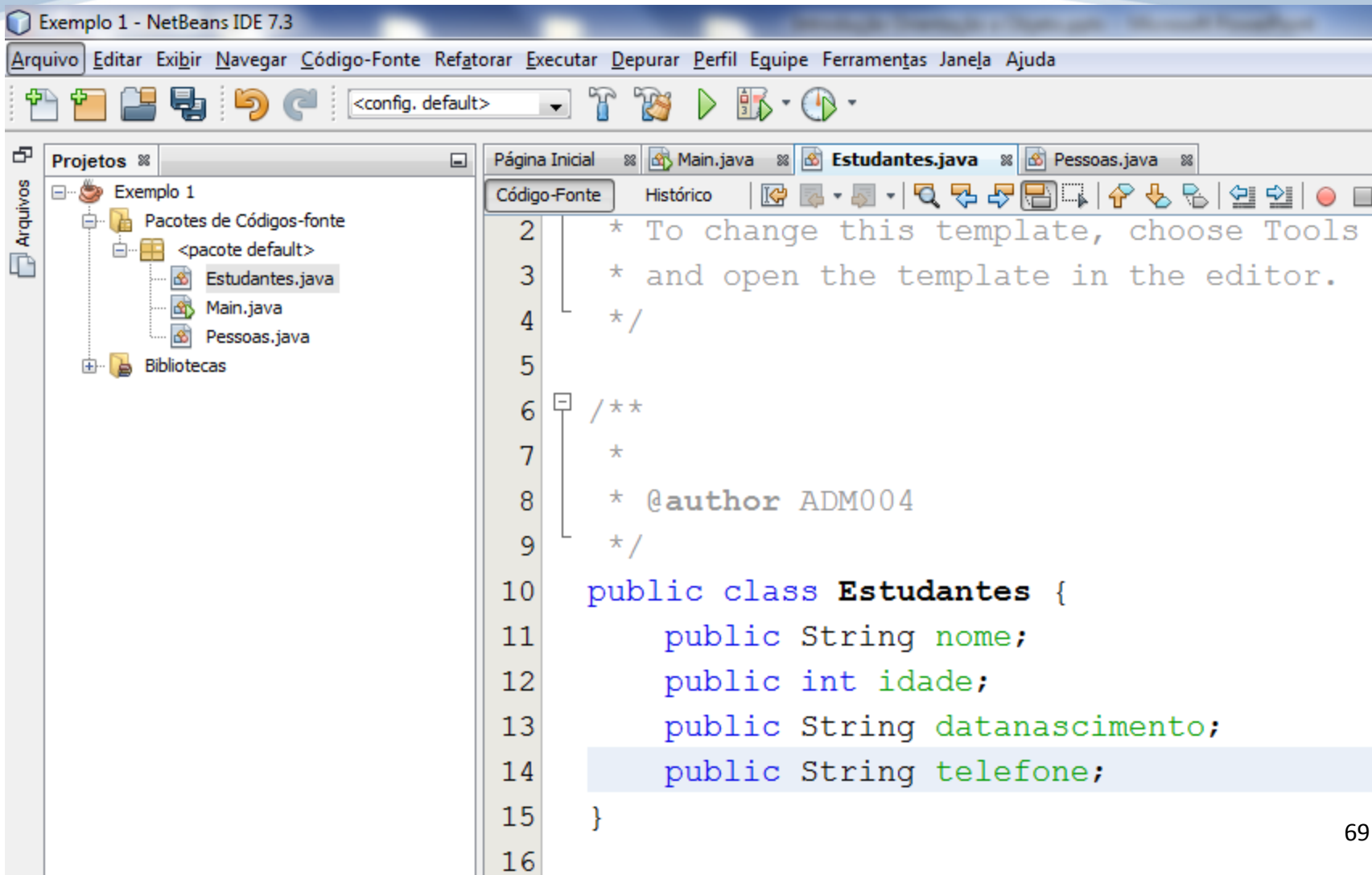
The screenshot shows the NetBeans IDE 7.3 interface. The main editor window displays the source code for 'Estudantes.java'. The code includes a package declaration, a class comment, and the definition of the 'Estudantes' class with four attributes: 'nome' (String), 'idade' (int), 'datanascimento' (String), and 'telefone' (String). A blue box with the word 'Atributos' and a red arrow points to the attribute declarations. The left sidebar shows the project structure with 'Exemplo1' containing 'Pacotes de Códigos-fonte' and 'Bibliotecas'. The bottom sidebar shows the 'Membros' (Members) view for 'Estudantes', listing the attributes: 'datanascimento : String', 'idade : int', 'nome : String', and 'telefone : String'.

```
1  /*
2   * This is a template, choose the appropriate
3   * template in the
4   */
5
6  /**
7   *
8   * @author ADM004
9   */
10 public class Estudantes {
11     String nome;
12     int idade;
13     String datanascimento;
14     String telefone;
15 }
16
```

# Exemplo

- Para definir e obter os valores das variáveis do Objeto temos que utilizar a palavra-chave **public**.

# Exemplo



# Objeto

# Objeto

- Na classe principal, dentro do void main, instanciaremos um Objeto da classe Estudante.

Nomedaclasse **objeto** = new Nomedaclasse();



Não precisa ser "objeto", pode ter outros nomes

# Objeto

- Lembrar que se a Classe estiver em outro pacote, deve fazer a “importação”.
- Clicar na lâmpada no lado esquerdo e escolher “adicionar importação da Classe...”



# Exemplo

The screenshot displays an IDE interface. On the left, a project explorer shows a project named 'Exemplo1' containing a package '<pacote default>' with files 'Estudantes.java', 'Main.java', and 'Pessoas.java'. Below this, a 'main - Navegador' window shows the 'Main' class with a 'main(String[] args)' method. The main editor window, titled 'Código-Fonte', shows the following Java code:

```
4  /*
5
6  /**
7   *
8   * @author ADM004
9   */
10 public class Main {
11
12     /**
13      * @param args the command line arguments
14      */
15     public static void main(String[] args) {
16
17         Estudantes aluno1 = new Estudantes();
18
19     }
20 }
21
```

# Exemplo














- Após instanciar a Classe, definir o conteúdo/valores de cada um dos atributos.
- Para definir o valor de um objeto, deve-se usar o ponto (.) :  
`nomeDoObjeto.nomeDoAtributo = valor`


# Exemplo

Estud

aluno1.

}

	<b>datanascimento</b>	String
	<b>idade</b>	int
	<b>nome</b>	String
	<b>telefone</b>	String
	<b>equals (Object o)</b>	boolean
	<b>getClass ()</b>	Class<?>
	<b>hashCode ()</b>	int
	<b>notify ()</b>	void
	<b>notifyAll ()</b>	void
	<b>toString ()</b>	String
	<b>wait ()</b>	void
	<b>wait (long l)</b>	void
	<b>wait (long l, int i)</b>	void

 main >

# Exemplo

```
public static void main(String[] args) {  
  
    Estudantes aluno1 = new Estudantes();  
    aluno1.nome = "Marcelo";  
    aluno1.idade = 30;  
    aluno1.datanascimento = "27/05/1984";  
    aluno1.telefone = "3242-2821";  
}  
}
```

# Exemplo

- Para mostrar os valores atribuídos utilizaremos o:  
`System.out.println();`
- Para rodar o programa, usa-se o comando “F6”.

# Exemplo

```
Estudantes aluno1 = new Estudantes();  
aluno1.nome = "Marcelo";  
aluno1.idade = 30;  
aluno1.datanascimento = "27/05/1984";  
aluno1.telefone = "3242-2821";
```

```
System.out.println("O nome de aluno é "+aluno1.nome);  
System.out.println("A idade de aluno é "+aluno1.idade);  
System.out.println("A data de nascimento de aluno é "+aluno1.datanascimento);  
System.out.println("O telefone de aluno é "+aluno1.telefone);
```

# Exercícios

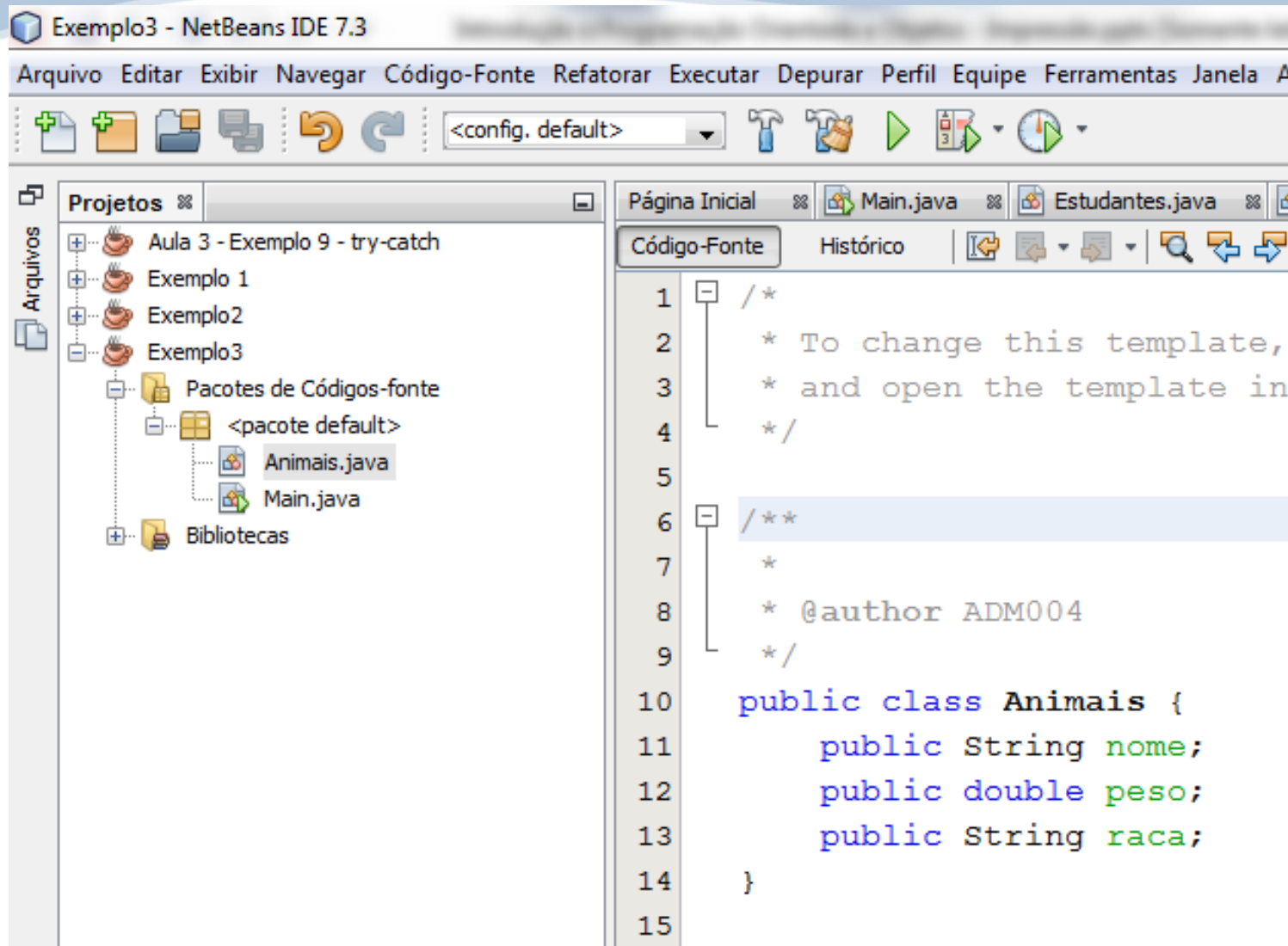
5) Faça uma Classe “Veículos” com os atributos marca, cor e preço. Na classe principal, crie 2 Objetos a partir da Classe “Veículos” chamado: “carro1” e “carro2”.

# Exemplo

- É possível também utilizar condicional e laços.
- Segue um exemplo comparando o peso de dois animais diferentes.



# Exemplo



# Exemplo

```
* @param args the command line arguments
*/
public static void main(String[] args) {
    Animais a1 = new Animais();
    Animais a2 = new Animais();

    a1.peso = 10.78;
    a2.peso = 3.87;

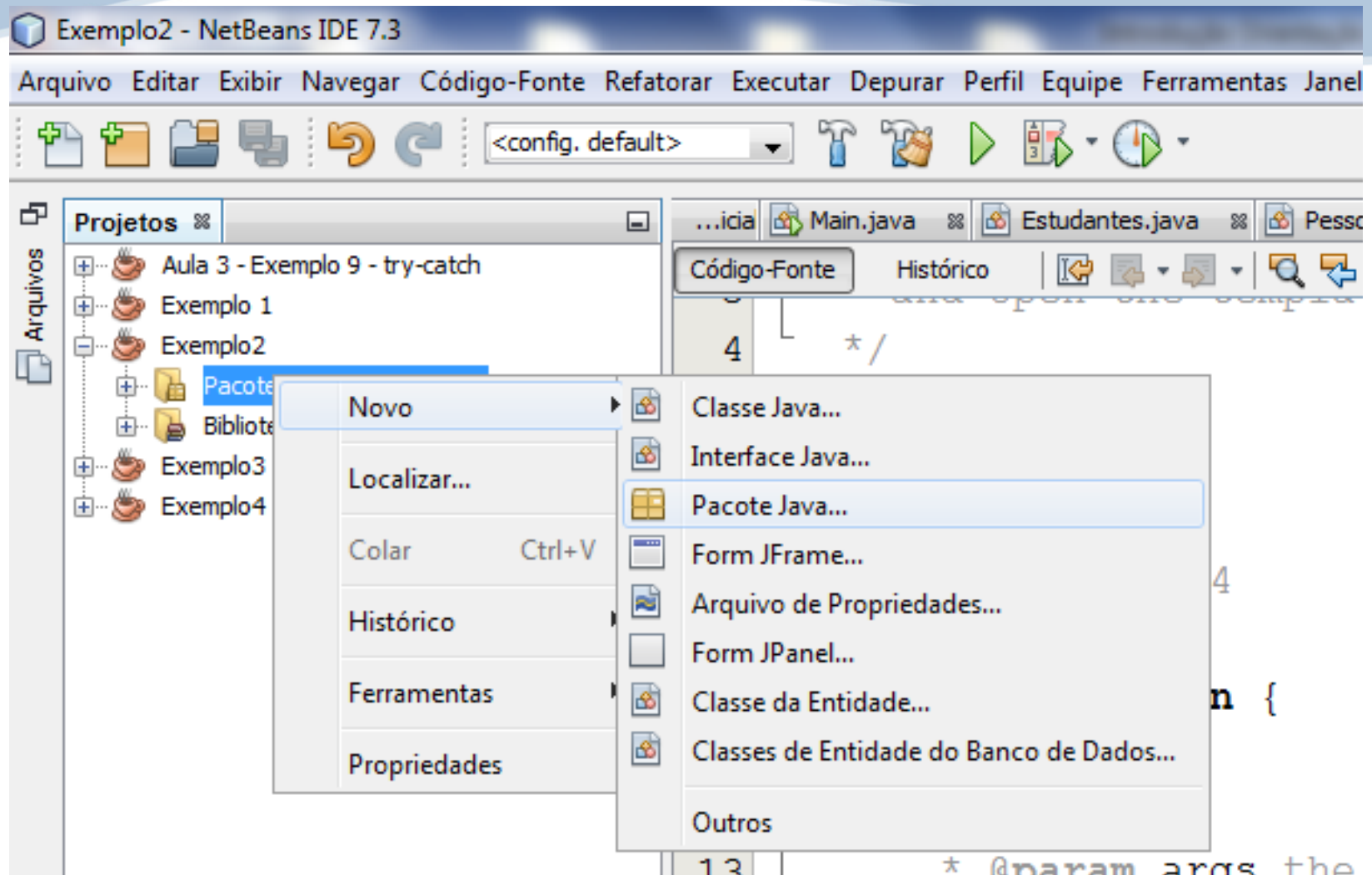
    if(a1.peso>a2.peso)
    {
        System.out.println("a1 mais pesado que a2");
    }
    else
    {
        System.out.println("a2 mais pesado que a1");
    }
}
```

# Pacotes

# Pacotes

- Os pacotes são iguais a pastas.
- Elas servem para organizar as Classes dentro do projeto.
- O nome dos pacotes devem **sempre** ser em letras **minúscula**.
- No lado esquerdo da tela, estão os projetos salvos. Criar o pacote desejado lá. Clicar botão direito -> "Novo" -> "Pacote..." -> Nomeie o pacote.

# Pacotes



# Classe Main

- Para criar a classe principal manualmente podemos utilizar a tecla de atalho:

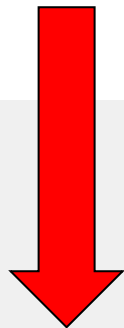
psvm + TAB

- Ficar  escrito da seguinte forma:

```
public static void main(String[] args) {  
    }  
}
```

# Exemplo

- Criar um novo projeto e não marcar a opção “Criar Classe Principal”.



Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).

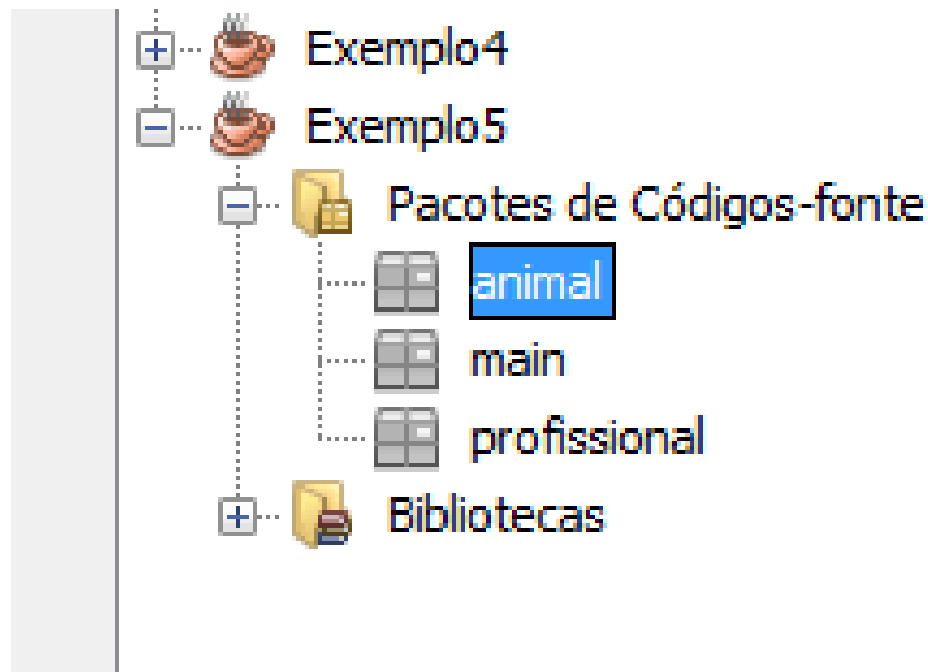
☐

Criar Classe Principal

exemplo5.Exemplo5

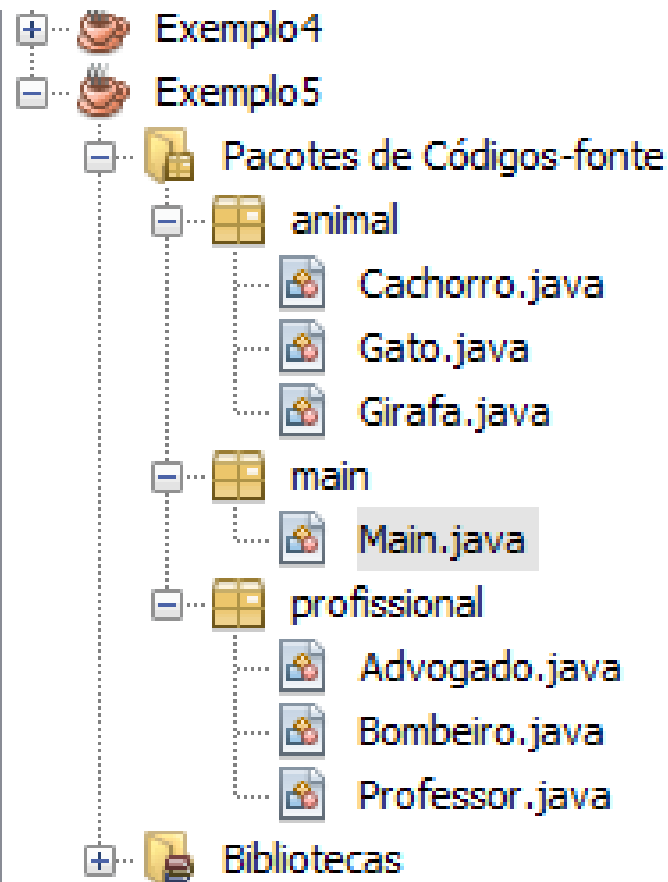
# Exemplo

- Criar os pacotes e depois colocar as classes em cada um dos pacotes.





# Exemplo



# Exemplo

- Não esquecer de colocar o psvm + TAB na classe Main.

```
    *  
    * @author ADM004  
    */  
public class Main {  
    public static void main(String[] args) {  
        |  
    }  
}
```

# Objeto

- Lembrar que se a Classe estiver em outro pacote, deve fazer a “importação”.
- Clicar na lâmpada no lado esquerdo e escolher “adicionar importação da Classe...”

# Exercícios

6) Faça uma Classe “Herois” e uma Classe “Viloes”, dentro do pacote “quadrinhos”, ambas com os atributos nome, superpoder e idade. Na classe principal, crie 1 Objeto de cada Classe.

# Métodos

# Método

- O que faz uma classe Banco?
- Quais são as ações de uma classe Banco?
- Para cada ação de uma classe damos o nome de método.

# Método

- Exemplos de métodos:
  - ✓ sacar uma quantidade x
  - ✓ depositar uma quantidade x
  - ✓ imprimir o nome do dono da conta
  - ✓ devolver o saldo atual
  - ✓ transferir uma quantidade x para uma outra conta e devolver o tipo de conta

# Método

- Primeiramente, devemos analisar o que o método irá retornar.
- O método pode ser do tipo:
  - ✓ int -> retorna um número inteiro
  - ✓ double -> retorna um número real
  - ✓ char -> retorna um caracter
  - ✓ String -> retorna uma palavras ou frases
  - ✓ void -> não tem retorno



# Método

- Outra coisa que devemos analisar é se o método possui ou não argumentos.
- Por exemplo, para calcular a área do quadrado é preciso saber o valor do lado do quadrado, ou seja precisa de 1 argumento do tipo double.
- Outro exemplo, para calcular a soma de dois números é preciso 2 argumentos do tipo double.

# Método

Analise os métodos abaixo, quais são os argumentos e o tipo de cada um:

- andar()
- consultarSaldo()
- media3Notas()
- mostrarTemperatura()
- cadastrarAluno()

# Método

- Para criar o método:
- Deve ser feito dentro da Classe.

```
public tipo nome (argumentos)
{
    comandos;
}
```

# Método

- Para usar um método de uma Classe é preciso criar um objeto dessa Classe.
- Para chamar o método, deve-se usar o ponto (.) :  
`nomeDoObjeto.nomeDoMetodo;`

# Método

- Exemplo 1:
- Quais métodos uma Classe Cachorro possui? E a classe Gato e a classe Passaro?
- Esses métodos possuem retorno?
- Esses métodos precisam de argumento?

# Método

- Métodos da Classe Cachorro:
  - ✓ latir() e morder()
- Métodos da Classe Gato:
  - ✓ miar()
- Métodos da Classe Passaro:
  - ✓ piar() e cantar()

# Método

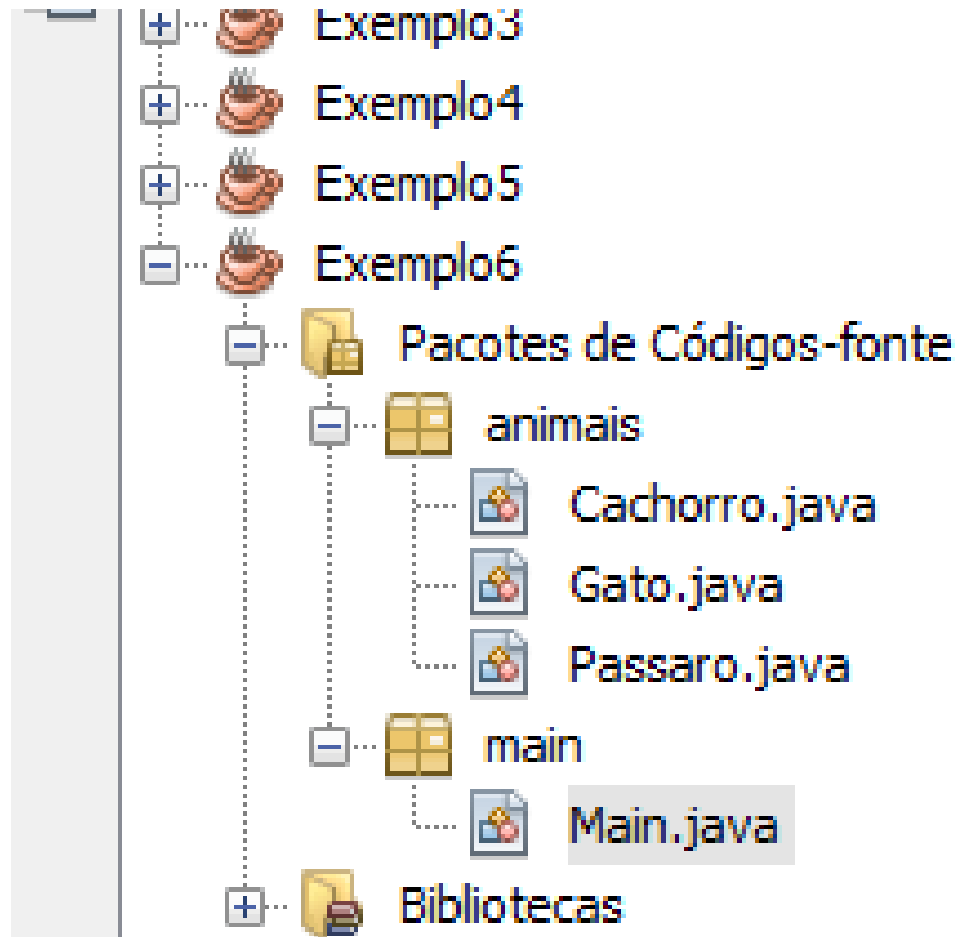
- Esses métodos possuem retorno?  
✓ Não! Então, serão do tipo void.
- Esses métodos precisam de argumento?  
✓ Não! Então, não será preciso mandar valores quando chamar o método.

# Método





# Método



# Método

- Classe Cachorro

```
10  //
11  public class Cachorro {
12      public void latir()
13      {
14          System.out.println("Au au au!");
15      }
16      public void morder()
17      {
18          System.out.println("Nhoc!");
19      }
20
21  }
22
```

# Método

- Classe Main

```
L2  */
L3  public class Main {
L4      public static void main(String[] args) {
L5          Cachorro c1 = new Cachorro();
L6          c1.latir();
L7          c1.morder();
L8      }
L9  }
L20
```

# Exercícios

7) Criar os métodos da Classe Gato e Passaro e chamá-los na classe principal.

8) Criar a Classe Dia com o método bomdia(), a Classe Tarde com o método boatarde() e a Classe Noite com o método boanoite(). Os métodos mostram “Bom dia!”, “Boa tarde!” e “Boa noite!”, respectivamente.

# Método

- Exemplo 2:
- Por exemplo, o método `somar()`, ele deverá retornar um número real, ou seja, o método será do tipo `double`.

# Método

- Outro ponto que devemos analisar é se o método tem ou não argumentos.
- Por exemplo, o método `somar()`, deve receber dois valores do tipo `double` para realizar a soma.
- Então, o método `somar()` deverá ficar da seguinte maneira:
  - ✓ `double somar(double a, double b)`

# Método

Exemplo4 - NetBeans IDE 7.3

Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda

<config. default>

Arquivos

Projetos

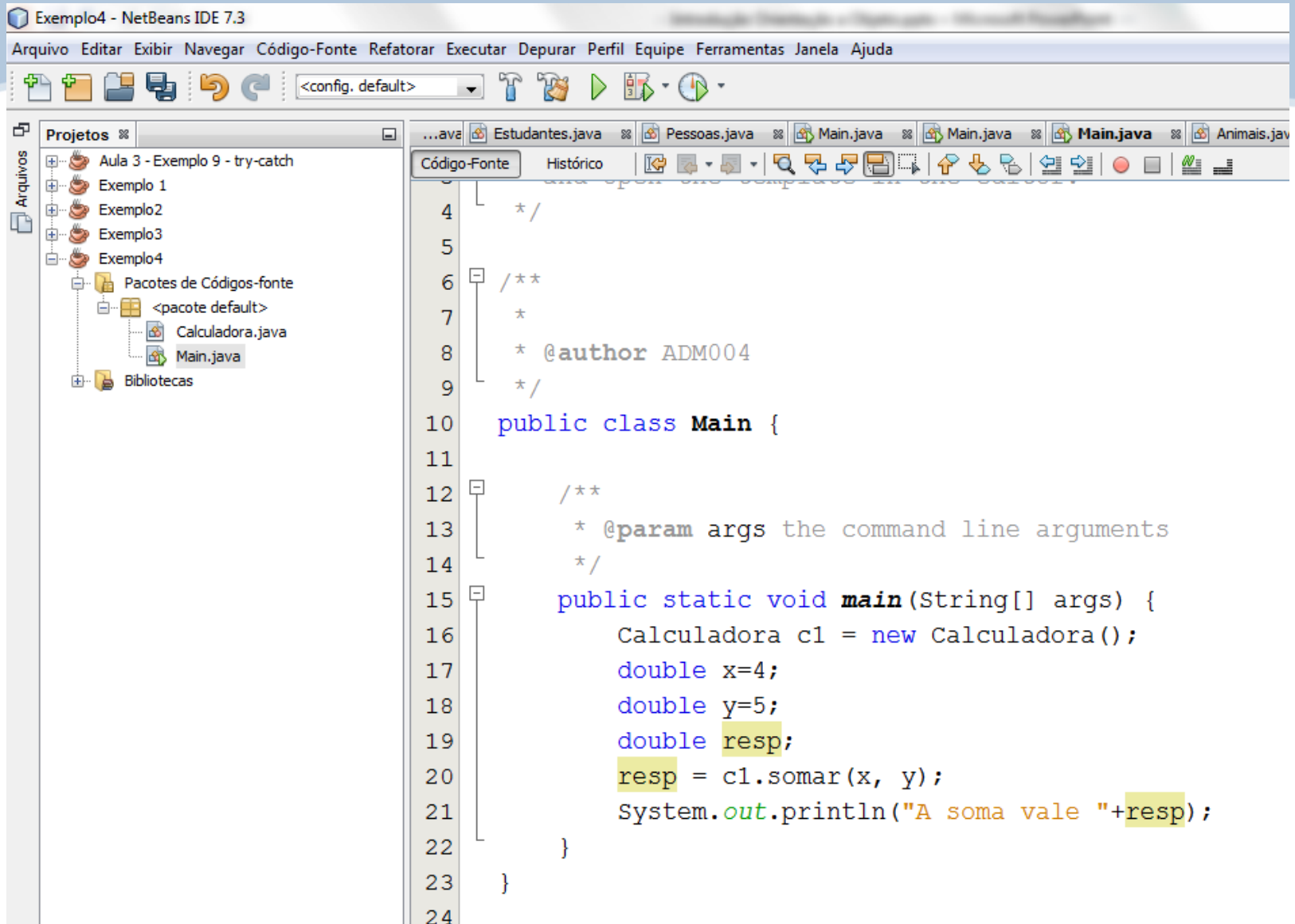
- Aula 3 - Exemplo 9 - try-catch
- Exemplo 1
- Exemplo2
- Exemplo3
- Exemplo4
  - Pacotes de Códigos-fonte
    - <pacote default>
      - Calculadora.java
      - Main.java
  - Bibliotecas

...ava Estudantes.java Pessoas.java Main.java Main.java Main.java

Código-Fonte Histórico

```
1  /*
2   * To change this template, choose Tools | Template
3   * and open the template in the editor.
4   */
5
6  /**
7   *
8   * @author ADM004
9   */
10 public class Calculadora {
11     public double somar(double a, double b)
12     {
13         return a+b;
14     }
15 }
16
```

# Método



Exemplo4 - NetBeans IDE 7.3

Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda

<config. default>

Arquivos

Projetos

- Aula 3 - Exemplo 9 - try-catch
- Exemplo 1
- Exemplo2
- Exemplo3
- Exemplo4
  - Pacotes de Códigos-fonte
    - <pacote default>
      - Calculadora.java
      - Main.java
  - Bibliotecas

```
4  */
5
6  /**
7   *
8   * @author ADM004
9   */
10 public class Main {
11
12     /**
13      * @param args the command line arguments
14      */
15     public static void main(String[] args) {
16         Calculadora c1 = new Calculadora();
17         double x=4;
18         double y=5;
19         double resp;
20         resp = c1.somar(x, y);
21         System.out.println("A soma vale "+resp);
22     }
23 }
24
```



# Classe Scanner

- Para que o usuário escolha os valores dos atributos/variáveis, pode-se usar os métodos da Classe Scanner.
- Criar um objeto no Main:
  - ✓ `Scanner leitor = new Scanner(System.in);`
  - ✓ `a = leitor.nextInt();`

# Exemplo Scanner

```
public class Main {  
    public static void main(String[] args) {  
        Scanner leitor = new Scanner(System.in);  
  
        double valor1;  
        double valor2;  
  
        System.out.println("Digite um número: ");  
        valor1 = leitor.nextDouble();  
        System.out.println("Digite um número: ");  
        valor2 = leitor.nextDouble();  
  
        System.out.println("O valor1 digitado foi "+valor1);  
        System.out.println("O valor2 digitado foi "+valor2);  
    }  
}
```

# Exercícios

9) Crie um pacote main com uma classe Main, e um pacote calculadora com a classe Calculadora. Criar um método que calcula a média de 3 notas.

# Exercícios

10) Faça uma Classe “Verificador” com o método verificaridade() que receba uma idade e não tenha retorno, o método diz se a pessoa é maior ou menor de idade. Na classe principal, crie 1 Objeto a partir da Classe “Verificador” chamado: “v1” e chame o método, fornecendo uma idade.

# Exercícios

11) Criar a classe Menu, Dia, Tarde e Noite. Criar os mesmo métodos do exercício anterior.

Na classe Menu, criar o método mostrar() que não recebe valores nem retorna nada. Dentro do método mostrar(), chamar os métodos das classes Dia, Tarde e Noite.

Na classe Main, chamar apenas o método mostrar().

# Exercícios

12) Crie um pacote main com uma classe Main, e um pacote banco com a Classe Cambio. Peça ao usuário para digitar o valor que possui em carteira de dólar e o valor do câmbio(real para dólar) e retorna o valor convertido em real.

# Exercícios

13) Crie um pacote main com uma classe Main, e um pacote banco com a Classe Cambio. Peça ao usuário para digitar o valor que possui em carteira de real, dólar e o valor do câmbio(real para dólar) e retorna o valor convertido em real.

# Exercícios

14) Crie um pacote main com uma classe Main, e um pacote exemplo com a classe Laço. Criar um método mostrar() na classe Laço que mostrar de 100 até 0.



# Exercícios

15) Crie um pacote main com uma classe Main, e um pacote exemplo com a classe Laço. Crie um método mostrar2() que receba 2 argumentos. Sendo "a" e "b" dois números inteiros, mostrar de "a" até "b".

# Exercícios

16) Crie um pacote main com uma classe Main, e um pacote calculadora com a classe Quadrado. Através de um looping e de um método calcular, mostre em tela os quadrados dos números de 1 a 20.

# Exercícios

17) Criar a Classe Bomba que contém os atributos: modelo, nivel(int), peso e classificacao(String).

Criar 1 objeto na Classe Main e verificar qual classificação ela se encaixa.

Se a bomba for de nível 5 e peso maior que 100, mostrar a mensagem "Super bomba". Se a bomba for de nível entre 2 e 4 e peso maior que 80, mostrar a mensagem "Bomba perigosa". Se não acontecer nenhum dos casos, mostrar a mensagem "Bomba".

Dica: usar && para realizar múltiplas comparações no mesmo if.

# Resumo

- Paradigma Orientado por Objetos – POO.
- Análise de sistemas orientados a objetos.
- Classes.
- Objetos e instanciação.
- Atributos e métodos.