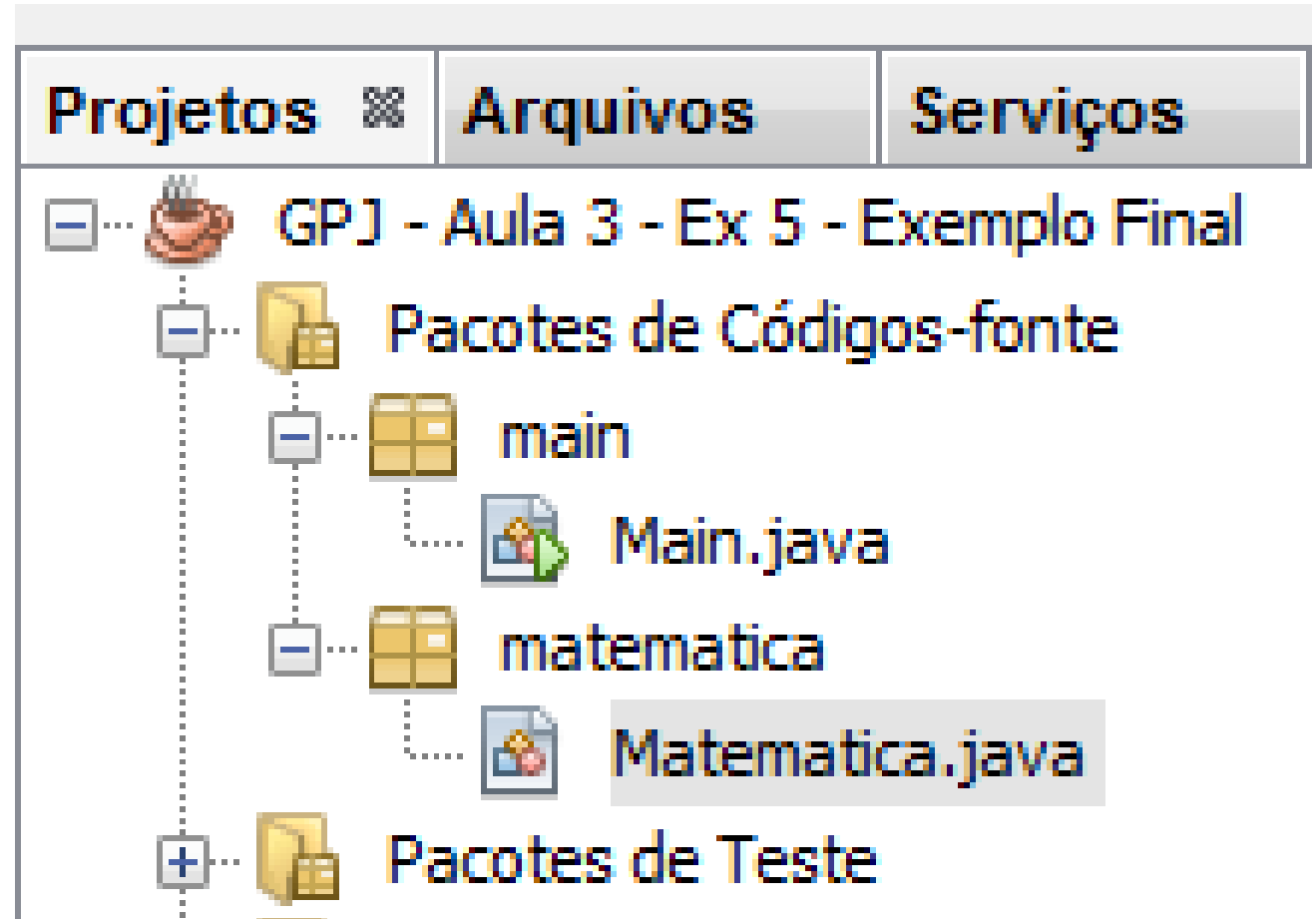


## 10. Final

- Palavra chave final usada em atributo não permite alterar seu valor, transformando-a em uma constante.
- Utilizando final em método não permite que ele seja sobreposto.
- Se não permite que altere seu valor, então nunca se usa final com abstract.

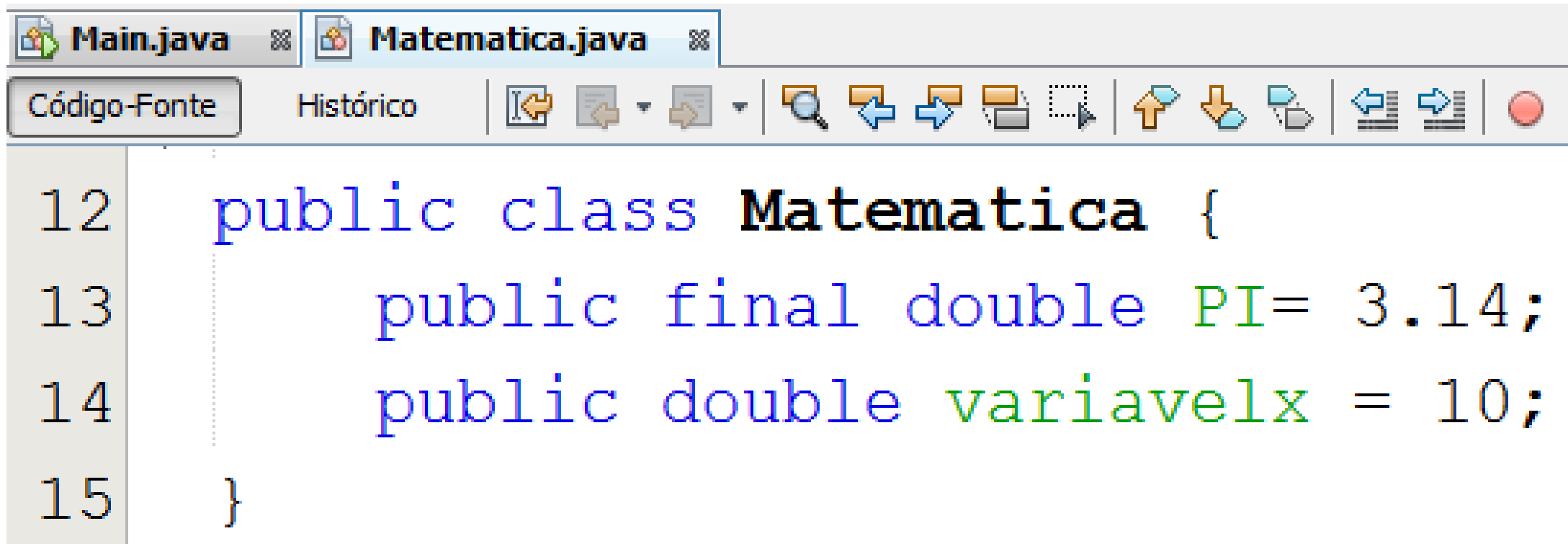
# 10. Final

**Exemplo de Final:** Criar o pacote **matematica** com a classe **Matematica** e o pacote **main** e a classe **Main**.



## 10. Final

Na classe **Matematica** criar um atributo final com o valor de **PI** e outro **variavelx** com valor de 10.



The screenshot shows an IDE window with two tabs: 'Main.java' and 'Matematica.java'. The 'Matematica.java' tab is active, showing the source code. The code defines a public class 'Matematica' with two public attributes: a final double 'PI' with a value of 3.14, and a double 'variavelx' with a value of 10. The code is as follows:

```
12 public class Matematica {  
13     public final double PI= 3.14;  
14     public double variavelx = 10;  
15 }
```

## 10. Final

Na classe **Main** instanciar os objetos m1 e m2 da classe **Matematica**.

The image shows a screenshot of a Java IDE. At the top, there are two tabs: 'Main.java' and 'Matematica.java'. Below the tabs is a toolbar with various icons for editing and navigating code. The main area displays the source code for 'Main.java'. The code is as follows:

```
14 public class Main {  
15     public static void main(String[] args) {  
16         Matematica m1 = new Matematica();  
17         Matematica m2 = new Matematica();  
    }
```

The code is color-coded: 'public' is blue, 'class' is blue, 'Main' is black, '{' is blue, 'public' is blue, 'static' is blue, 'void' is blue, '*main*' is black, '(String[] args)' is black, '{' is blue, 'Matematica' is black, 'm1' is black, '=' is black, 'new' is blue, 'Matematica()' is black, ';' is black, 'Matematica' is black, 'm2' is black, '=' is black, 'new' is blue, 'Matematica()' is black, ';' is black, and '}' is blue.

## 10. Final

Mostrar os valores dos atributos e em seguida alterar o valor atributo **PI** e do atributo **variavelx**.

```
19      System.out.println("Valor de Pi: "+m1.PI);  
20      System.out.println("Valore de Variavelx: "+m1.variavelx);
```

Repare que não permite alterar valor do atributo **PI**

```
22      //m1.PI = 80; //Não aceita mudar o valor de PI, pois é do tipo final  
23      m1.variavelx = 90;  
24      System.out.println("Valor da Variavelx: "+m1.variavelx);  
25      System.out.println("");  
26      System.out.println("Valor da Variavelx m2: "+m2.variavelx);  
27  }
```

# 10. Final

Repare na saída que mesmo alterando o atributo em um objeto, o outro não sofre alteração, pois os atributos são universais, então cada objeto tem seu próprio atributo

∴ Saída - GPJ - Aula 3 - Ex 5 - Exemplo Final (run)



run:



Valor de Pi: 3.14



Valore de Variavelx: 10.0



Valor da Variavelx: 90.0

Valor da Variavelx m2: 10.0

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

# Métodos Equals

- Compara 2 objetos e retorna um boolean
- Boolean = True/False

# 11. Static

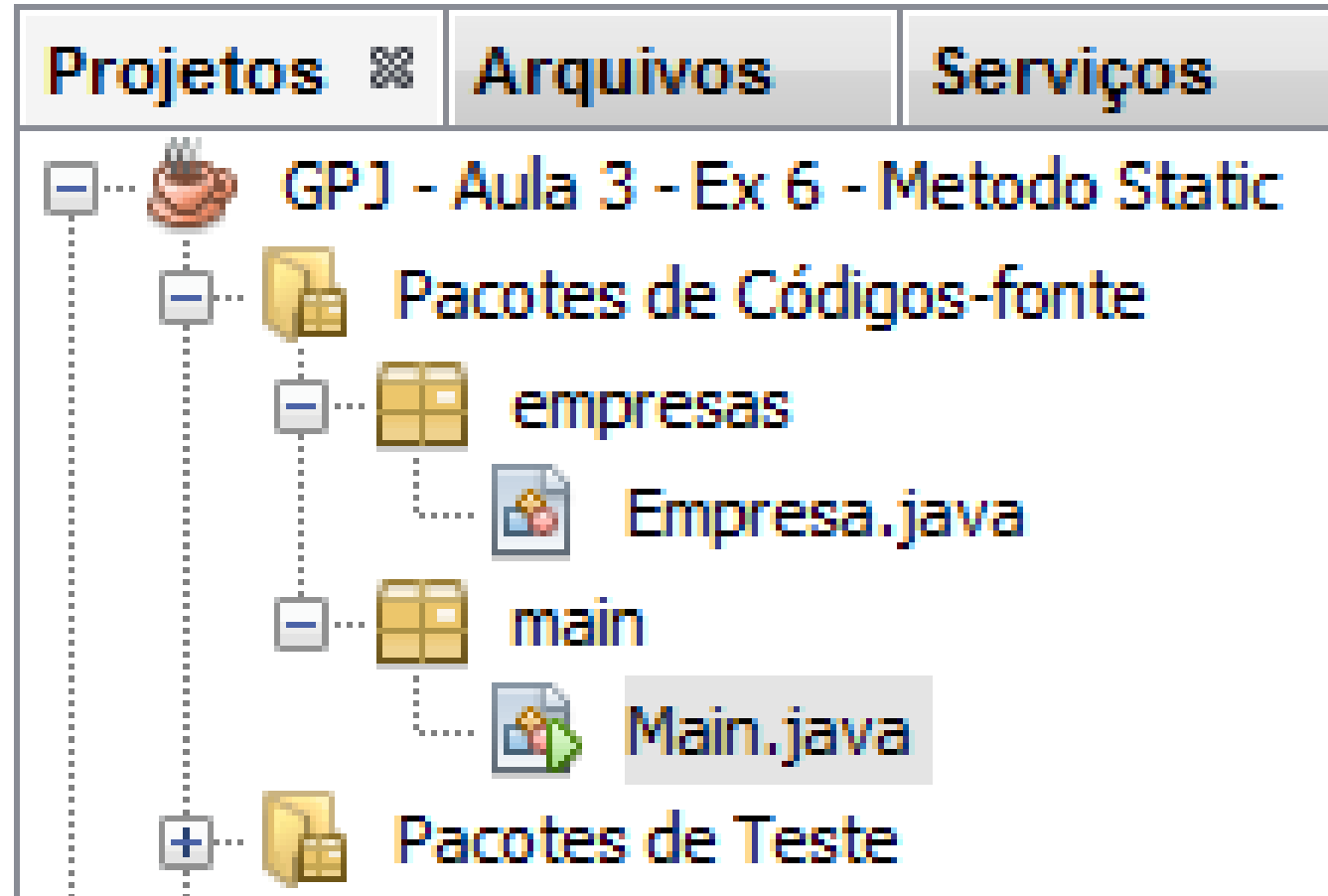
- Palavra chave Static tem uma função para método e outra para atributo.
- Quando usado em método não é necessário instanciar o objeto.
- A maioria das classes prontas usam métodos e atributos do tipo static.



# 11. Static

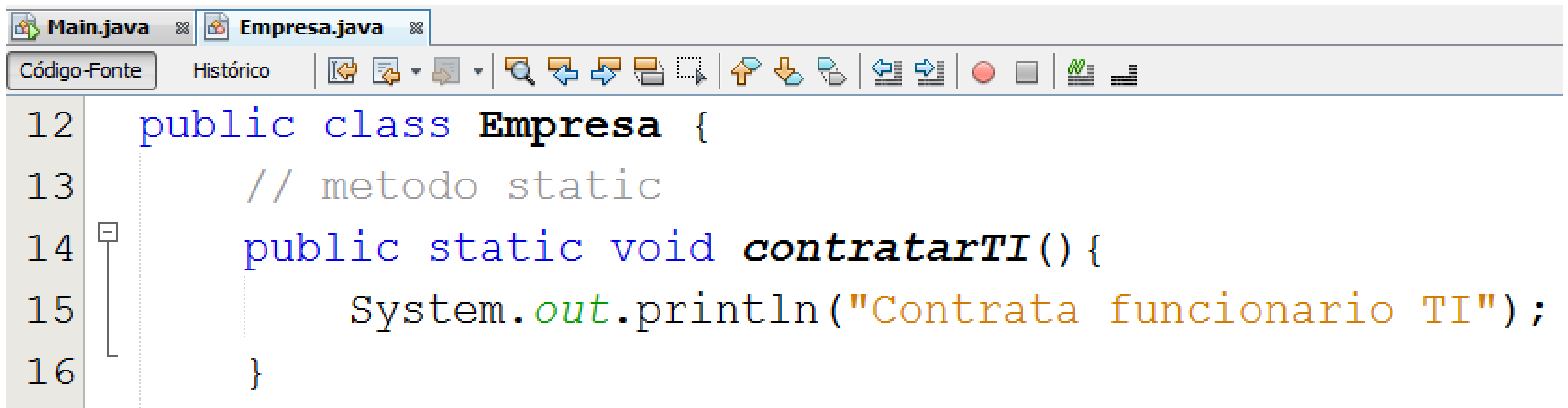
## Exemplo de Método Static:

Criar o pacote **empresas** com a classe **Empresas** e o pacote **main** e a classe **Main**.



# 11. Static

Na classe **Empresa** criar o método **contratarTi** do tipo **static**.



The screenshot shows an IDE window with two tabs: 'Main.java' and 'Empresa.java'. The 'Código-Fonte' (Source Code) tab is active. The code in 'Empresa.java' is as follows:

```
12 public class Empresa {  
13     // metodo static  
14     public static void contratarTI() {  
15         System.out.println("Contrata funcionario TI");  
16     }
```

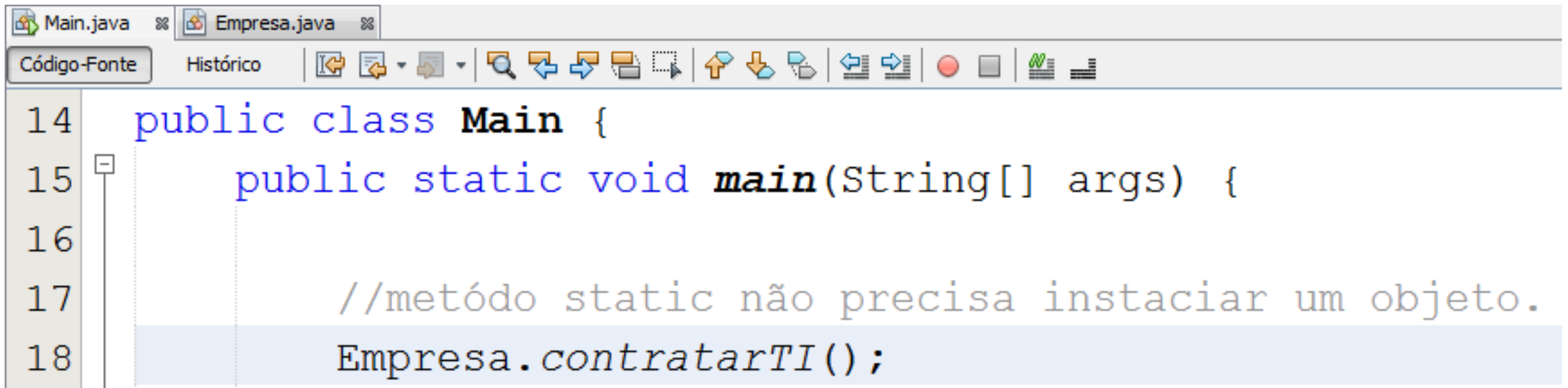
# 11. Static

Criar o método não static **contrataADM**.

```
18 //metodo não static
19 public void contrataADM() {
20     System.out.println("Contrara funcionario ADM");
21 }
22 }
```

# 11. Static

Na classe **Main** chamar o método do tipo **static** sem a necessidade de instanciar o objeto.



```
14 public class Main {
15     public static void main(String[] args) {
16
17         //método static não precisa instanciar um objeto.
18         Empresa.contratarTI();
19     }
```

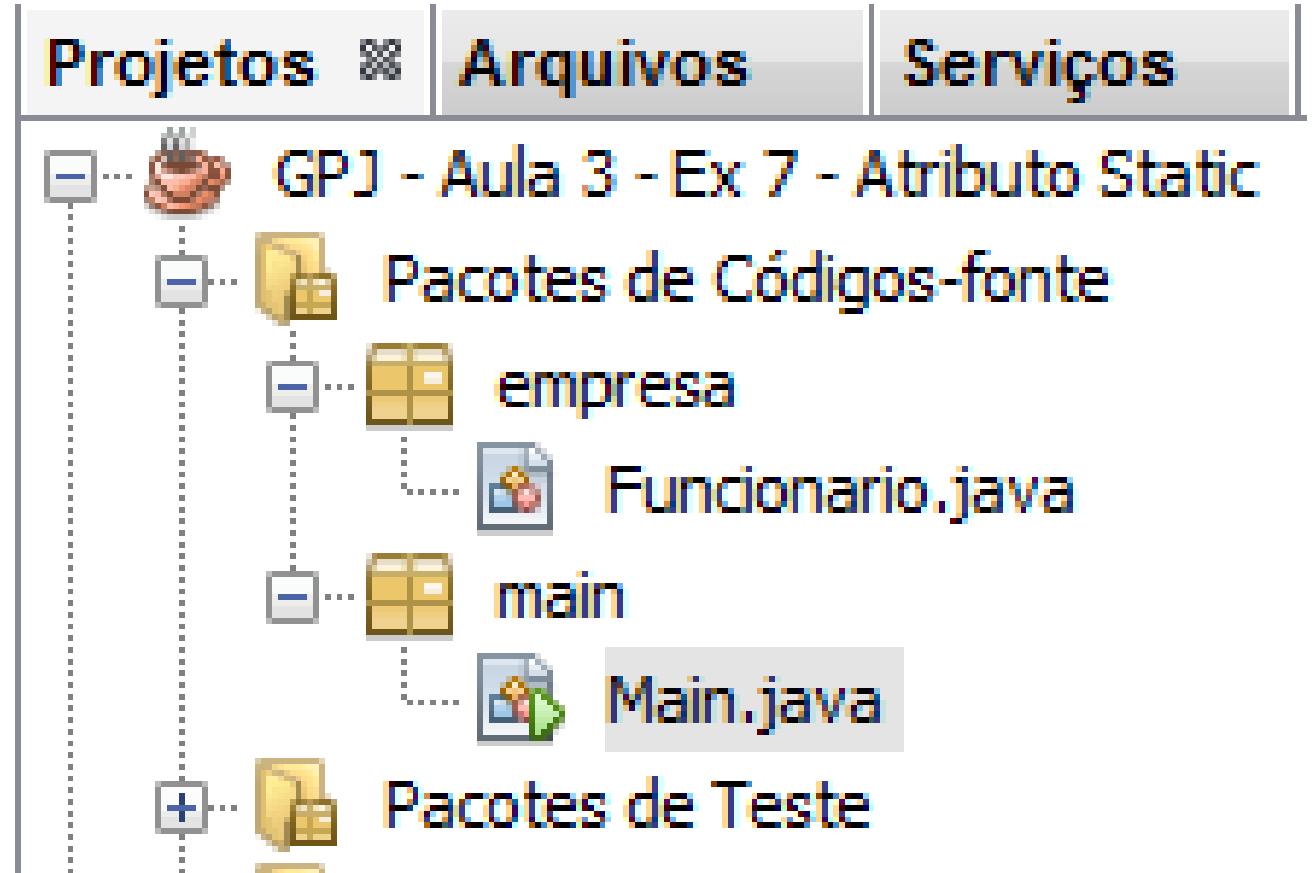
# 11. Static

- Quando usado em atributo todos os objetos compartilham o mesmo atributo.
- Alterando o valor do atributo do tipo static em um objeto, todos os objetos que utilizam o mesmo atributo serão alterados. Pode-se ser comparado com uma variável global, pois no java não existe variável global.

# 11. Static

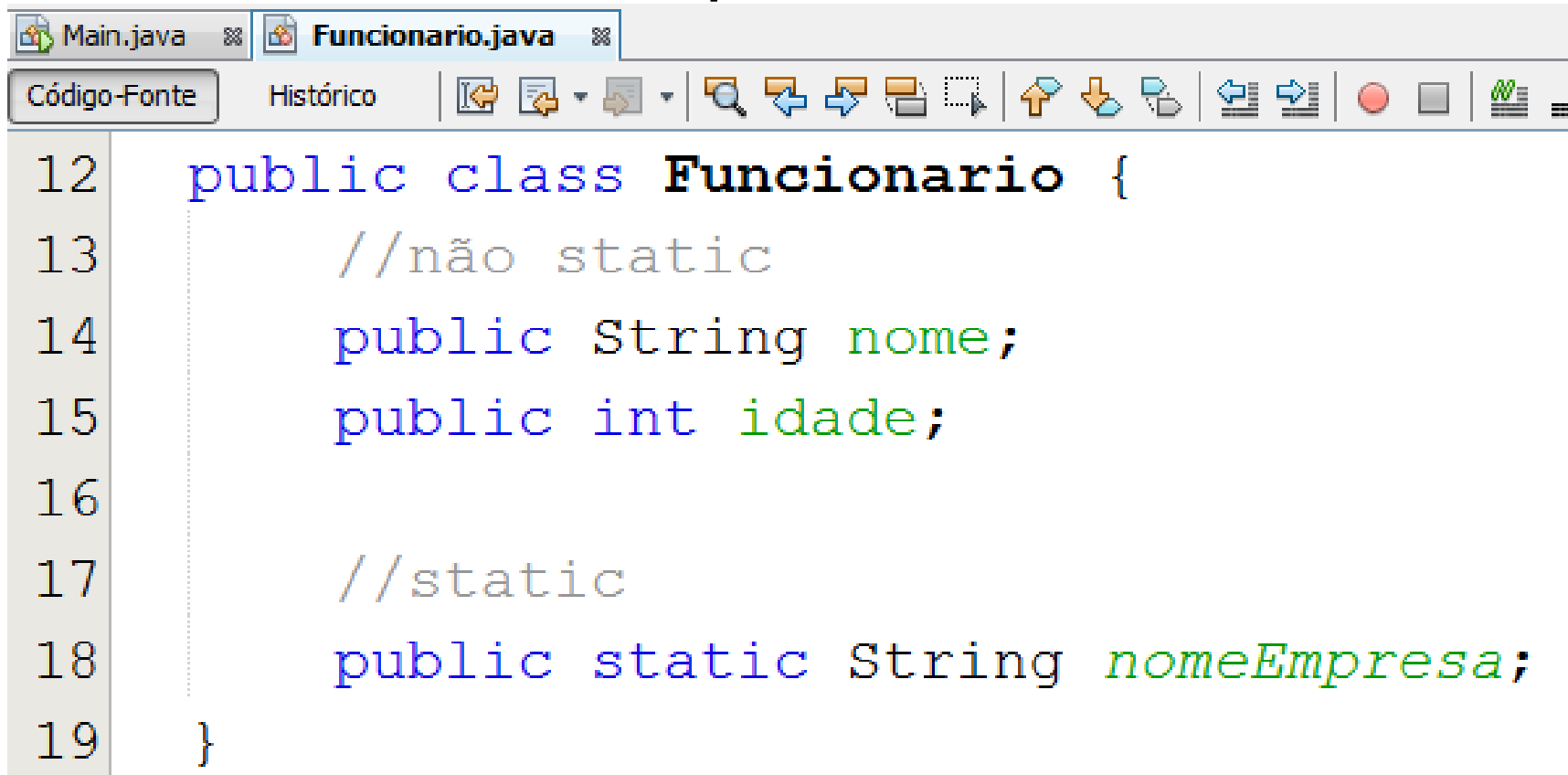
## Exemplo de Atributo Static:

Criar o pacote empresa com a classe Funcionario e o pacote main e a classe Main.



# 11. Static

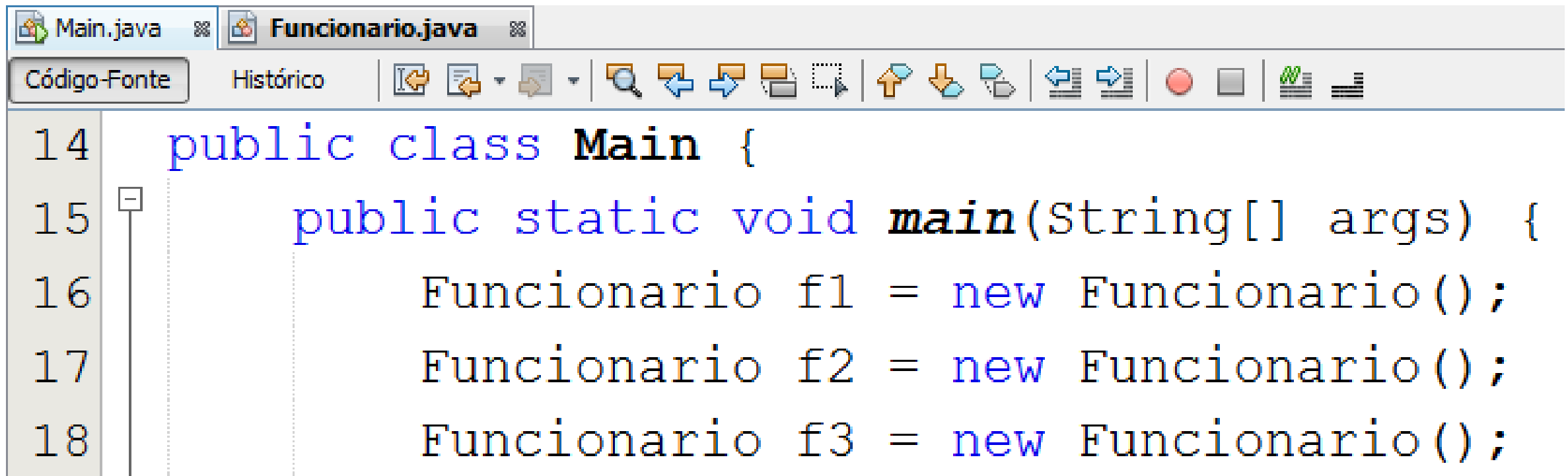
Na classe **Funcionario** criar os atributos **nome** e **idade**, e outro atributo do tipo static chamado **nomeEmpresa**.



```
12 public class Funcionario {
13     //não static
14     public String nome;
15     public int idade;
16
17     //static
18     public static String nomeEmpresa;
19 }
```

# 11. Static

Na classe **Main** instanciar os objetos **f1**, **f2** e **f3** da classe **Funcionario**.

A screenshot of a Java IDE window. The title bar shows two tabs: 'Main.java' and 'Funcionario.java'. Below the tabs is a toolbar with various icons for code editing and navigation. The main area displays the source code of 'Main.java'. The code is as follows:

```
14 public class Main {  
15     public static void main(String[] args) {  
16         Funcionario f1 = new Funcionario();  
17         Funcionario f2 = new Funcionario();  
18         Funcionario f3 = new Funcionario();
```

The code is color-coded: keywords like 'public', 'class', 'static', 'void', 'new', and 'String' are in blue; the method name '**main**' is in bold black; and variable names and class names are in black. The line numbers 14 through 18 are visible on the left side of the code editor.



# 11. Static

Atribuir os nomes, idades e empresas diferentes para cada objeto.

20

21

22

f1.nome = "Rafael"; 24

f2.nome = "Pedro"; 25

f3.nome = "Igor"; 26

f1.idade = 17;

f2.idade = 50;

f3.idade = 42;



f1.nomeEmpresa = "Empresa X";

f2.nomeEmpresa = "Empresa Y";

f3.nomeEmpresa = "Empresa Z";

# 11. Static

Pedir para exibir na tela o nome, idade e Nome da empresa de cada objeto instanciado.

```
32      System.out.println("Nome f1 "+f1.nome);
33      System.out.println("Idade f1 "+f1.idade);
34      System.out.println("Nome Empresa "+f1.nomeEmpresa);
35      System.out.println("Nome f2 "+f2.nome);
36      System.out.println("Idade f2 "+f2.idade);
37      System.out.println("Nome Empresa "+f2.nomeEmpresa);
38      System.out.println("Nome f3 "+f3.nome);
39      System.out.println("Idade f3 "+f3.idade);
40      System.out.println("Nome Empresa "+f3.nomeEmpresa);
```

# 11. Static

Observe o resultado:

↳ Saída - GPJ - Aula 3 - Ex 7 - Atributo Static (run)

```
run:
Nome f1 Rafael
Idade f1 17
Nome Empresa Empresa Z
Nome f2 Pedro
Idade f2 50
Nome Empresa Empresa Z
Nome f3 Igor
Idade f3 42
Nome Empresa Empresa Z
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Perceba que o nome da empresa de todos os objetos estão iguais ao do último objeto que foi atribuído esse nome.

## 12. Thread

- Todo programador está familiarizado com a programação sequencial, pois esta é a forma de programação mais comum até o momento. Programas do tipo “Hello World”, que ordenam uma lista de nome ou que geram e imprimem uma lista de números primos, etc são programas tipicamente sequenciais, no qual cada um possui : seu início, sequência de execução e fim.

## 12. Thread

- Há situações que precisamos executar duas coisas ao mesmo tempo.

Exemplo: **Exibir uma barra de tarefas enquanto um relatório é carregado**

## 12. Thread

### Outros exemplos :

- **Navegador**: Efetuar diversos downloads e ao mesmo tempo gerenciar as diferentes velocidades dos servidores, enquanto se navega normalmente.
- **Editor de texto**: Salvamento periódico

## 12. Thread

- Em Java, quando falamos de processamento paralelo, falamos de **Threads**

**Thread** = Novo processo independente! =  
Processamento paralelo

## 12. Thread

- Existem duas maneiras de se utilizar Thread:
- Uma das maneiras é com herança.

- Exemplo:

Para criar = `public class Nome_classe extends Thread {`

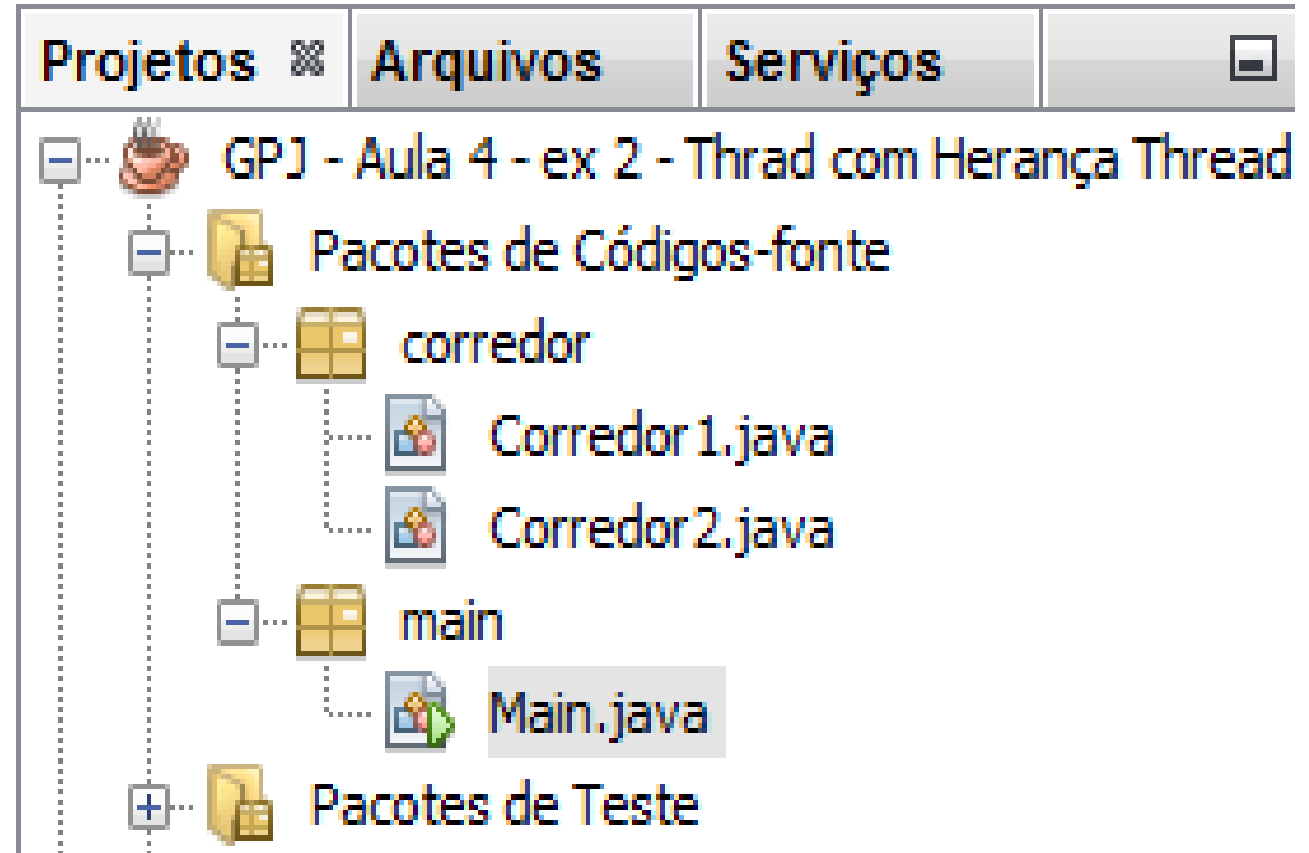
Para iniciar = instanciar o objeto e usar

**`objeto.start ( );`**



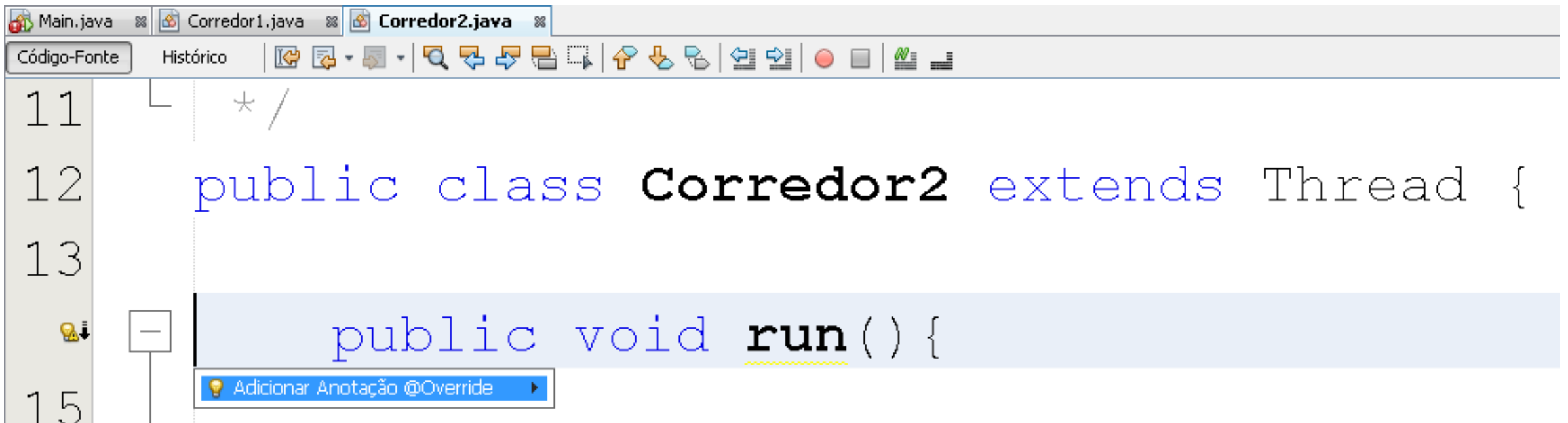
# 12. Thread

Exemplo de Thread com Herança: Criar o pacote **main** com a classe **Main** e o pacote **corredor** com as classes **Corredor1** e **Corredor2**



## 12. Thread

Na classe **Corredor2** colocar como filha da classe Thread, criar o método **run ( )** e adicionar Anotação **@Override**



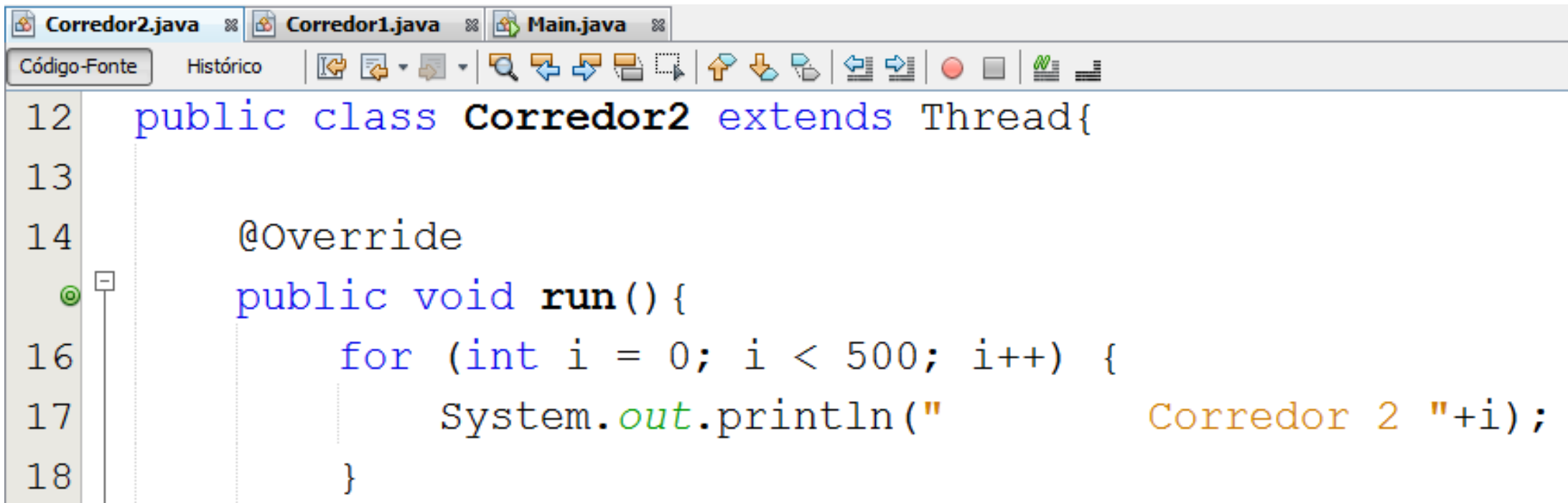
```
11  * /
12  public class Corredor2 extends Thread {
13
14      public void run() {
15
```

Adicionar Anotação @Override

## 12. Thread

Após adicionar @Override, criar um laço para que seja executado 500 vezes a palavra corredor 2 e o valor de i.

Tecla de atalho → for + Tab.

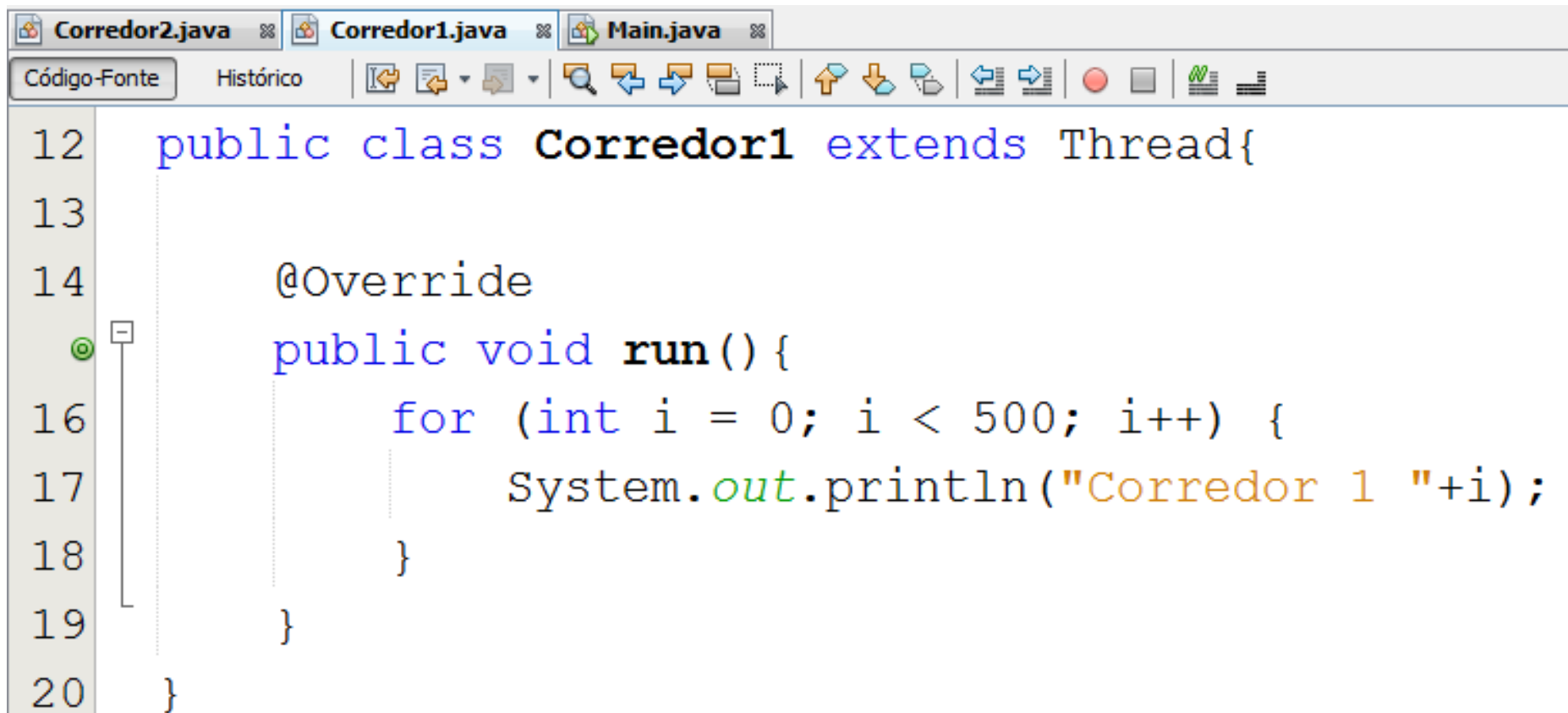
A screenshot of a Java IDE window. The title bar shows three open files: Corredor2.java, Corredor1.java, and Main.java. The 'Código-Fonte' (Source Code) tab is active. The code editor displays the following Java code:

```
12 public class Corredor2 extends Thread{
13
14     @Override
15     public void run() {
16         for (int i = 0; i < 500; i++) {
17             System.out.println("        Corredor 2 "+i);
18         }
```

The code is color-coded: keywords like 'public', 'class', 'extends', 'void', 'run', 'for', and 'int' are in blue; the variable 'i' is in blue; the string 'Corredor 2' is in orange; and the variable 'i' in the string is in blue. A green cursor is positioned at the start of line 15, and a small square icon is visible next to it.

## 12. Thread

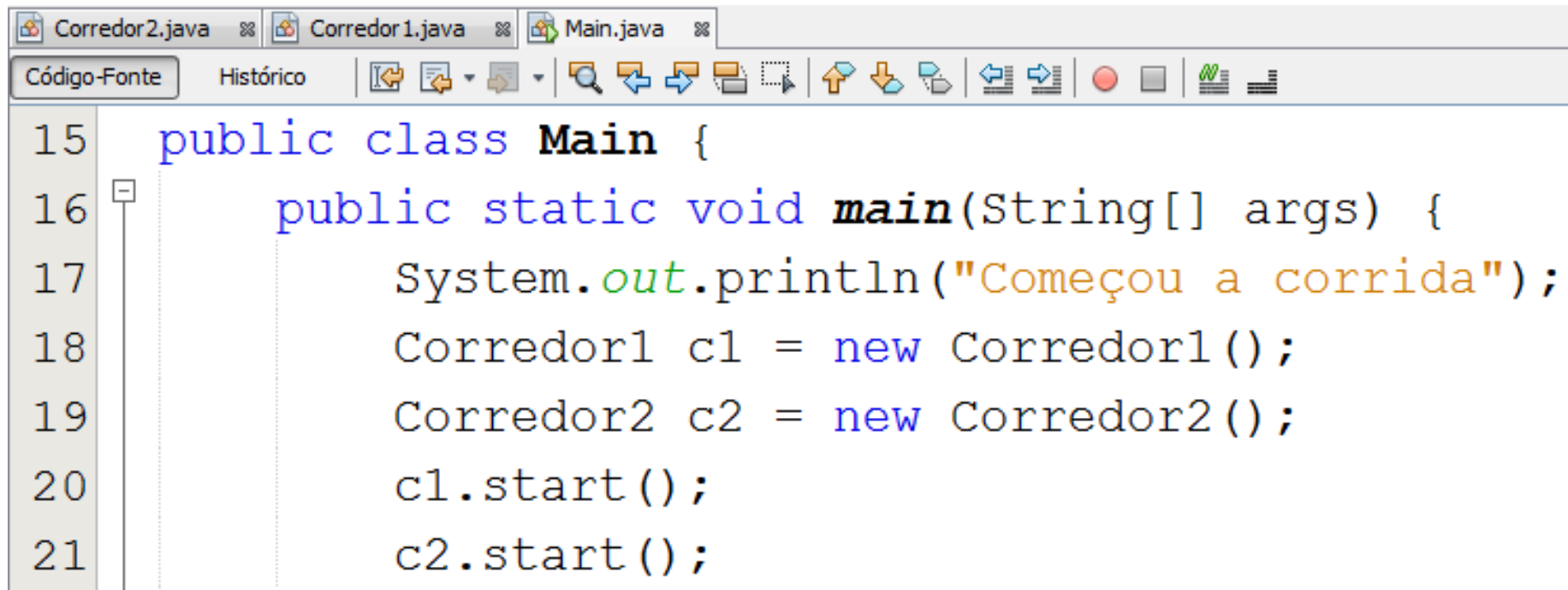
Na classe **Corredor1**, fazer os mesmos procedimentos da classe **Corredor2**.



```
Corredor2.java x Corredor1.java x Main.java x
Código-Fonte Histórico
12 public class Corredor1 extends Thread{
13
14     @Override
15     public void run() {
16         for (int i = 0; i < 500; i++) {
17             System.out.println("Corredor 1 "+i);
18         }
19     }
20 }
```

## 12. Thread

Na classe **Main** instanciar os objetos das classes **Corredor1** e **Corredor2** e chamar o método **start( )** para as classes



```
Corredor2.java  Corredor1.java  Main.java
Código-Fonte  Histórico
15  public class Main {
16      public static void main(String[] args) {
17          System.out.println("Começou a corrida");
18          Corredor1 c1 = new Corredor1();
19          Corredor2 c2 = new Corredor2();
20          c1.start();
21          c2.start();
```

## 12. Thread

Repare que na saída o processo dos métodos ocorrem em paralelo, não importando qual objeto é iniciado primeiro.

```
Saída - GPJ - Aula 4 - ex 2 - Thrad com Herança
Corredor 1 103
Corredor 1 104
    Corredor 2 0
Corredor 1 105
    Corredor 2 1
    Corredor 2 2
    Corredor 2 3
    Corredor 2 4
    Corredor 2 5
Corredor 1 106
Corredor 1 107
Corredor 1 108
```

## 12. Thread

- A outra maneira é com a interface Runnable.

- Exemplo:

Para criar = `public class Lebre implements Runnable{`

Criar a Sobreposição do método **run()**

Para usar = Instanciar o objeto da classe

`Thread thread = new Thread(objeto);`

**`thread.start ( );`**

## 12. Thread



**Exemplo Thread com Interface:** Criar o pacote **main** com a classe **Main** e o pacote **animais** com as classes **tartaruga** e **lebre**.



## 12. Thread

Na classe **Lebre** chamar a interface Runnable com a palavra implements e Implementar Todos os Métodos Abstratos

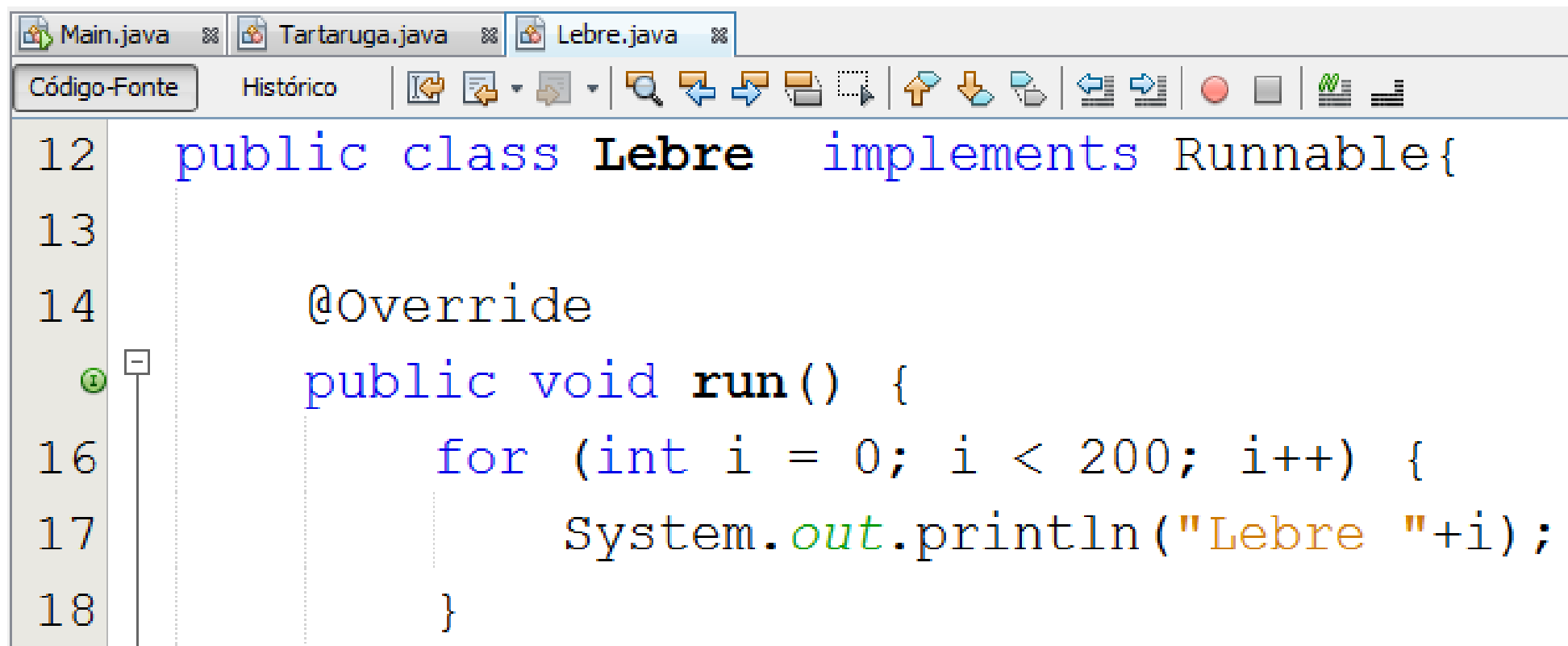
```
11  */
12
13  public class Lebre implements Runnable {
14
15  }
```



- Implementar Todos os Métodos Abstratos
- Tornar Lebre classe abstrata

## 12. Thread

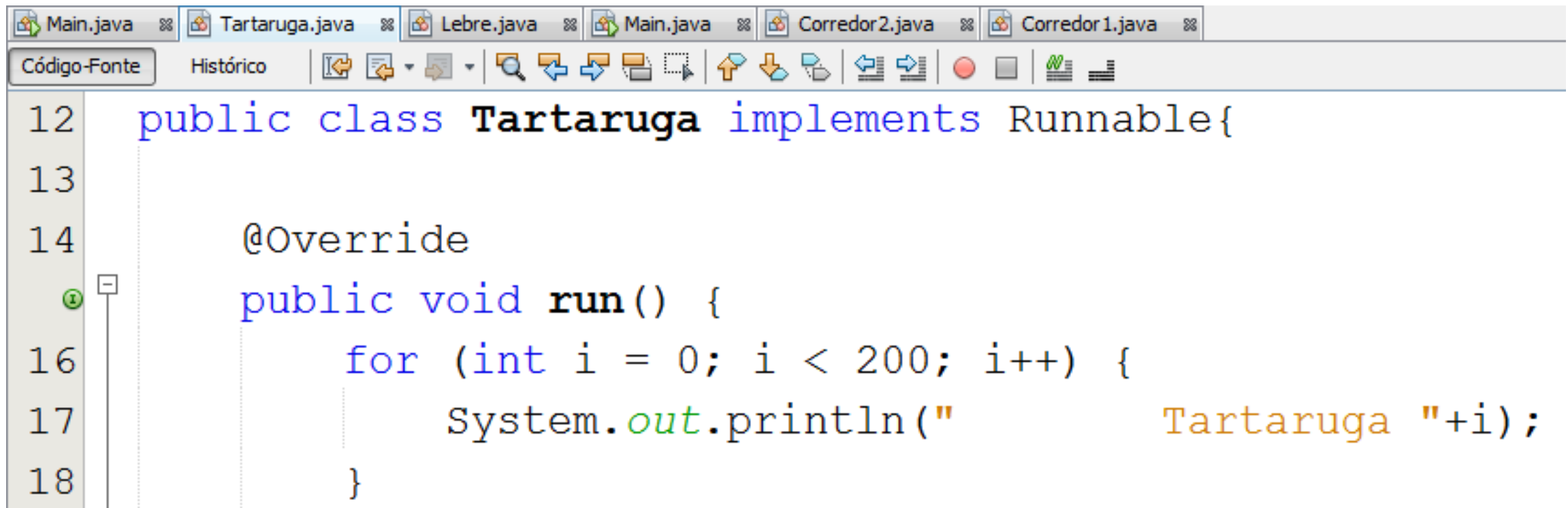
Criar um laço para que mostre na tela a palavra Lebre e a linha em que está.

A screenshot of a Java IDE window. The title bar shows three open files: Main.java, Tartaruga.java, and Lebre.java. The 'Lebre.java' file is active. The interface includes a menu bar with 'Código-Fonte' and 'Histórico', and a toolbar with various editing icons. The code editor displays the following Java code:

```
12 public class Lebre implements Runnable{
13
14     @Override
15     public void run() {
16         for (int i = 0; i < 200; i++) {
17             System.out.println("Lebre "+i);
18         }
19     }
```

## 12. Thread

Na classe **Tartaruga**, fazer o mesmo processo da classe **lebre**, porém mostrando a palavra **Tartaruga**.

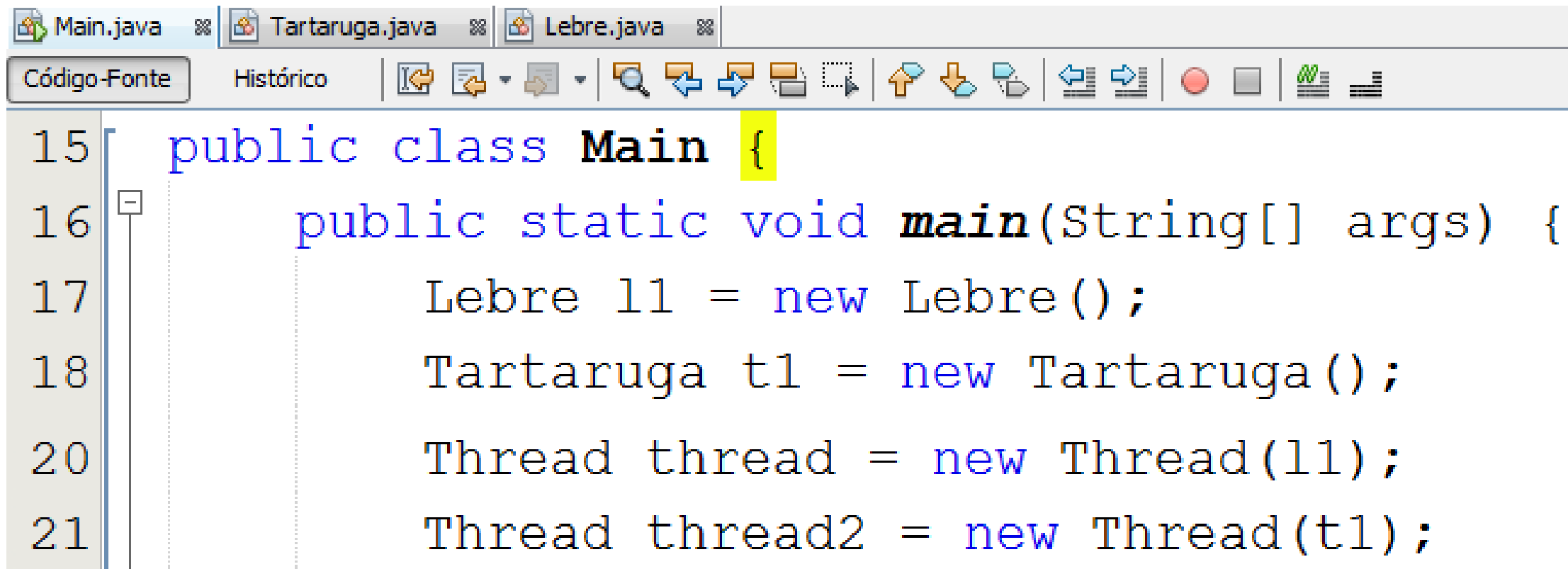


```
12 public class Tartaruga implements Runnable{
13
14     @Override
15     public void run() {
16         for (int i = 0; i < 200; i++) {
17             System.out.println("          Tartaruga "+i);
18         }
19     }
20 }
```

The screenshot shows an IDE window with several tabs: Main.java, Tartaruga.java, Lebre.java, Main.java, Corredor2.java, and Corredor1.java. The 'Código-Fonte' (Source Code) tab is active. The code for the **Tartaruga** class is displayed, showing it implements the **Runnable** interface. The **run()** method contains a loop from 0 to 199, printing "Tartaruga " followed by the loop index **i**.

## 12. Thread

Na classe **Main** instanciar os objetos **Tartaruga** e **Lebre**, e os objetos **thread** e **tread2** da classe **Thread**.



```
15 public class Main {
16     public static void main(String[] args) {
17         Lebre l1 = new Lebre();
18         Tartaruga t1 = new Tartaruga();
20         Thread thread = new Thread(l1);
21         Thread thread2 = new Thread(t1);
```

## 12. Thread

Nos objetos **thread** e **thread2** chamar a função **.start( )**

```
23 System.out.println("Começou a Corrida ");  
24 thread.start();  
25 thread2.start();
```

# 12. Thread

Saída - GPJ - Aula 4 - ex 3 - Thrad com Interface Runnable (run)



Lebre 172



Tartaruga 62



Lebre 173



Lebre 174

Tartaruga 63

Lebre 175

Lebre 176

Tartaruga 64

Lebre 177

Lebre 178

Tartaruga 65

Tartaruga 66

Lebre 179

Tartaruga 67

Note que os processos são executados em paralelo e a ordem fica aleatória na saída, pois não há prioridade na execução do programa.

## 12. Thread

**Exercícios:** Criar um programa com 4 corredores em paralelo.

## 13. Java Lang - pacote java.lang

- É uma API (Biblioteca) do Java que possui muitas classes prontas. Por padrão, quando se instala o Java ela já vem com essa API.
- Algumas classes do Java Lang.
- Classe Wrapper
- Classe Math



## 13. Java Lang - pacote java.lang

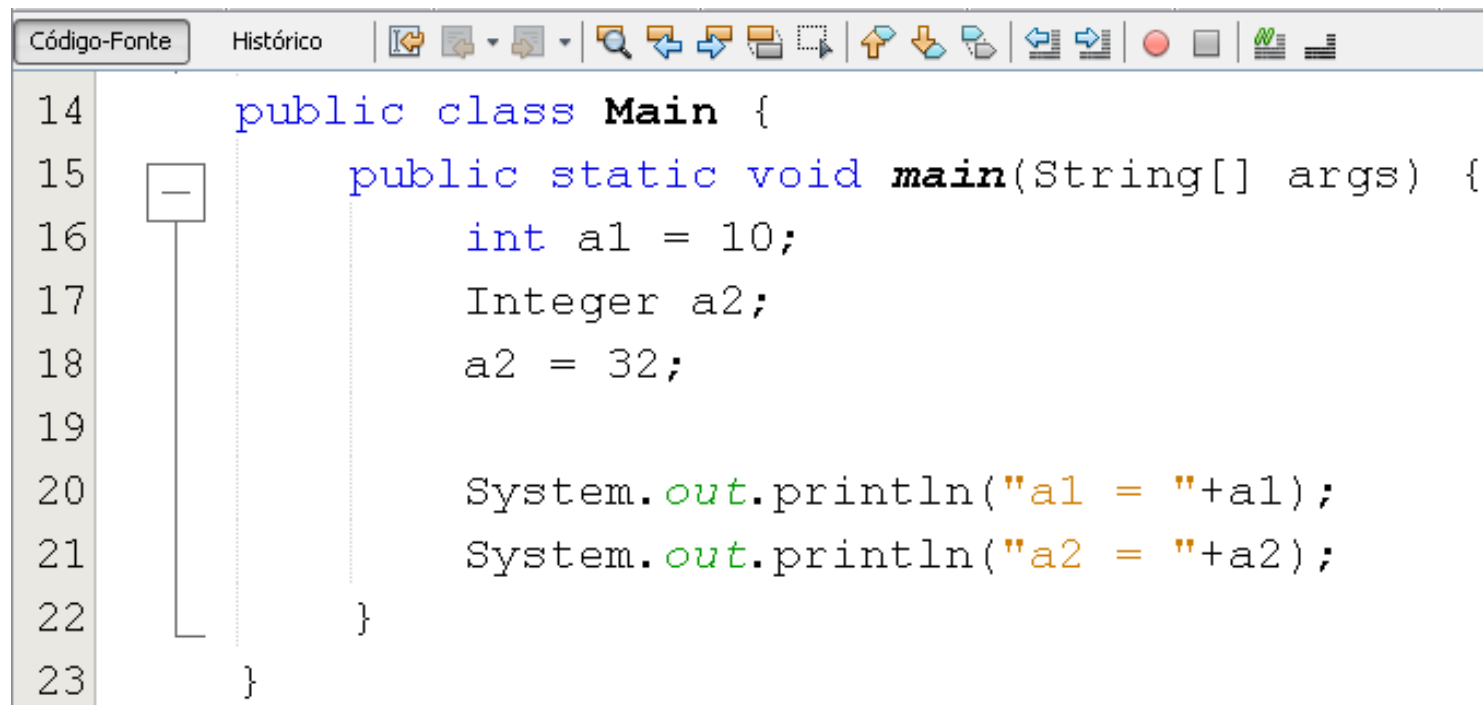
- Os **Wrapper** são classes especiais que possuem métodos capazes de fazer conversões em variáveis primitivas e também de encapsular tipos primitivos.
- Existe classe wrapper para cada tipo primitivo.

## 13. Java Lang - pacote java.lang

- Uma diferença que temos é que os tipos Primitivos possuem métodos diferenciando-os das variáveis que não possuem. Sendo assim, temos que instanciar objeto, porém sem a necessidade de colocar = new Integer
- Exemplos de tipos primitivos:
  - Integer      -Double      -String

## 13. Java Lang - pacote java.lang

- **Exemplo de tipos primitivos:** Criar o pacote **main** e Classe **Main**. Criar a variável inteira **a1** e o tipo primitivo Integer **a2**, mostrar valor de **a1** e **a2**.



The screenshot shows an IDE window with a toolbar at the top containing icons for file operations, search, and execution. Below the toolbar, the code editor displays the following Java code:

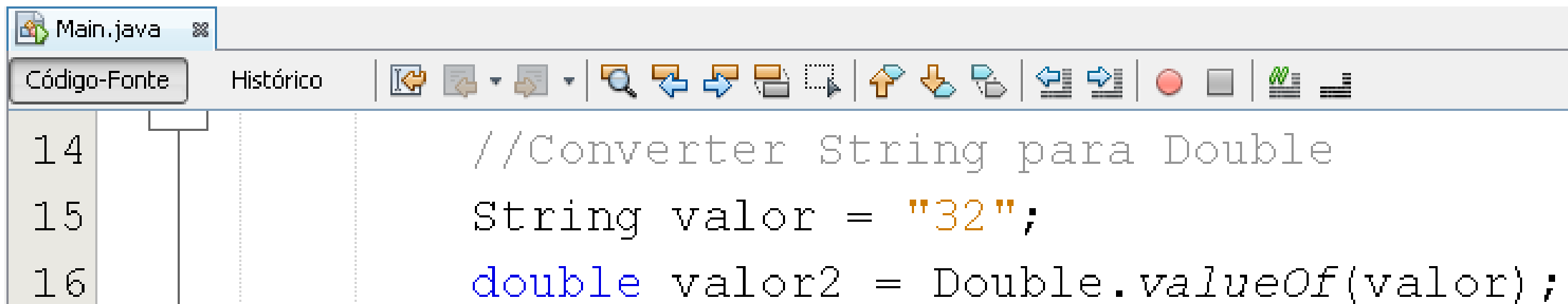
```
14 public class Main {  
15     public static void main(String[] args) {  
16         int a1 = 10;  
17         Integer a2;  
18         a2 = 32;  
19  
20         System.out.println("a1 = "+a1);  
21         System.out.println("a2 = "+a2);  
22     }  
23 }
```

## 13. Java Lang - pacote java.lang

- Conversões de tipos é quando temos uma variável do tipo int/double e a convertemos em uma String ou temos uma String e a convertemos para variável do tipo int/double.
- Usamos o método `.valueOf( )` para fazer a conversão.
- Exemplo: `String a = "32";`  
`double a2 = Double.valueOf(a);`

## 13. Java Lang - pacote java.lang

- Exemplo Conversão de tipo: Criar um pacote **main** e classe **main**. Criar uma String valor com os caracteres 32, criar uma variável **double valor2** e atribuir o valor da conversão de String para double.



The screenshot shows an IDE window titled 'Main.java'. The 'Código-Fonte' (Source Code) tab is active. The code is as follows:

```
14 //Converter String para Double
15 String valor = "32";
16 double valor2 = Double.valueOf(valor);
```

## 13. Java Lang - pacote java.lang

Em seguida criar a variável **double x1** com o valor de 98 e criar uma **String x2** sendo atribuído o valor convertido de x1.

```
18 //converter double para String
19 double x1 = 98;
20 String x2 = String.valueOf(x1);
```

## 13. Java Lang - pacote java.lang

- Como vimos, String não é uma variável e sim um tipo primitivo, por isso que se inicia com letra maiúscula, assim como quando fazemos a conversão de tipo String para double, o Double também inicia-se com letra maiúscula.
- A classe String é muito forte em Java, podendo efetuar varias conversões.

## 14. Classe String

- Algumas das conversões que podem ser efetuadas com String são:
- Para deixar todas as letras maiúsculas usa-se o método `.toUpperCase( )`;
- Para deixar todas as letra minúscula o método é `.toLowerCase( )`;



## 14. Classe String

- Para subtrair caracteres usa-se `.substring(1, 3);`

Com 1 argumento ele tira os caracteres até aquele valor.

Com 2 argumentos ele pega os caracteres do 1º argumento até os caracteres do 2º argumento.

- Para trocar caracteres/palavras de uma frase utilizamos o `.replace(caractere,palavra);`

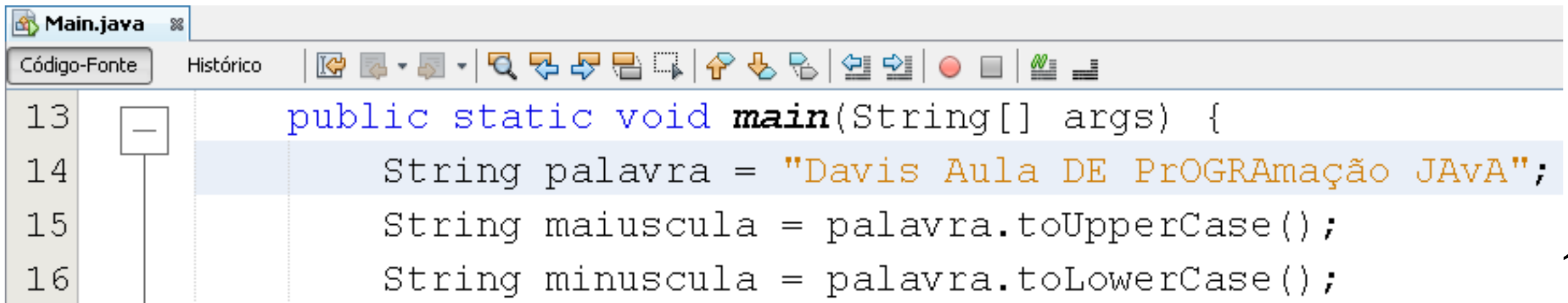
## 14. Classe String

- Se quisermos saber o tamanho da frase, teremos que utilizar o `.length( )`; porém temos que salvar esse valor em uma variável do tipo `int`.
- Outro método que podemos utilizar para saber a posição de uma letra/palavra é o `(.indexOf(palavra, letra))`, também devemos salvar em `int`.

# 14. Classe String

## Exemplos Trabalhando com String:

1) Letras maiúsculas e minúsculas: Criar o pacote **main** e classe **Main**. Criar as String's **palavra**, **maiuscula** e **minúscula**, chamar os métodos para deixa-las maiúscula e minúsculas salvando-as nas String's correspondentes.



```
13 public static void main(String[] args) {
14     String palavra = "Davis Aula DE PrOGRamação JAvA";
15     String maiuscula = palavra.toUpperCase();
16     String minusculta = palavra.toLowerCase();
```

# 14. Classe String

## Exemplos Trabalhando com String:

1) Pedir para exibir as String's **maiúscula** e **minúscula** na tela.

```
17  
18         System.out.println("Maiuscula "+maiuscula);  
19         System.out.println("Minuscula "+minuscula);  
20     }
```

Saída - GPJ - Aula 4 - ex 7 - Maiusculo e Minusculo (run) ✖

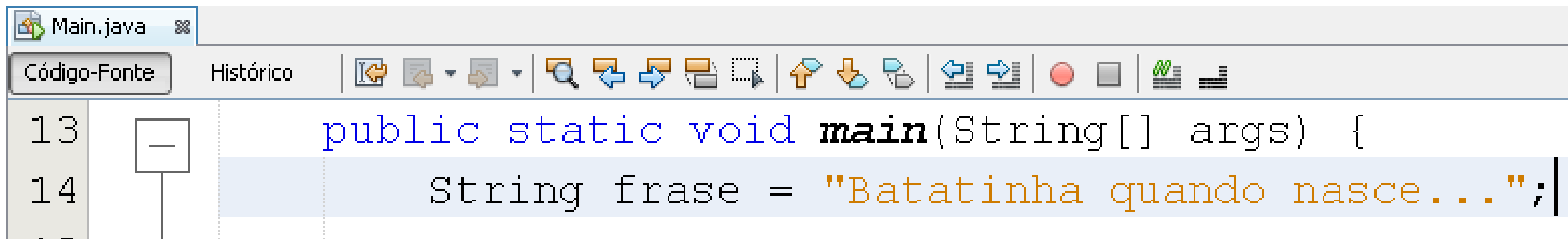
```
run:  
Maiuscula DAVIS AULA DE PROGRAMAÇÃO JAVA  
Minuscula davis aula de programação java  
CONSTRUÍDO COM SUCESSO (tempo total: 2 segundos)
```

Veja as saídas que ficaram todas maiúsculas ou minúsculas

# 14. Classe String

## Exemplos Trabalhando com String:

2) Subtrair palavras: Na classe **Main** criar a String **frase** e escrever algo.



The screenshot shows a Java IDE window titled 'Main.java'. The 'Código-Fonte' (Source Code) tab is active. The code editor displays the following code:

```
13 public static void main(String[] args) {  
14     String frase = "Batatinha quando nasce...";  
15 }
```

The code is written in a monospaced font. The line numbers 13 and 14 are visible on the left. The code is color-coded: 'public' is blue, 'static' is blue, 'void' is blue, '**main**' is bold black, 'String' is blue, 'args' is black, '{' is black, 'String' is blue, 'frase' is black, '=' is black, and the string value is in orange. The IDE has a toolbar with various icons for editing and running code.

# 14. Classe String

## Exemplos Trabalhando com String:

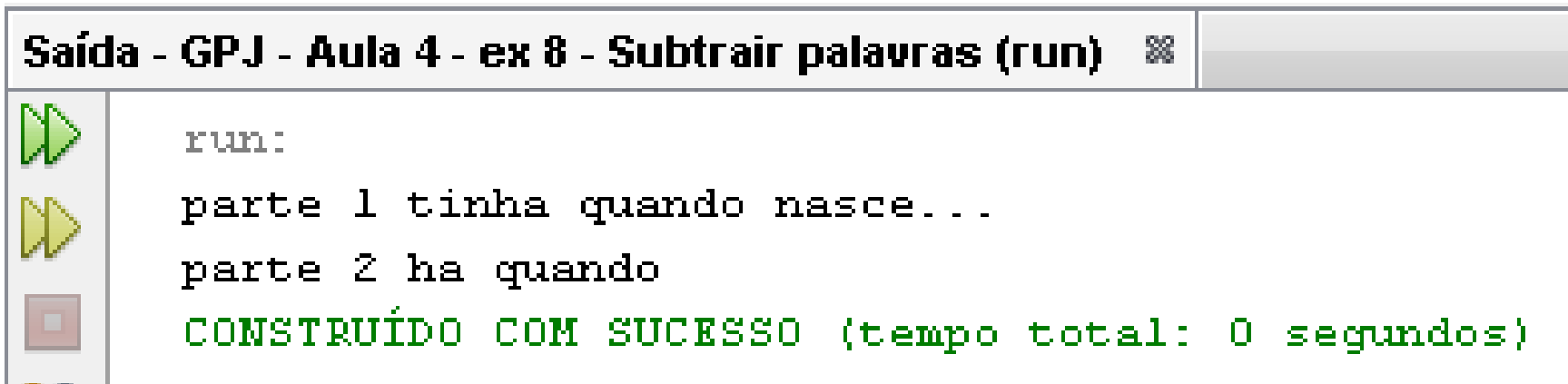
2) Criar a String **parte1** e pegar a frase sem os 4 primeiros caracteres, na String **parte2**, pegar do intervalo 7 até o 16 caractere. Em seguida mostrar as String's **parte1** e **parte2**.

```
16 String parte1 = frase.substring(4);  
17 String parte2 = frase.substring(7, 16);  
  
19 System.out.println("parte 1 "+parte1);  
20 System.out.println("parte 2 "+parte2);  
21 }
```

# 14. Classe String

## Exemplos Trabalhando com String:

2) Note na saída que os caracteres nas posições indicadas foram cortados.



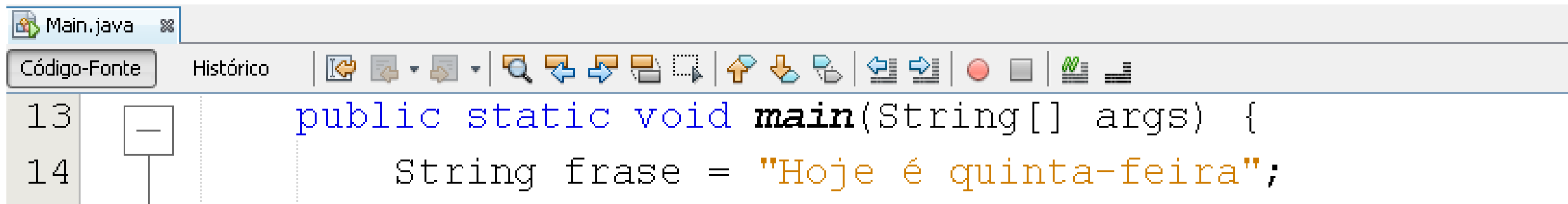
```
Saída - GPJ - Aula 4 - ex 8 - Subtrair palavras (run)
run:
parte 1 tinha quando nasce...
parte 2 ha quando
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

**Obs:** esse valor de posição não necessariamente precisa ser um numero, podendo ser uma variável do tipo int

# 14. Classe String

## Exemplos Trabalhando com String:

3) Trocar palavra, mostrar posição do caracter e mostrar tamanho da frase. Na classe **Main** criar a String **frase** e escrever “Hoje é quinta-feira”.



```
13 public static void main(String[] args) {  
14     String frase = "Hoje é quinta-feira";
```



# 14. Classe String

## Exemplos Trabalhando com String:

3) Criar uma String **trocar** e nela atribuir o valor da troca da palavra “Hoje” pela “Amanhã”, criar uma variável int **posição** e salvar a posição da letra “e”, criar a variável int **tamanho** e mostrar e atribuir o tamanho da frase.

```
16 String trocar = frase.replace("Hoje", "Amanhã");  
17 int posicao = frase.indexOf("e");  
18 int tamanho = frase.length();
```

# 14. Classe String

## Exemplos Trabalhando com String:

3) Pedir para mostrar a String trocar as variáveis posição e tamanho.

```
20      System.out.println("trocar : "+trocar);  
21      System.out.println("posição : "+posicao);  
22      System.out.println("Tamanho : "+tamanho);
```

Saída - GPJ - Aula 4 - ex 9 - Trocar, posição e tamanho (run)

```
run:  
trocar : Amanhã é quinta-feira  
posição : 5  
Tamanho : 19  
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

Veja que na saída foi alterado a palavra, a posição e o tamanho da frase foi mostrado.

# 14. Classe String

## Exercícios Trabalhando com String:

- 1) Criar um programa que, dado um número de telefone no padrão (11)99999-0000, dizer se é fixo ou celular. V1
- 2) Do exercício acima, padronizar os números de telefone da seguinte forma “11987651234”, mostrar o ddd, o número do telefone padronizado e se é celular, fixo ou indefinido.

V2

# 14. Classe String

## Exercícios Trabalhando com String:

3) Dado um nome completo, mostrar apenas o primeiro nome da pessoa.

4) Do exercício anterior, mostrar apenas o primeiro e o último nome da pessoa.

## 15. Classe Math

- A classe **Math** contém métodos para executar operações numéricas básicas, como arredondar valores decimais, e operações matemáticas complexas, como as funções exponenciais, logarítmicas e trigonométricas.

## 15. Classe Math

### •Principais Métodos:

- Round** : retorna o número natural mais próximo.
- Max**: Maior número
- Min**: Menor número
- Pow** : Potência
- Sqrt** : Raiz quadrada

## 15. Classe Math

### Exercício:

Escrever um aplicativo que leia dois valores inteiros informados pelo usuário.

- Informe qual a raiz quadrada do menor número.
- Informe o valor do maior número elevado à 3.

## 16. API Java.IO

- A API Java.IO é utilizada para manipular arquivos, pastas e/ou diretórios utilizando entrada e saída de dados.
- Uma das maneiras mais simples é utilizando a classe pronta File, que possuem alguns métodos importantes para manipular arquivos.



## 17. Classe File

Alguns metodos da Classe File são:

- `exists( )` – verifica se existe o arquivo especificado.

Exemplo: `arquivo.exists();`

- `isFile( )` – Verifica se o item é um arquivo.

Exemplo: `arquivo.isFile();`

## 17. Classe File

- `isDirectory( )` – Verifica se o item é uma pasta.

Exemplo: `arquivo.isDirectory ( );`

- `.length( )` – Mostra a quantidade de arquivos na pasta.

Exemplo: `arquivo.length( );`

## 17. Classe File

- Para utilizarmos os métodos da classe file, precisamos colocar o caminho do arquivo no qual queremos trabalhar. Você indica o caminho e com os métodos buscamos o que foi solicitado.
- Em Java para indicar o caminho sempre usar “\” ou “/” .
  - Caminho -> Externo - C:\\...
  - Caminho -> Interno - src\\main\\...

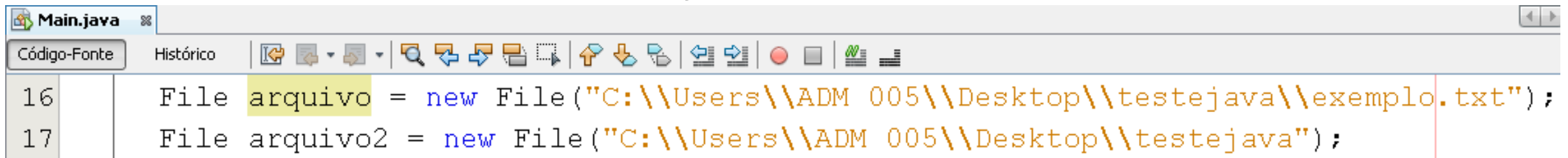
## 17. Classe File

- Para abrir e/ou verificar arquivos ou pastas em um endereço fora dos arquivos do Java, utilizamos o endereço do sistema operacional.

Exemplo: C:\\...

## 17. Classe File

**Exemplo Classe File:** Criar uma pasta na área de trabalho com 3 arquivos. Um deles **.txt** com o nome **exemplo**. No Netbens, criar pacote e classe **Main**. Instanciar os objetos da classe **File** com o nome **arquivo**, nele colocar o endereço e nome do arquivo no seu construtor e no **arquivo2** apenas o endereço.



```
16 File arquivo = new File("C:\\Users\\ADM 005\\Desktop\\testejava\\exemplo.txt");
17 File arquivo2 = new File("C:\\Users\\ADM 005\\Desktop\\testejava");
```

## 17. Classe File

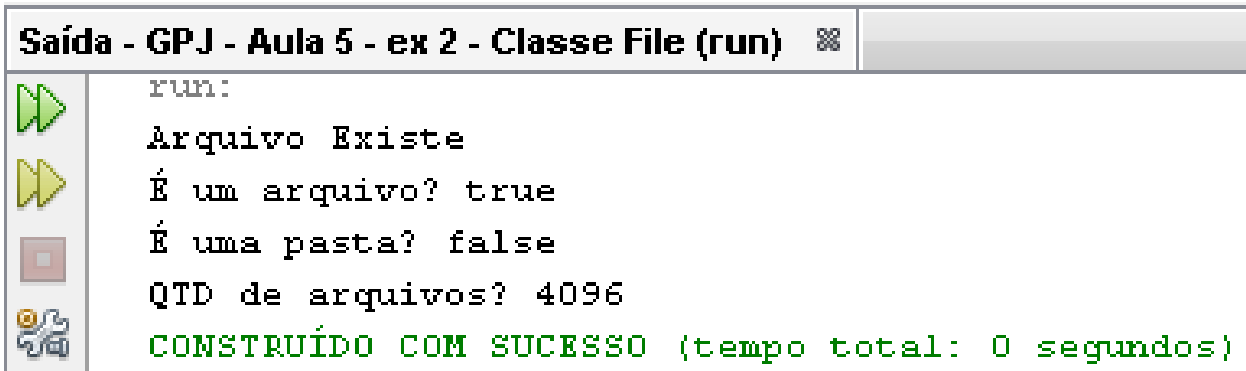
Após instanciar os objetos, criar uma condição para verificar se o arquivo existe ou não existe.

```
19     if (arquivo.exists()) { // se existir exibe true
20         System.out.println("Arquivo Existe");
21     }
22     else {
23         System.out.println("Não Existe");
24     }
```

## 17. Classe File

Em seguida, pedir para mostrar se é um arquivo, se é um diretório e a quantidade de arquivos que tem no endereço informado

```
25      System.out.println("É um arquivo? "+arquivo.isFile());
26      System.out.println("É uma pasta? "+arquivo.isDirectory());
27      System.out.println("QTD de arquivos? "+arquivo2.length());
```



```
Saída - GPJ - Aula 5 - ex 2 - Classe File (run)
run:
Arquivo Existe
É um arquivo? true
É uma pasta? false
QTD de arquivos? 4096
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

A saída mostra que o arquivo existe. Deu verdadeiro para arquivo e falso para pasta, além de mostra a quantidade de arquivos.

## 17. Classe File

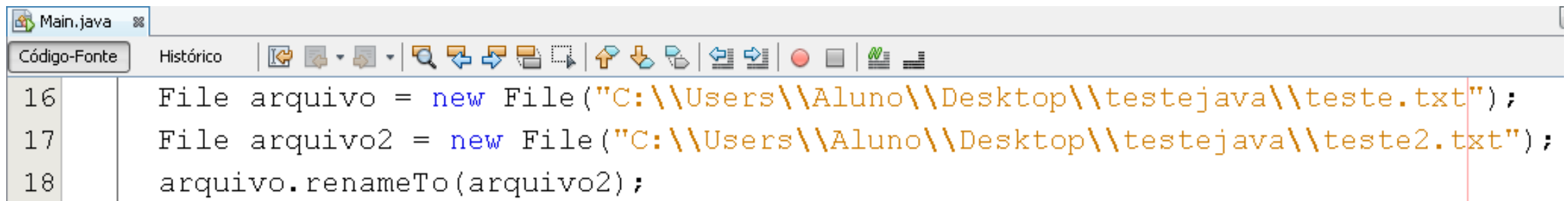
- Em Java é possível chamar vários arquivos que chamam outros arquivos, como fazemos com métodos.
  - Também é possível deletar e renomear arquivos com os métodos da classe File.
- .renameTo( ); - comando para renomear arquivos.
- .delete( ), - Comando para deletar um arquivo.



## 17. Classe File

**Exemplo Classe File:** Criar o pacote e classe **Main**.

Instanciar os objetos da classe File com nome **arquivo** com o endereço e nome do arquivo no construtor e no **arquivo2** com o endereço e o novo nome. Chamar o método **.renameTo( )**;

A screenshot of a Java IDE window titled 'Main.java'. The window has a menu bar with 'Código-Fonte' and 'Histórico', and a toolbar with various icons. The code editor shows three lines of Java code: line 16 creates a File object 'arquivo' with a path to 'teste.txt', line 17 creates a File object 'arquivo2' with a path to 'teste2.txt', and line 18 calls 'arquivo.renameTo(arquivo2);'.

```
16 File arquivo = new File("C:\\Users\\Aluno\\Desktop\\testejava\\teste.txt");
17 File arquivo2 = new File("C:\\Users\\Aluno\\Desktop\\testejava\\teste2.txt");
18 arquivo.renameTo(arquivo2);
```

## 17. Classe File

- Podemos também abrir um arquivo dentro da pasta do próprio projeto, evitando assim, que ao abrir o projeto em outro computador, não será necessário criar/copiar as pastas e os arquivos, pois os mesmos já estarão no projeto.

Utilizamos o caminho, conhecido como caminho absoluto, “scr” ao invés de “C:”. Muito utilizado para imagens.

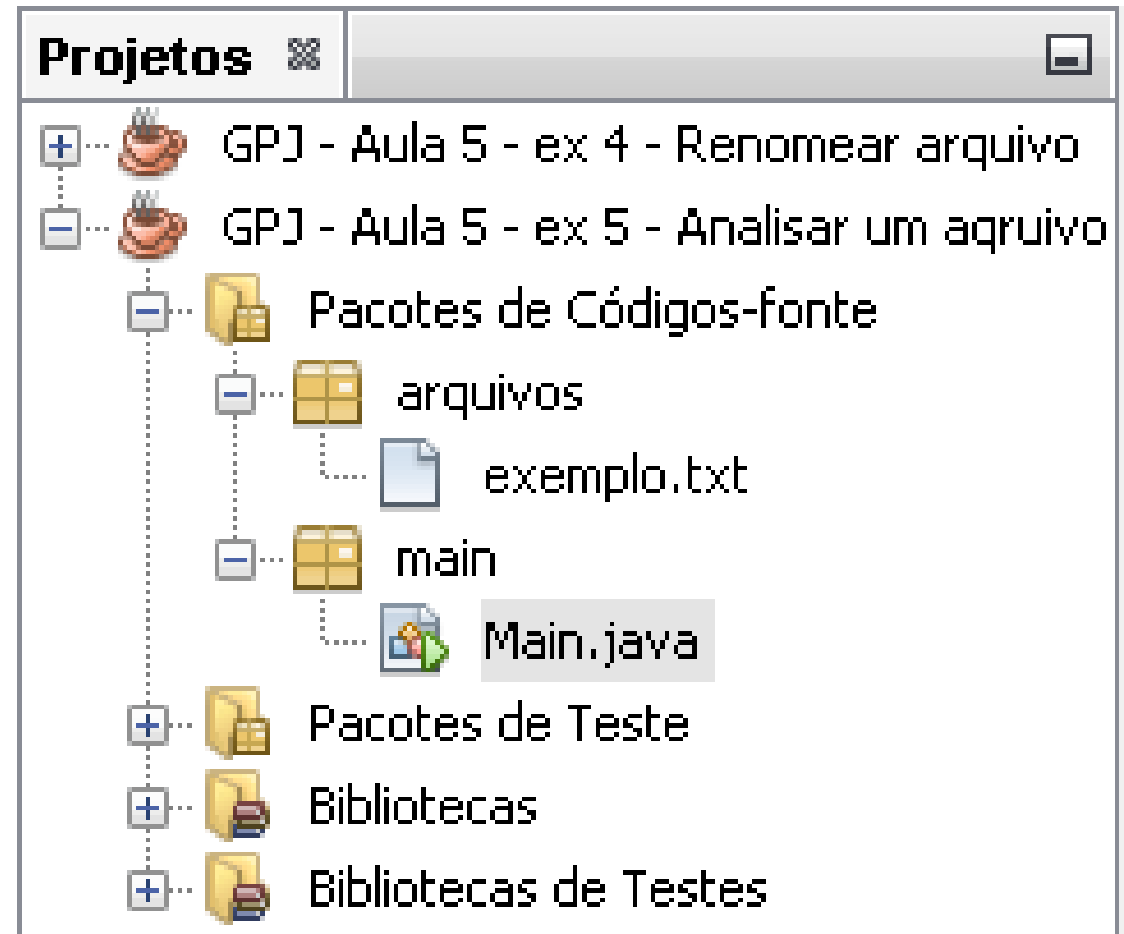
Exemplo: scr\\pasta\\arquivo

# 17. Classe File

## Exemplo Classe File

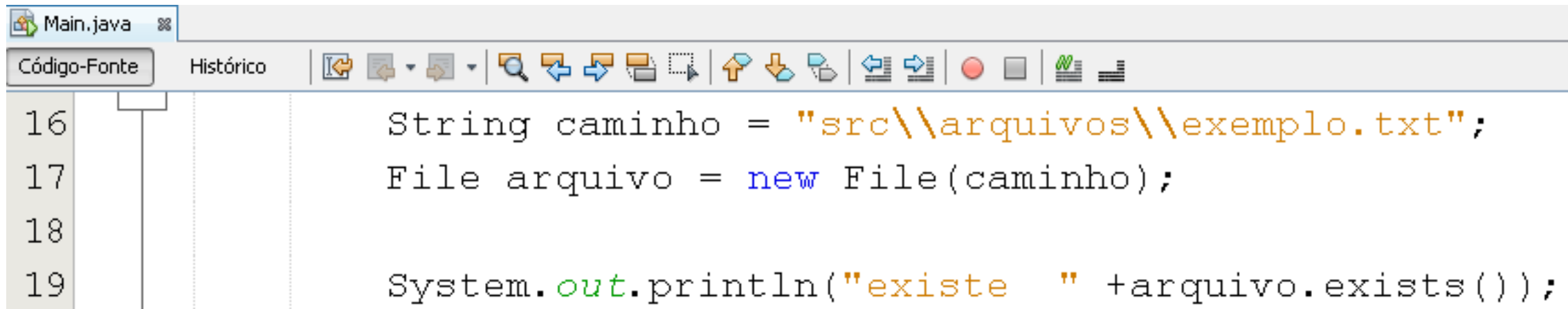
Arquivos no próprio projeto:

Criar o pacote **arquivos** e copiar o arquivo exemplo no pacote, criar o pacote **main** e a classe **Main**.



## 17. Classe File

Na classe **Main** criar uma String com caminho e atribuir o endereço e o nome do arquivo. Instanciar o objeto da classe **File** com a String **caminho** em seu construtor. Mostrar na tela se existe o arquivo informado.

A screenshot of a Java IDE window titled 'Main.java'. The window has a menu bar with 'Código-Fonte' and 'Histórico'. Below the menu bar is a toolbar with various icons for editing and development. The main area shows Java code with line numbers 16, 17, 18, and 19 on the left. The code defines a String 'caminho' with the value 'src\\arquivos\\exemplo.txt', creates a File object 'arquivo' using 'new File(caminho)', and prints the result of 'arquivo.exists()' to the console using 'System.out.println'.

```
16 String caminho = "src\\arquivos\\exemplo.txt";  
17 File arquivo = new File(caminho);  
18  
19 System.out.println("existe " +arquivo.exists());
```