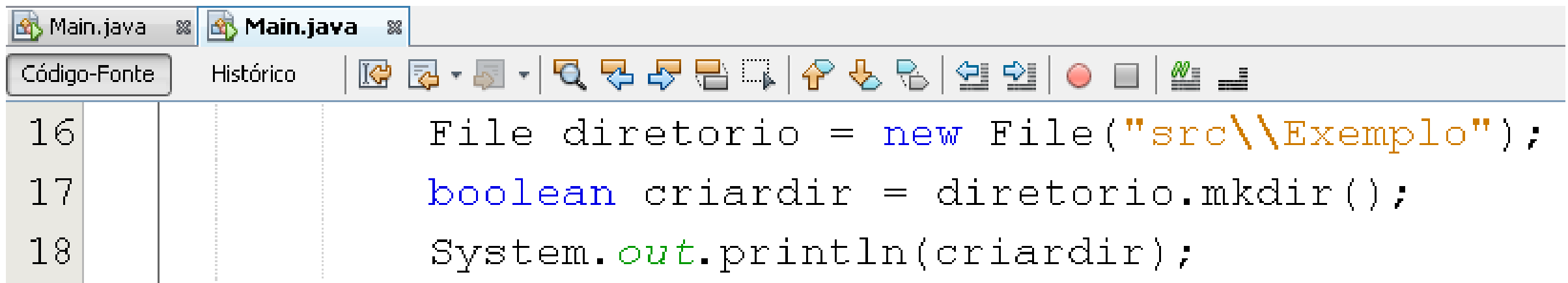


18. Criando Diretórios(pastas)

- Para criarmos um diretório, devemos Instanciar um objeto do tipo **File** e no seu construtor passamos o caminho onde será criado o diretório.
- Exemplo:

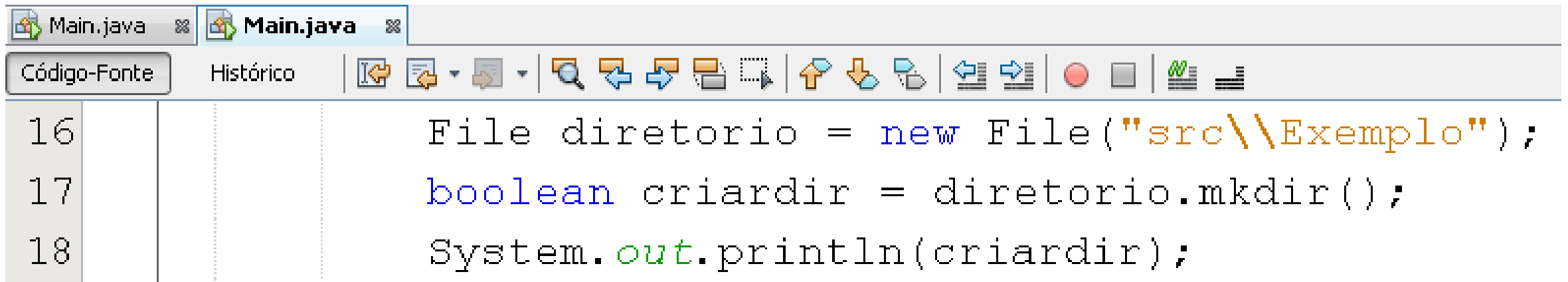


The screenshot shows an IDE window with two tabs, both labeled 'Main.java'. The active tab is 'Main.java'. Below the tabs is a toolbar with various icons for file operations and editing. The main area displays the following Java code:

```
16      File diretorio = new File("src\\Exemplo");
17      boolean criardir = diretorio.mkdir();
18      System.out.println(criardir);
```

19. Criando Arquivos

- Para criarmos um arquivo, devemos Instanciar um objeto do tipo **File** e no seu construtor passamos o caminho onde será criado o diretório. Em seguida criar uma variável booleana e atribuir o diretório com o método **mkdir()**;
- Exemplo:



The screenshot shows an IDE window with two tabs, both labeled 'Main.java'. The 'Código-Fonte' (Source Code) tab is active. Below the tabs is a toolbar with various icons for file operations and editing. The code editor displays the following Java code:

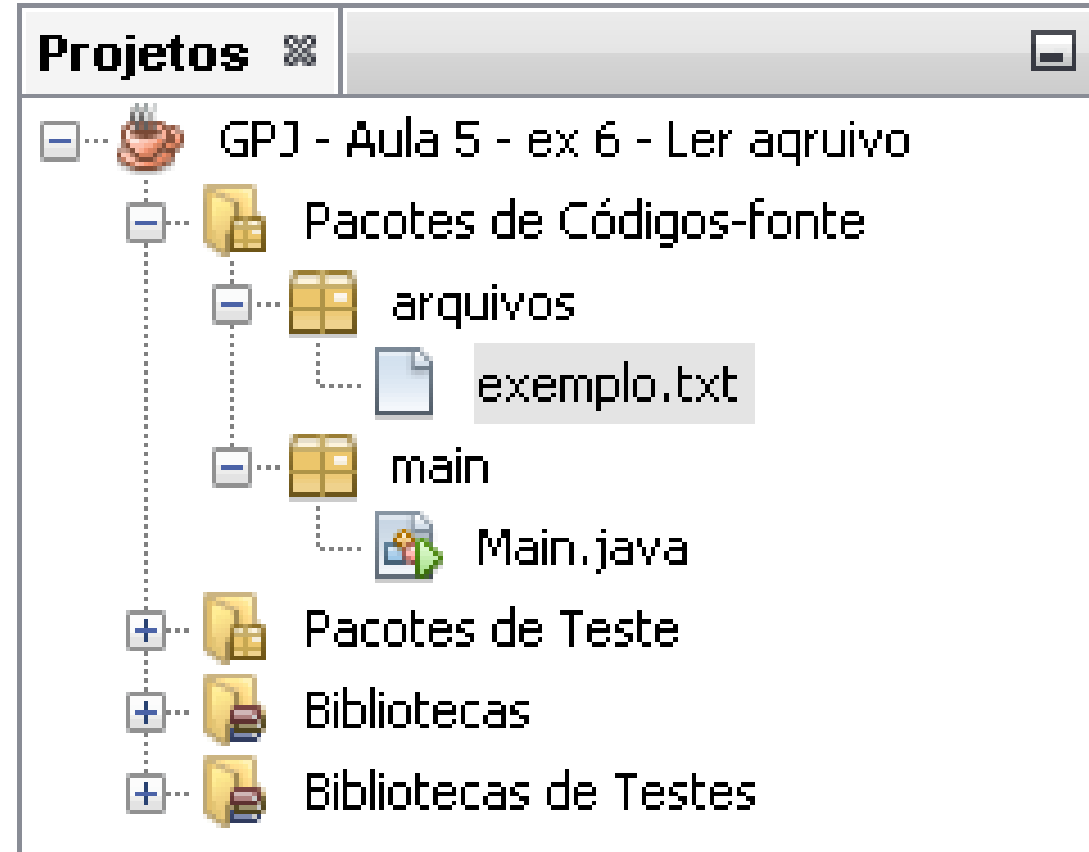
```
16      File diretorio = new File("src\\Exemplo");
17      boolean criardir = diretorio.mkdir();
18      System.out.println(criardir);
```

20. Leitor de arquivos

- Em Java também é possível ler um arquivo. Para podermos abrir e ler um arquivo, usamos duas classes principais. Que são:
 - `InputStream`
 - `FileInputStream`
- Quando lemos um arquivo usando o método `.read()` ele vem em binário, então temos que converter em `String`.

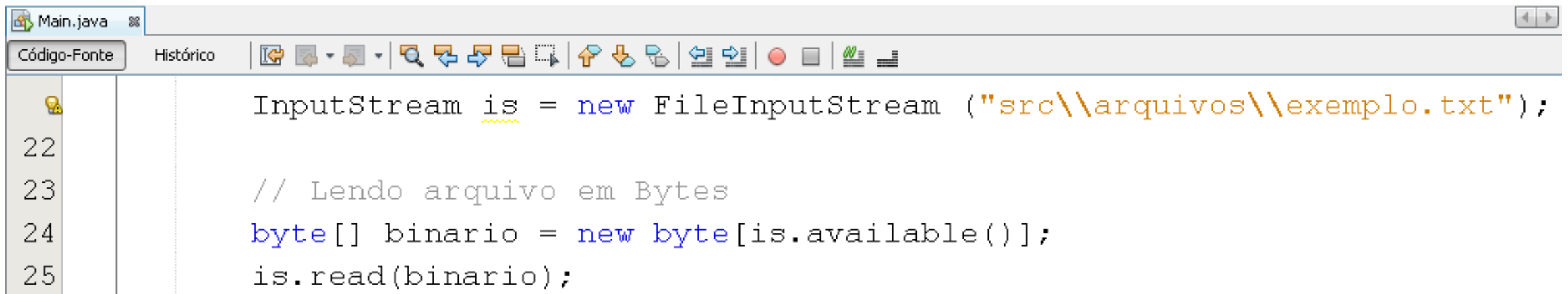
20. Leitor de arquivos

Exemplo de Leitor de Arquivos: Criar um pacote **main** e classe **Main**, copiar o pacote **arquivos** do projeto anterior.



20. Leitor de arquivos

Instanciar o objeto da classe **InputStream** com o da classe **FileInputStream**, colocando o endereço e nome do arquivo. Instanciar o objeto byte verificando se o arquivo é válido, em seguida ler o arquivo salvando no objeto criado.

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 the following Java code:

```
InputStream is = new FileInputStream ("src\\arquivos\\exemplo.txt");

// Lendo arquivo em Bytes
byte[] binario = new byte[is.available()];
is.read(binario);
```

The line numbers 22, 23, 24, and 25 are visible on the left side of the editor.

20. Leitor de arquivos

Fazer a conversão do conteúdo binário para String, instanciando um objeto da classe String e atribuindo o valor da conversão binário para “UTF-8”. Em seguida pedir para mostrar o texto, no final fechar o arquivo. “.close();

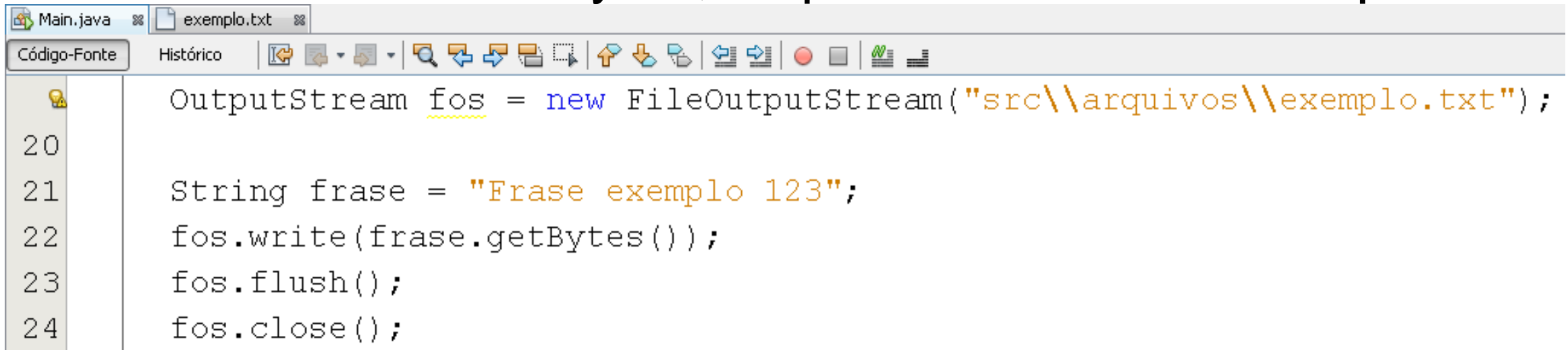
```
27 //Convertendo para texto
28 String texto = new String(binario, "UTF-8");
29 System.out.println(texto);
30 is.close();
```

21. Escritor de arquivos

- Para escrever em um arquivo também utilizamos duas classes que são as seguintes:
- `OutputStream`
- `FileOutputStream`
- Para escrever no arquivo, temos que abrir o arquivo, converter a `String` em `Bytes`, gravar usando o método `.write()`; e fechar o arquivo.

21. Escritor de arquivos

- **Exemplo Escrevendo no arquivo:** Criar um pacote e classe **Main**, copiar a pasta **arquivos** do projeto anterior e apagar o conteúdo do arquivo. Instanciar o objeto da classe de escrita, criar a frase, usar o método “.write” para escreve-la, transformando-a em Bytes, limpar buffer e fechar arquivo.

A screenshot of an IDE window with two tabs: 'Main.java' and 'exemplo.txt'. The 'Main.java' tab is active, showing a code editor with a toolbar at the top. The code in the editor is as follows:

```
OutputStream fos = new FileOutputStream("src\\arquivos\\exemplo.txt");  
20  
21 String frase = "Frase exemplo 123";  
22 fos.write(frase.getBytes());  
23 fos.flush();  
24 fos.close();
```


Exercício de arquivos

- V.1 – Criar um programa contador, ele deverá contar toda vez que o arquivo for aberto e salvar o novo valor dentro do próprio arquivo.
- V.2 – Transformar o contador do exercício anterior em um método contar.

Exercício de arquivos

- Criar um novo projeto com o nome **CriandoArquivosDiretorio**
- Criar dois diretórios conforme abaixo: **(nota: efetuar validação da existência dos diretórios)**
 - Unidade C: , nome da pasta **Cidades**.
 - Desktop, nome da pasta **Países**
- Dentro da pasta **Cidades** criar **3** arquivos: **idades_sudeste.txt**, **idades_nordeste.xls** e **idades_norte.doc**
- Dentro da pasta **Países** criar **2** arquivos: **países_europeus.ppt** e **países_asiaticos.pdf**
- Renomear os arquivos conforme abaixo:
 - **Cidades_Sudeste.txt** para **Cidades_Sul.txt**
 - **Países_europeus.ppt** para **países_africanos.ppt**
- Excluir os arquivos: **países_asiaticos.pdf** e **idades_norte.doc**
- Listar o conteúdo das duas pastas.

Exercício de arquivos

- Criar um diretório na unidade C: com o nome **ArquivosTexto**
- Criar dois arquivos do tipo texto (.txt) com os nomes: **pessoas.txt** e **idades.txt**
- No arquivo **pessoas.txt** escrever o nome de 5 pessoas.
- No arquivo **idades.txt** escrever o nome de 4 cidades.

API Java .útil

22. Classe SimpleDateFormat

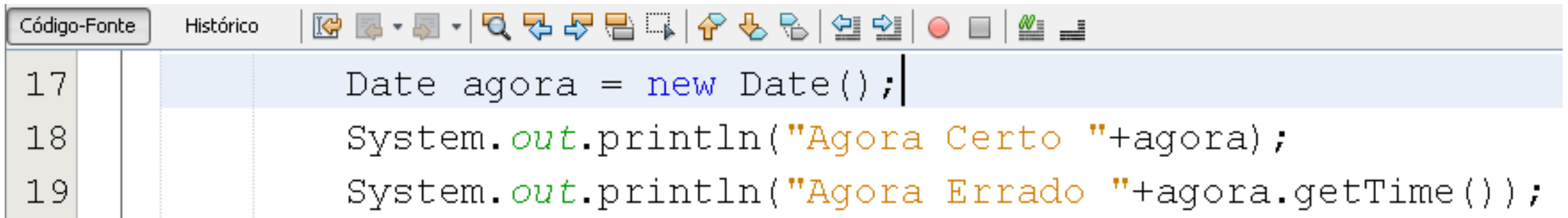
- Para trabalhar com data no Java temos 3 classes:
- A Classe Date, porém é limitada.
- A Classe Calendar é uma Classe abstrata, para usar é necessário usar o método estático **getInstance()**
- SimpleDateFormat – Permite converter para um padrão determinado.

22. Classe SimpleDateFormat

- A classe Date pega a data no PC no padrão Americano.
- A classe SimpleDateFormat permite fazer alterações no formato da data, passando do padrão americano para o brasileiro.

22. Classe SimpleDateFormat

Exemplo Date e SimpleDateFormat: Na classe **Main**, Instanciar objeto da classe **Date** para pegar a data atual do PC e mostrar a data, utilizar o método `.getTime()` para pegar a data em segundos desde 1970.

A screenshot of a Java IDE window. The window has a title bar with 'Código-Fonte' and 'Histórico'. Below the title bar is a toolbar with various icons for file operations, editing, and running. The main area shows three lines of Java code:

```
17 Date agora = new Date();  
18 System.out.println("Agora Certo "+agora);  
19 System.out.println("Agora Errado "+agora.getTime());
```

22. Classe SimpleDateFormat

Exemplo Date e SimpleDateFormat: Instanciar o objeto da classe **SimpleDateFormar** para converter a data para String e colocar no padrão desejado, salvar em uma String formatando com o método `.format` e mostrar a saída.

```
21 //Conversão de date para string
22 SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
23 String agora2 = formato.format(agora);
24 System.out.println("Agora 2 "+agora2);
```

22. Classe SimpleDateFormat

Exercício Date e SimpleDateFormat: 1) Criar um programa para mostrar que dia da semana é hoje, em português.

2) Faça um aplicativo que recupere a hora do dia e exiba uma mensagem conforme regra abaixo:

- **6h – 12h = Bom dia**
- **12h – 18h = Boa tarde**
- **Caso contrário = Boa noite**

22. Classe SimpleDateFormat

- Para converter tipos de dados para o formato date utilizamos o método .parse

- Exemplo:

```
Date data2 = formato.parse(data);
```

22. Classe SimpleDateFormat

- **Exercícios:** Construa um aplicativo onde o usuário informa a sua data de nascimento e ao final o sistema calcula quantos dias faltam para o seu aniversário e mostre qual dia da semana cairá. Caso já tenha ocorrido, exibir somente “Aniversário já passou!”.

23. Arrays

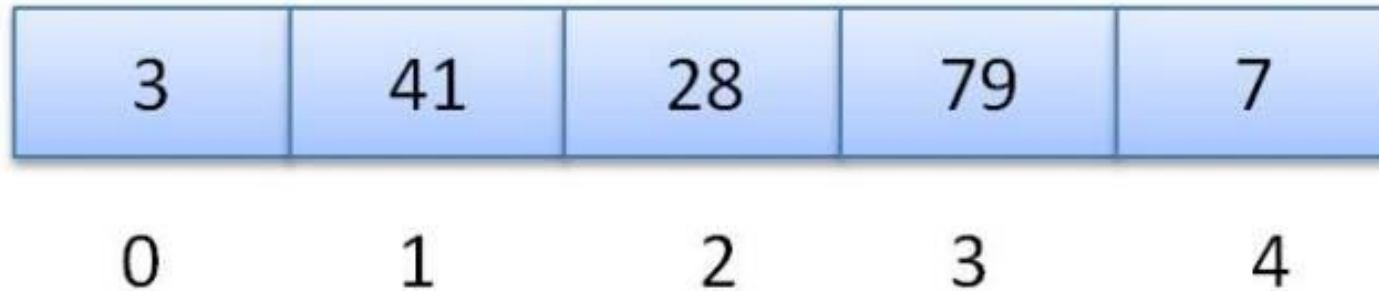
- Imagina criar 100 variáveis de um mesmo tipo, como seria?

Exemplo: `int a1, a2, a3 ... a99, a100;`

Não seria viável, porém temos como usar um arranjo para fazemos essa declaração.

23. Arrays

Os **objetos** contêm um número fixo de valores de um único tipo. O comprimento de um **array** é estabelecido quando criado, sendo que após a criação o seu comprimento fica fixo.

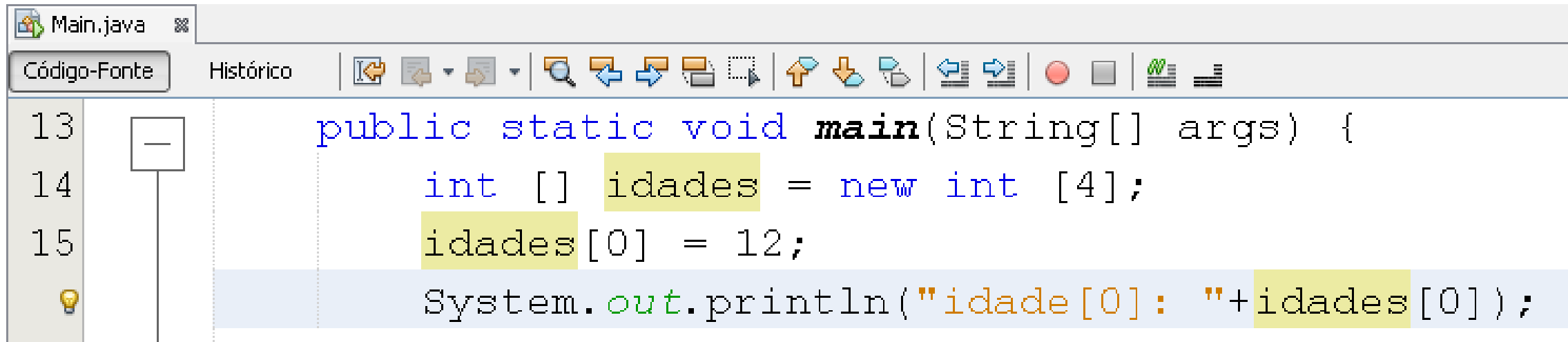


Sintaxe:

```
tipo[ ] nome_do_array = new tipo[numero_de_elementos];
```

23. Arrays

Exemplo Array: Na classe **Main**, criar um array inteiro com o nome **idades** com 4 posições, na posição [0] inserir uma idade. Pedir para mostrar a idade.

A screenshot of a Java IDE window titled 'Main.java'. The window has a toolbar with various icons for editing and running code. The code is displayed in a text area with line numbers 13, 14, and 15 on the left. The code defines a main method that creates an integer array named 'idades' with 4 elements, sets the first element to 12, and prints it. The variable 'idades' and the value '12' are highlighted in yellow. The print statement is highlighted in light blue.

```
13 public static void main(String[] args) {  
14     int [] idades = new int [4];  
15     idades[0] = 12;  
    System.out.println("idade[0]: "+idades[0]);
```

24. Coleções (Collections)

- O Java, por padrão, possui uma série de recursos prontos (APIs) para que possamos tratar de estrutura de dados, também chamados de coleções (**Collections**).
- Armazena diferentes tipos de elementos, desde variáveis até objetos efetuando a alocação dinâmica da memória.

24. Coleções (Collections)

- Em Collections não é preciso definir o tamanho da lista. Isso só é possível porque o Java não utiliza endereço de memória para armazenamento.
- Uma vantagem é poder pegar várias informações digitadas pelo usuário.
- Existem alguns tipos diferentes de Coleções, entre elas as List e Map.

25. ArrayList

- É o mais genérico que temos em Java, o ArrayList é bem parecido com um vetor, porém em vetor, todos os elementos são do mesmo tipo e em ArrayList não tem esse limite, assim como um vetor, a ordem que adicionamos os elementos é a ordem que ele ficará armazenado.
- Para utilizar um arraylist temos que instanciar o objeto;
`ArrayList lista = new ArrayList();`

25. ArrayList

Alguns métodos mais utilizados no ArrayList

.add() : Adiciona elementos

.remove() : remove elementos

.clear() : limpa lista

.toArray() : lista os elementos

.get() : lista o elemento de um índice

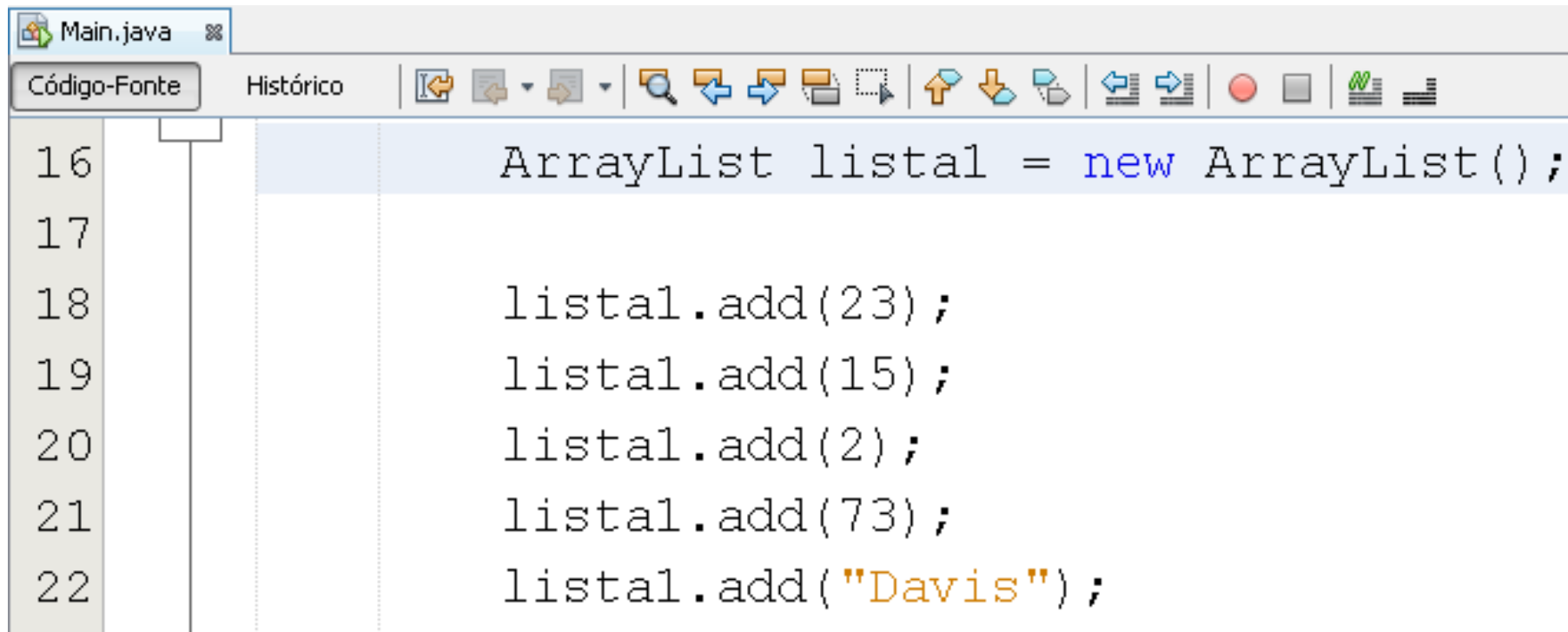
.indexOf() : Lista o índice de um elemento

.size() : número de elementos da lista

.set() : atualização de elementos

25. ArrayList

Exemplo de ArrayList: Na classe **Main** instanciar o objeto ArrayList, adicionar 5 elementos na lista, podendo ser inteiro, flutuante ou String.

A screenshot of a Java IDE window titled 'Main.java'. The window has a toolbar with various icons for editing and running code. Below the toolbar, there are tabs for 'Código-Fonte' (Source Code) and 'Histórico' (History). The source code is displayed in a text area with line numbers on the left. The code defines an ArrayList named 'lista1' and adds five elements to it: 23, 15, 2, 73, and the string 'Davis'.

```
16 ArrayList lista1 = new ArrayList();  
17  
18 lista1.add(23);  
19 lista1.add(15);  
20 lista1.add(2);  
21 lista1.add(73);  
22 lista1.add("Davis");
```

25. ArrayList

Exemplo de ArrayList: Mostrar a quantidade de elementos da lista “**.size()**”. Para mostrar os elementos da lista temos várias maneiras, uma é usando o sout e pedir pra mostrar a lista.

```
24 //quantidade de elementos V1
25 System.out.println("QTD: "+lista.size());
26
27 //mostrar todos elementos V1
28 System.out.println("\nElementos da lista: "+lista);
```

25. ArrayList

A outra maneira é mostrando os elementos da lista um de cada vez, utilizando o número de seu elemento.

```
30      //mostrar todos elementos V2
31      System.out.println("\nElementos 0:  "+listal.get(0));
32      System.out.println("Elementos 1:  "+listal.get(1));
33      System.out.println("Elementos 2:  "+listal.get(2));
```

A terceira maneira é usando um looping para mostrar os elementos, temos que utilizar o size para contar o tamanho.

```
35      System.out.println("\nVersão 3");
36      //mostrar todos os elementos.
37      for (int i = 0; i < listal.size(); i++) {
38          System.out.println("Elemento "+i+" = "+listal.get(i));
```

25. ArrayList

Note na saída a diferença das 3 maneiras de mostrar os elementos de um ArrayList.

```

Saída - GPJ - Aula 6 - ex 1 - Exemplo ArrayList (run)

run:
QTD: 5

Elementos da lista:  [23, 15, 2, 73, Davis]

Elementos 0:  23
Elementos 1:  15
Elementos 2:  2

Versão 3
Elemento 0 = 23
Elemento 1 = 15
Elemento 2 = 2
Elemento 3 = 73
Elemento 4 = Davis
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

```

Exercício ArrayList

1. Criar um ArrayList com o nome **SaladaDeFrutas**.
2. Adicionar os seguintes elementos: **Morango, Laranja, Abacaxi, Mamão, Kiwi e Banana**.
3. Exibir o tamanho do ArrayList.
4. Exibir o elemento de **índice 3**.
5. Exibir o índice do elemento **Kiwi**.
6. Remover os elementos **Laranja e Banana**.
7. Atualizar o elemento **0 para Cereja**.
8. Atualizar o elemento de índice **Mamão** para **Limão**.
9. Exibir **todos** os elementos do ArrayList.
10. Quais serão as frutas da salada de frutas?

Exercício ArrayList

- Criar uma lista só com números, adicionar na lista 2 só os elementos que foram maior de 18, mostrar o total e os elementos da lista 2.
- Criar uma lista com vários números. Remover todos os números que são maior ou igual a 50.

25. ArrayList

- A outra maneira de exibição dos valores de um ArrayList é utilizando o Iterator.
- Para utilizar o Iterator é necessário instanciar o objeto e utilizar o método `.next()`;

Exemplo:

```
Iterator iter1 = arraylist.iterator();  
    while(iter1.hasNext()){  
        System.out.println(iter1.next());  
    }
```


25. ArrayList

Execícioo ArrayList utilizando Interator:

1.Criar um ArrayList com o nome **seqnum**.

1.Desafio: Armazenar nesse ArrayList todos os números ímpares entre 1 e 1000.

1.Exibir o resultado usando **Interator**.

26. TreeSet

- Outro tipo de coleção é o Set. O Set é parecido com o ArrayList, porém não permite repetição de elementos nem de elementos de tipos diferentes (int, double, String).
- O TreeSet efetua a ordenação automática da forma mais natural possível.
- Para utilizar o TreeSet temos que instanciar objeto.

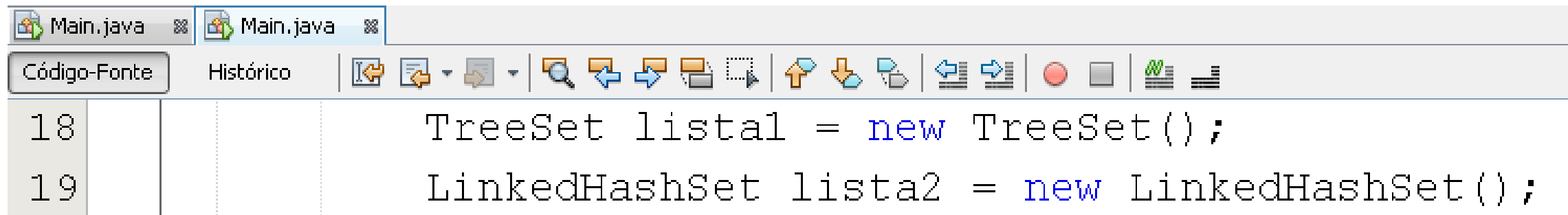
Exemplo: `TreeSet lista1 = new TreeSet();`

27. LinkedHashSet

- O LinkedHashSet , assim como o TreeSet, não permite elementos repetidos, porém ele não realiza ordenação.
- Assim como o ArrayList, o LinkedHashSet permite que seus elementos sejam de tipos diferentes de dados.
- Para mostrar os elementos do TreeSet e do LinkedHashSet utilizamos o Iterator.

Exemplo TreeSet e LinkedHashSet

- Instanciar os objetos das classes TreeSet e LinkedHashSet como lista1 e lista2 sucessivamente.



The screenshot shows an IDE window with two tabs labeled 'Main.java'. The 'Código-Fonte' (Source Code) tab is active. The code editor displays two lines of Java code: line 18: `TreeSet lista1 = new TreeSet();` and line 19: `LinkedHashSet lista2 = new LinkedHashSet();`. The code is written in a monospaced font with syntax highlighting: `new` is in blue, and the semicolons are in black. The IDE interface includes a toolbar with various icons for editing and running code.

```
18      TreeSet lista1 = new TreeSet();  
19      LinkedHashSet lista2 = new LinkedHashSet();
```

28. TreeMap

- Mapas (também chamados de arrays associativos) de objetos são mais uma das formas de organizarmos coleções de objetos. São conjuntos de pares de objetos, sendo um chamado chave e o outro valor.
- A chave pode ser um número ou uma String.

- Sintaxe:

TreeMap mapa1 = new TreeMap();

28. TreeMap

João	Java
Maria	Banco de Dados
José	Lógica
Marta	Cálculo

- **Não permite chaves duplicadas. Valor sim.**

28. TreeMap

- Métodos para utilizar o TreeMap

- .put : Inserir valores (K,V)**

- .get : retorna o elemento de acordo com a chave**

- .entrySet : exibe os valores no formato Key-Value**

- .values : exibe somente os valores**

- .keySet : exibe as chaves**

- .replace: atualiza valores**

28. TreeMap

1. Criar um TreeMap com o nome **controle_de_ips**
2. A chave serão do tipo String
3. Cadastrar os seguintes elementos:

IP	DNS
200.201.10.45	www.uol.com.br
178.890.0.1	www.globo.com
0.0.12.123	www.google.com
200.202.0.4	www.aptech.com.br
189.890.0.1	www.prefeitura.sp.gov.br

1. Retornar o elemento referente ao IP 0.0.12.123
2. Exibir todas as chaves do HashMap
3. Excluir o IP 189.890.0.1
4. Substituir o DNS do uol para www.g1.com.br

29. Expressões Regulares (REGEX)

- Em qualquer software, sempre existem **campos que passam por algumas validações**. O objetivo de ter essas validações é simplesmente o fato de que os dados que são informados podem ter o valor e forma determinados pelo sistema.

29. Expressões Regulares (REGEX)

- Uma expressão regular é uma String especialmente formatada. O principal objetivo dessas expressões é fazer validações nos dados de um programa, assegurando que estes dados estejam em um determinado formato.

- CPF
- CEP
- Telefone

29. Expressões Regulares (REGEX)

Exemplos de Regex:

[r m p]ato → Só permite as palavras rato, mato e pato.

[A-Z] → Somente as letras maiúsculas de A à Z são aceitas.

[a-z A-Z]{n, } → Aceita letras de A à Z maiúscula e minúscula ao menos “n” vezes.

[A-Z]{n} → Tem que ter “n” letras maiúsculas.

[0-9]{n,m} → Aceita de 0 à 9 de “n” até “m” vezes.

29. Expressões Regulares (REGEX)

Para verificar o padrão estabelecido temos que escrever iniciando com “^” e finalizar com “\$”. Também temos que utilizar as Classes Pattern e Matcher. Utilizamos “.find” para verificar a comparação.

29. Expressões Regulares (REGEX)

Exemplo Expressão Regulares: Na Classe **Main** criar uma String com o CPF e outra com o padrão a ser verificado.

```
18 String exemplo = "319.102.988-59";  
19 String padrao = "^[0-9]{3}.[0-9]{3}.[0-9]{3}-[0-9]{2}$";  
20  
21 Pattern p = Pattern.compile(padrao);  
22 Matcher m = p.matcher(exemplo);
```

29. Expressões Regulares (REGEX)

Utilizar uma condicional com o método “.find” para verificar se o padrão está correto ou incorreto.

```
24      if (m.find()) {  
25          System.out.println("Padrão Correto");  
26      }  
27      else {  
28          System.out.println("Padrão Incorreto");  
29      }
```

29. Expressões Regulares (REGEX)

Criar um programa no qual o usuário informa o número de seu telefone e valide se o mesmo está conforme o padrão abaixo.

(XX) – XXXX – XXXX

Nota:

Para os parênteses e traços, utilizar o padrão para qualquer caractere.

30. Genéricos - Generics

Limita a coleção ou o ArrayList a um tipo específico de dado, podendo ser dos tipos: Integer; String; Double;

Exemplos:

`ArrayList<String> listatexto1 = new ArrayList<>();` Limita em String.

`ArrayList<Integer> listatexto2 = new ArrayList<>();` Limita em int.

31. Expressão Lambda

- Função própria do Java 8.
- Quando se cria um método, você tem que dizer os tipos dos argumentos e o tipo de retorno, porém, quando trabalhamos com a Expressão lambda, não é necessário informar esses dados.
- Essa expressão é muito útil para se usar com `arrayList`.

31. Expressão Lambda

- Para utilizar a Expressão Lambda, necessitamos importar a classe **Predicate**.

- Sintaxe:

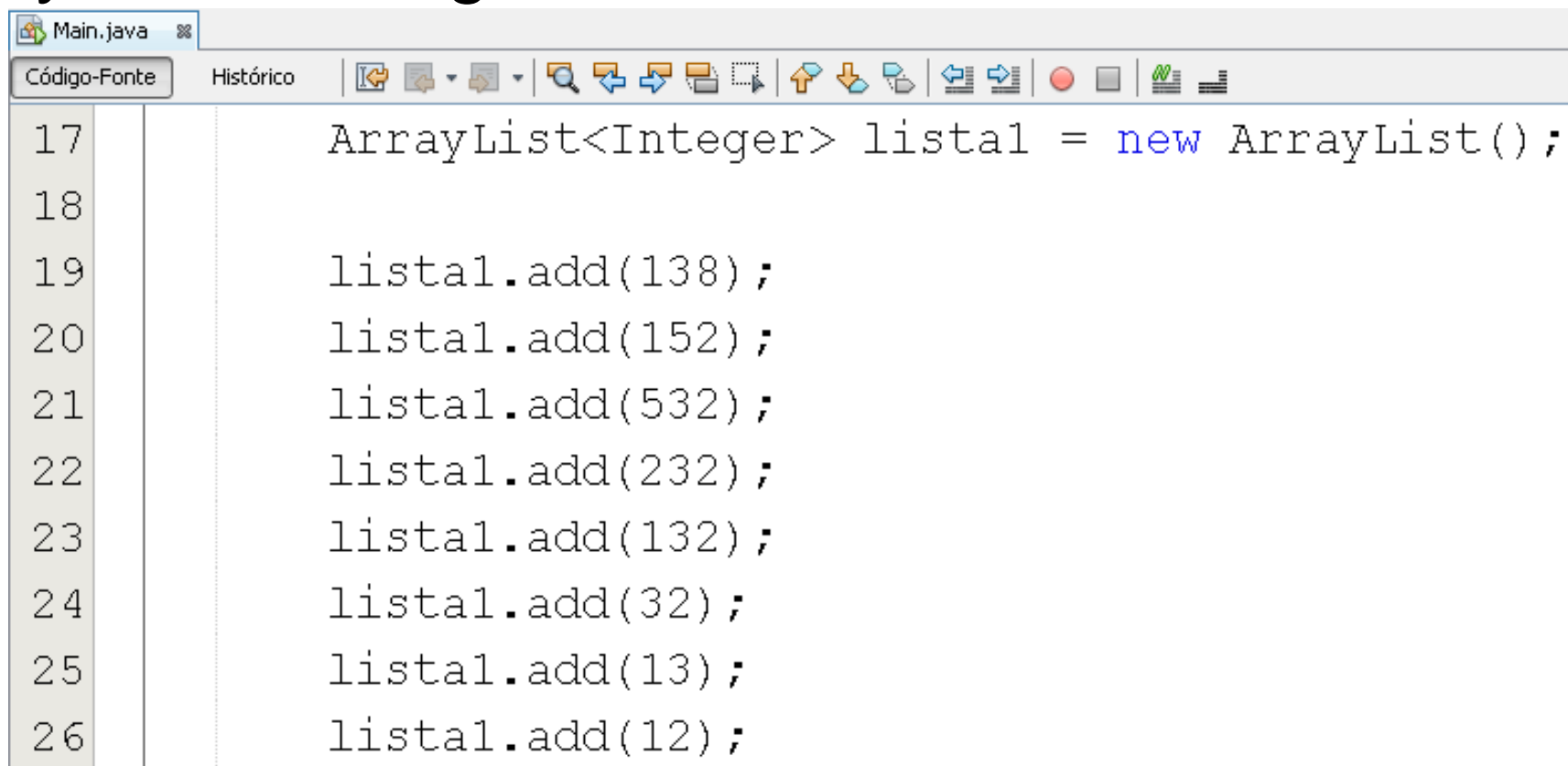
```
n -> { if (predicate.test (n) ) {  
        sout (n) } } // comando equivalente a um método em
```

Java

N = são os elementos da lista.

31. Expressão Lambda

- **Exemplo Expressão Lambda:** Na classe **Main** criar um **ArrayList** com alguns valores atribuídos.

A screenshot of an IDE window titled 'Main.java'. The window has a toolbar with various icons for editing and running code. Below the toolbar, there are two tabs: 'Código-Fonte' (active) and 'Histórico'. The code is written in Java and shows the creation of an ArrayList named 'lista1' and the addition of several integer values to it. The line numbers 17 through 26 are visible on the left side of the code editor.

```
17     ArrayList<Integer> lista1 = new ArrayList();  
18  
19     lista1.add(138);  
20     lista1.add(152);  
21     lista1.add(532);  
22     lista1.add(232);  
23     lista1.add(132);  
24     lista1.add(32);  
25     lista1.add(13);  
26     lista1.add(12);
```

31. Expressão Lambda

- Criar um método chamado **avaliaExpressao** para avaliar o ArrayList criado de acordo com o pedido, utilizando o método da classe Predicate, conforme abaixo.

```
public static void avaliaExpressao(ArrayList<Integer> lista1,  
37 Predicate<Integer> predicate) {  
38     lista1.forEach(n -> {if(predicate.test(n)) {  
39         System.out.println(n);  
40     }});  
41 }
```

31. Expressão Lambda

- Chamar o método criado pedindo apenas os números acima de 50 e chamar outra vez apenas solicitando os números pares.

```
28      System.out.println("Mostrar todos acima de 50:");  
29      avaliaExpressao(listal, (n) -> n >= 50);  
30      System.out.println("");  
31      System.out.println("Mostrar todos numeros Pares:");  
32      avaliaExpressao(listal, (n) -> n % 2 == 0);
```

31. Expressão Lambda

Note que na saída está listado apenas os números acima de 50 e na outra saída só os números pares.

```

Saída - GPJ - Aula 6 - ex 10 - Exemplo Expressão Lambda (run)

run:
Mostrar todos acima de 50:
138
152
532
232
132

Mostrar todos numeros Pares:
138
152
532
232
132
32
12

CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
.
```

32. Sobrecarga de Método

- Métodos na mesma classe que possuem retorno/argumentos diferentes, mas possuem o mesmo nome.

Ex.: `media(double a, double b)`

`media(double a, double b, double c)`

33. Anotações = @.... - Comentário

- @Override - Aviso de sobreposição
- @Deprecated - métodos obsoletos (obs.: Funciona, porém não recomendado utilizar)