



Arrays e Exceções

Arrays em Java

Definição

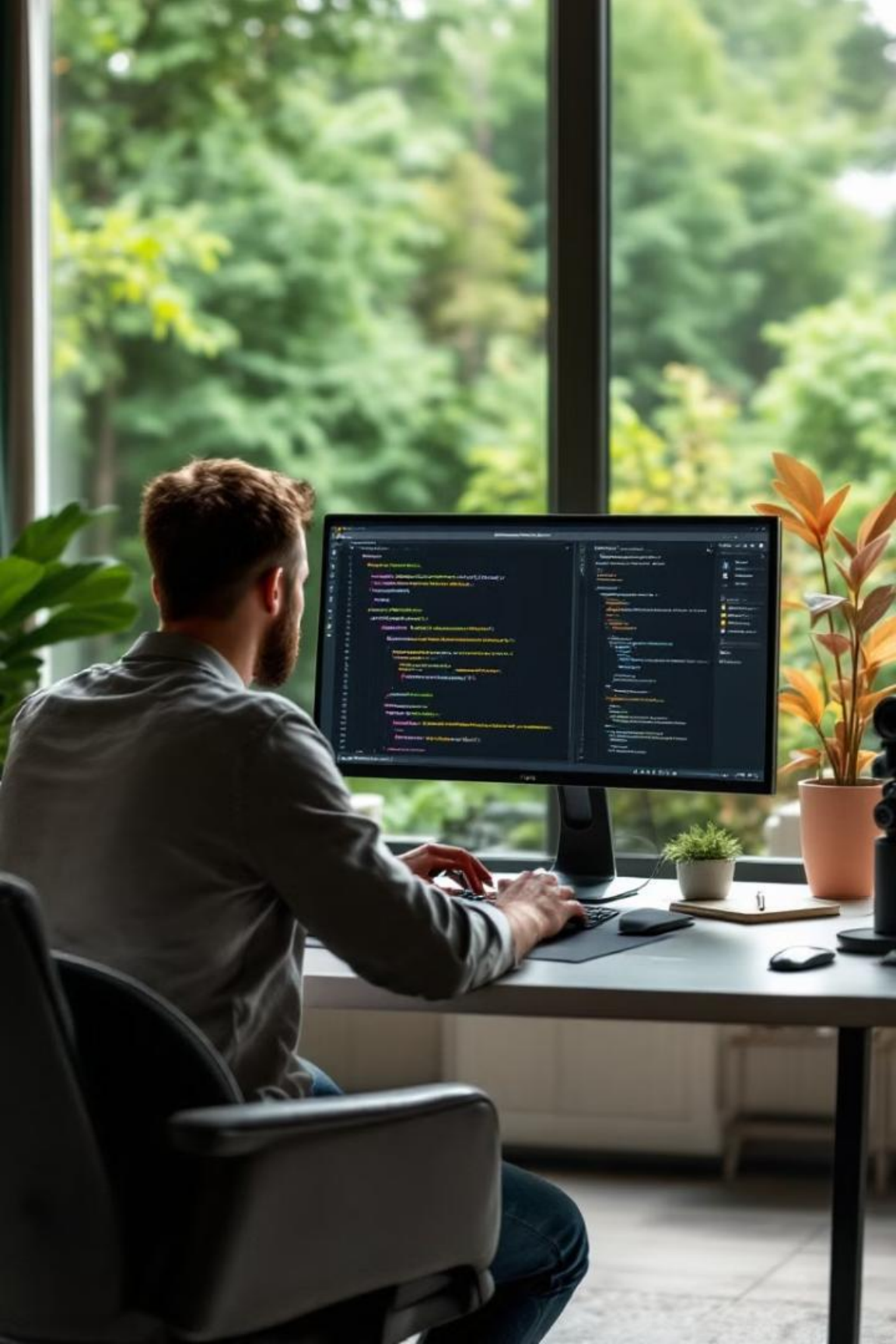
Estrutura de dados homogênea.

Tamanho Fixo

Alocado na criação.

Alocação Contígua

Elementos adjacentes na memória.



Declarando um Array

Sintaxe

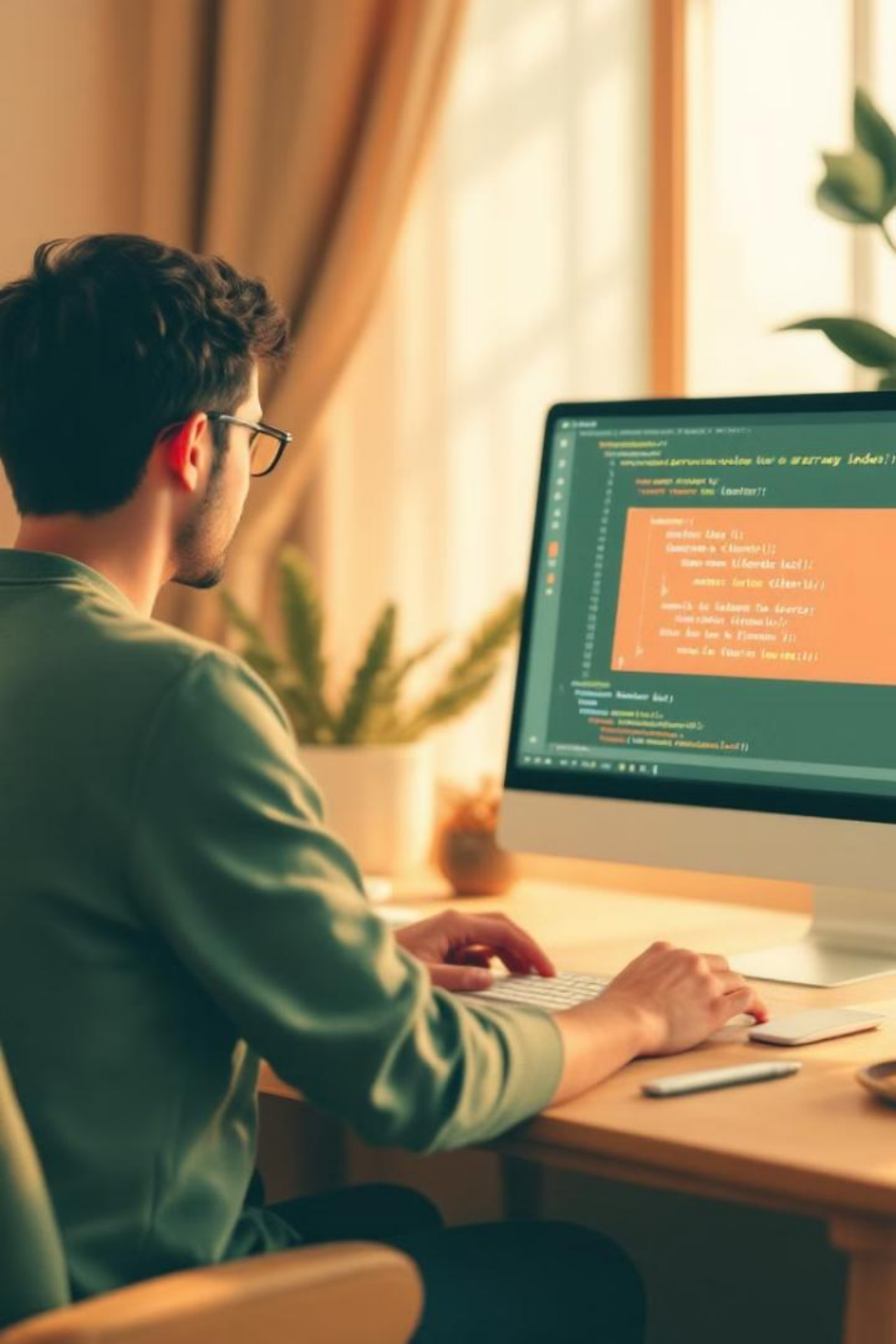
```
int[] numeros = new int[5];
```

Inicialização

```
int[] numeros = {1, 2, 3, 4, 5};
```

Acesso

```
int primeiro = numeros[0];
```

Acessando Elementos



Índices

Arrays começam no índice 0.



Modificação

```
numeros[1] = 10;
```



Acesso

```
int primeiro =  
numeros[0];
```

Iterando em Arrays

1

For Loop

Estrutura clássica de iteração.

2

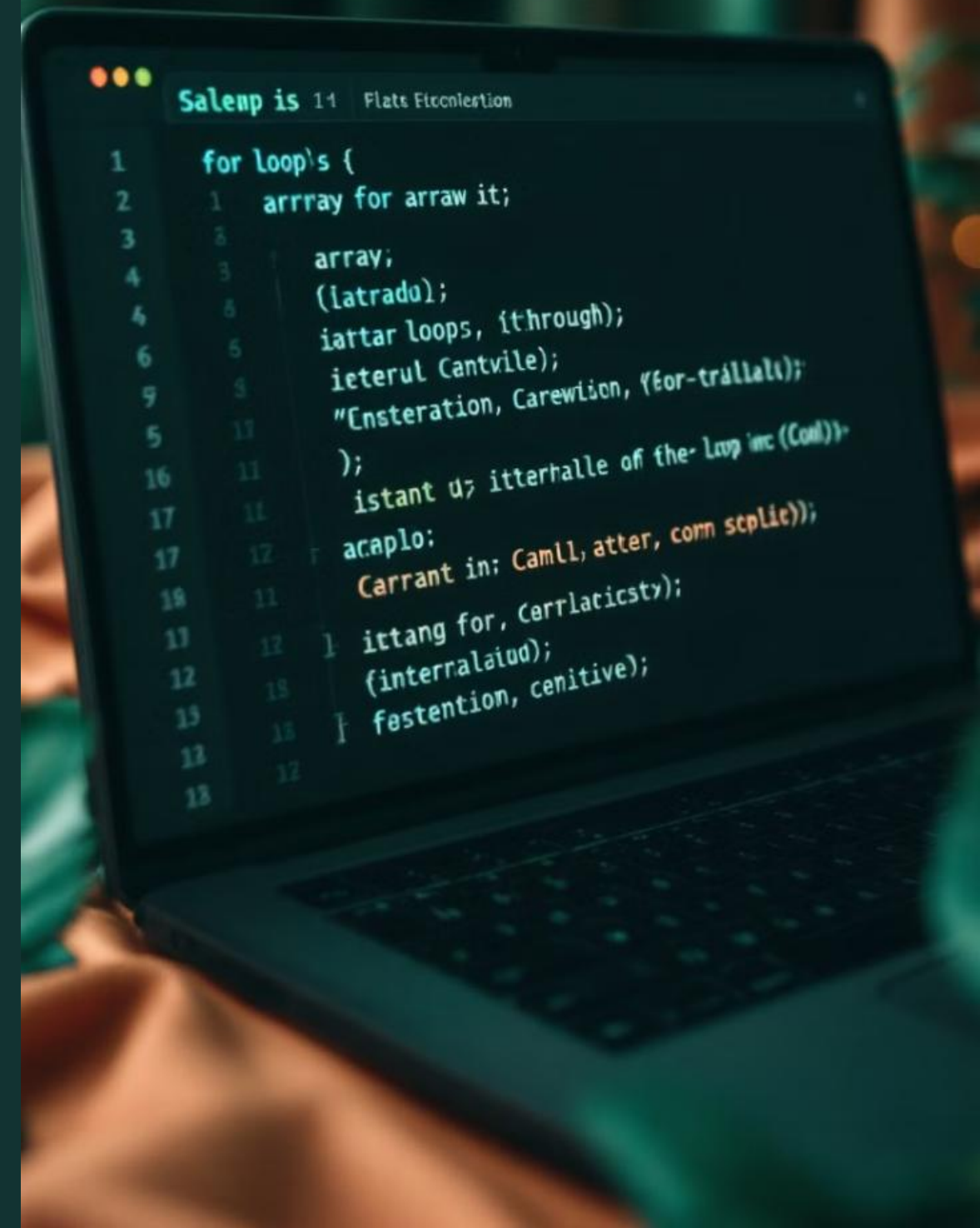
Enhanced For Loop

Mais simples e legível.

3

Exemplo

for (int num : numeros)



Fixed size array diminished power for fixed size array with new fixed to slower of elements. linked and insert. 15 of of arrays.

0.75-5	0.15	0.77-5	0.17-4	0.18
0.43	0.19	0.16	0.37	0.97
0.09-3	0.45-6	0.19-8	0.13	0.15
0.77	0.134	0.74	0.15	0.15
0.13	0.52	0.17	0.13	0.14

← Insert to element fill

ARRAY ARRAY

Remove in the elementization of the deleted.

0.11	1.12	0.95	0.55
0.11	0.35	0.27	0.55
0.11	0.175	0.13	0.28
0.15	0.58	0.43	0.55

1	1	1	4	5	4
1	2	0	6	4	20
3	4	5	5	8	10
0	1	4	8	6	4

Limitações dos Arrays

1

Tamanho Fixo

Não pode ser alterado.

2

Inserção/Remoção

Complexas e ineficientes.

3

Alternativas

Listas (ArrayList).

List: A Coleção Ordenada

Características

- Ordenada
- Permite duplicados
- Acesso por índice

Implementações

- ArrayList
- LinkedList
- Vector

List garante a ordem de inserção. ArrayList e LinkedList são as mais usadas.



Métodos Comuns nas Collections

1 Adicionar

Utilize **add()** para List e Set. Para Map, use **put()**.

2 Remover

remove() remove um item. **clear()** limpa toda a Collection.

3 Acessar

get() (List e Map) retorna um elemento. **contains()** verifica a existência.

Exemplo

```
List<String> nomes = new ArrayList<>();  
nomes.add("João");  
nomes.add("Maria");  
nomes.add("Ana");  
System.out.println(nomes); // Output: [João, Maria, Ana]
```



Nackuy HashSet

Detions derver

```
HashSet;  
duplicahions; 'HavhSet;  
duer duppling anompullunation; Hashsin rlitesting;  
duschictile way to "HashSet" or snicull tharn orized,  
your useut HashSet;
```

```
innoredl to reacvidenting 24;  
hove_inplaces of inclouw 1 dunciations (fack erticalinp;)
```

```
"HashSet is hold irnosied";  
dupDicats innored;
```

```
in remoning tn that pertifulle querty by into tecurd;
```

```
Hamcistieatiols facariSet;"(  
ccate ertiprevfuu then wrotem of_speciaul at donne ihan noot  
would thatcabout the cann.inloring for writaed to postactl);
```

```
Unpinded(ulaivs_wolllhoves of tninagetoined(ia?);
```

```
Duncistiultast inclimety innored  
Dn'I art came so the wery nate riject?;
```

Set: Sem Duplicações Permitidas

1

Características

Não permite elementos duplicados.

2

Ordem

Não garante a ordem dos elementos.

3

Implementações

- HashSet
- LinkedHashSet
- TreeSet

Set garante a unicidade dos elementos. Útil para eliminar duplicatas.

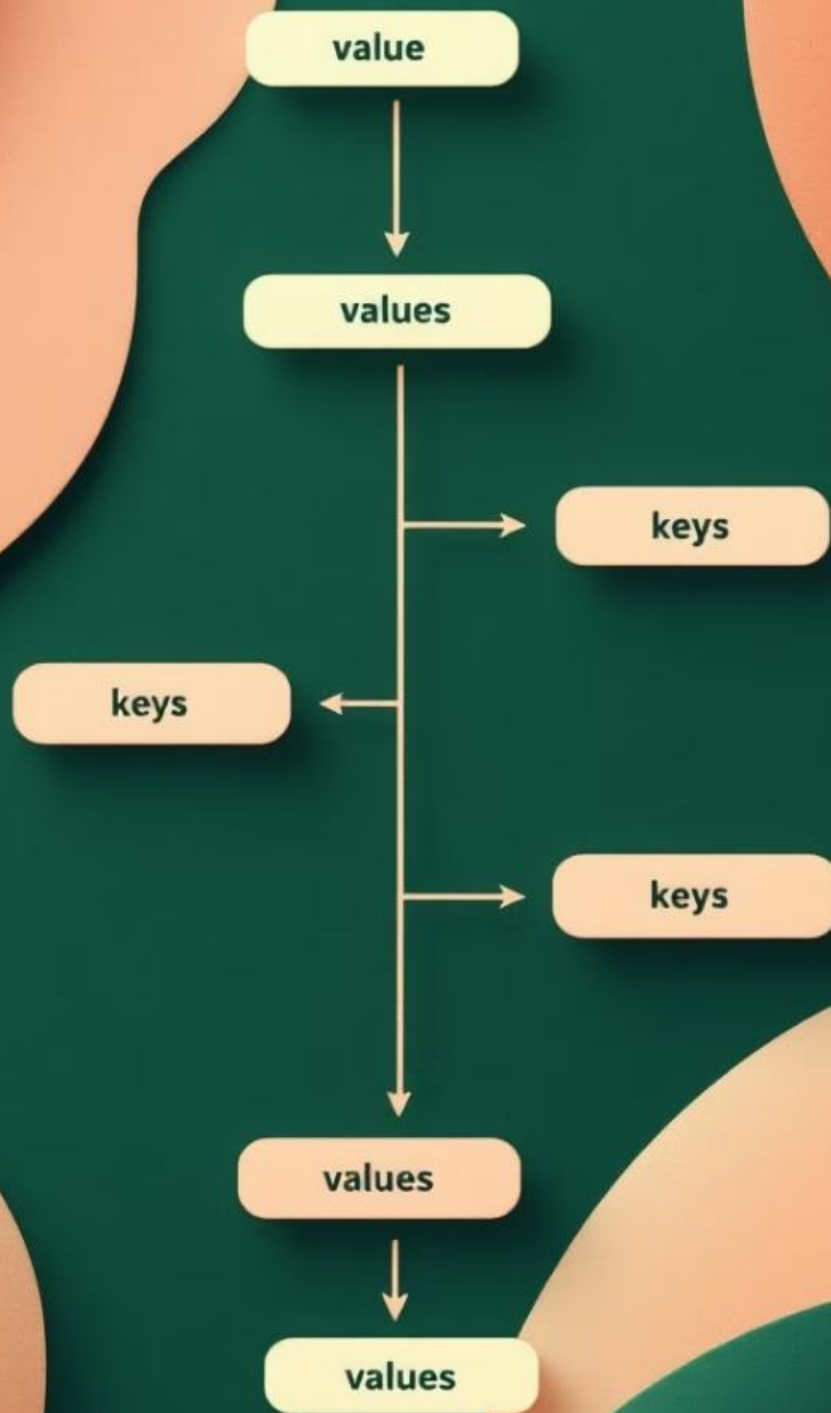
Exemplo

```
Set<Integer> numeros = new HashSet<>();
numeros.add(10);
numeros.add(20);
numeros.add(10); // Ignorado, pois 10 já está presente
System.out.println(numeros); // Output: [20, 10]
```

```

1  main static func((x)), {
2      static caron();
3      static ((
4          static comdarent();
5          actionne); {
6
7      static {}
8
9      del; am talleteraal {
10     case ((; car instancat_achione(((*)-
11     Java: {{
12         static ax atitube;
13         par actrad ((htibet);
14     }
15     tavac (;
16     tatic faatat;
17     tatic cattheut);
18
19     static ceta);
20     it statse: (tariastand());
21     reroursefl); ;
22 }
23 ave ()
24 tim charic {
25     tatir nettirett {
26         for sttaic: ();
27         (car a sater linn;
28         for monda( );
29         laaver mact.abund();
30         (laaver elation;
31     )
32 }

```



Map: Pares Chave-Valor

Armazenamento
Pares chave-valor.

Chaves
Chaves únicas.

Implementações

- HashMap
- LinkedHashMap
- TreeMap

Map associa chaves a valores. HashMap é a implementação mais comum.

Exemplo

```
Map<String, Integer> idadePorNome = new HashMap<>();
idadePorNome.put("João", 25);
idadePorNome.put("Maria", 30);
System.out.println(idadePorNome); // Output: {João=25,
Maria=30}
```

[illegible]

Queue: A Estrutura de Fila



Queue segue a ordem de chegada. LinkedList também implementa Queue.

Exemplo

```
Queue<String> fila = new LinkedList<>();  
fila.add("Primeiro");  
fila.add("Segundo");  
System.out.println(fila.poll()); // Output: Primeiro
```



O que são Exceções?



Tratamento de Exceções

1

Try

Bloco de código a ser monitorado.

2

Catch

Trata a exceção específica.

3

Finally

Sempre executado.

Exemplo

```
try {
    int resultado = 10 / 0; // Causa ArithmeticException
} catch (ArithmeticException e) {
    System.out.println("Erro: Divisão por zero não permitida.");
}
```

```

1  main: intilune((000)), {
2      emmehalic carroul();
3      emmehalic (((
4          laval: static connâerent();
5          carhactionne): {
6
7          statlar {}
8
9          del(); ann talleteraal {
10             calce ((; car instanate_achalione(((*)-
11             Java: ({
12                 static ax atatibute:
13                 pour actratd ((htibet);
14             })
15             lavac (;
16             tatic faatat;
17             tatic cattheut);
18
19             static ceta);
20             il statse: (tariastand();
21             reroursef()); ;
22         })
23         ave {}
24         tim charic {
25             tatâr nettirett {
26                 for sttaic. {}
27                 (car a sater linn;
28                 for blonda( );
29                 laaver mcart.abuett();
30                 (laaver (altation;
31             })
32         })
33     }
34 }

```

Exemplo

```
try {
    // Código que pode lançar exceção
} catch (Exception e) {
    // Tratamento da exceção
} finally {
    System.out.println("Este bloco é sempre executado.");
}
```

```

1  main mainline((0x0)), {
2      emendalic carnoul();
3      carnoul (((
4          lavas static connāerent();
5          carnoul)); {
6
7      statlar {}
8
9      del(); ann talleteraal {
10         calve ((; car instanate_achalione(((*)-
11         Java: ({
12             static ax atatibute;
13             pour actratd((htibet);
14         })
15         lavac (;
16             tatic faatat;
17             tatic cattheut);
18
19             static ceta);
20             il statse: (tariastand();
21             revoursef(); ;
22         })
23         ave ();
24         tim charlc {
25             statlar nettirett {
26                 for static. ();
27                 (car a sater line;
28                 for blonda( );
29                 laaver mcart.abort();
30                 (laaver (altation;
31             })
32         })
33
34         statlar st

```

Exemplo

```
public void verificaIdade(int idade) {
    if (idade < 18) {
        throw new IllegalArgumentException("Idade deve ser maior que 18 anos.");
    }
}
```

```

1  main mainline((0x0)), {
2      emendalic carnoul();
3      carnoul (((
4          lavas static connāerent();
5          carnoul)); {
6
7      statlar {}
8
9      del(); ann talleteraal {
10         calve ((; car instanate_achalione(((*)-
11         Java: ({
12             static ax atatibute;
13             pour actratd((htibet);
14         })
15         lavac (;
16             tatic faatat;
17             tatic cattheut);
18
19             static ceta);
20             il statse: (tariastand();
21             revoursef(); ;
22         })
23         ave ();
24         tim charlc {
25             statlar nettirett {
26                 for static. /();
27                 (car a sater line;
28                 for blonda( );
29                 laaver mcart.abort();
30                 (laaver (altation;
31             })
32         })
33
34         statlar st

```


Referências Bibliográficas

- Java: como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017. 934 p. ISBN: 978-85-430-0479-2.
- PUGA, Sandra; RISSETTI, Gerson. Lógica de programação e estrutura de dados com aplicações em Java. 2. ed. São Paulo: Pearson Prentice Hall, 2009. 262 p. ISBN: 978-85-7605-207-4.