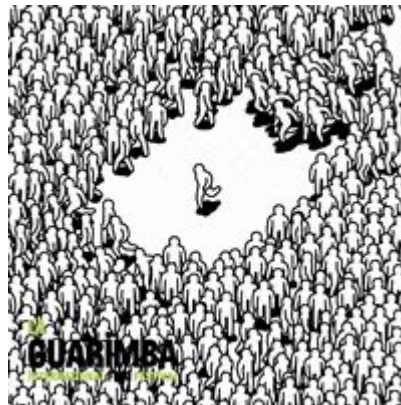


JAVA



VARIAVEIS COMPOSTAS

VETORES E MATRIZES

Variável simples

`int n = 2`



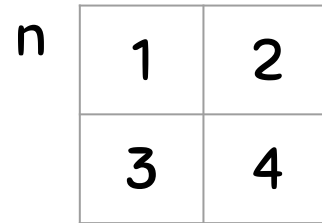
Variável vetor

`int n[] = {1,2,3}`



Variável matriz

`int n[][]
={{1,2},{3,4}}`



VETORES E MATRIZES

Variável vetor

`int n[] = {1,2,3}`

n

1	2	3
---	---	---

Acessando o
valor 2

`N[1] = 2`

`N[1][2] = 2`

Variável matriz

`int n[][]
= {{1,2},{3,4}}`

n

1	2
3	4

VETORES

```
int n[] = {32, 29, 45}
```

n	32	29	45
	0	1	2

$n[0] = 32$

$n[1] = 29$

$n[2] = 45$



MÃO NA MASSA

```
1 package com.company;  
2  
3 public class Main {  
4     public static void main(String[] args) {  
5  
6         // Declarando e inserindo os valores ao mesmo tempo  
7         int numbers[] = {23, 56, 34, 12, 78};  
8  
9         System.out.printf("0 valor inserido no índice 0 é: %d \n", numbers[0]);  
10        System.out.printf("0 valor inserido no índice 3 é: %d", numbers[3]);  
11    }  
12 }  
13 }
```

Run: Main x

```
▶ ↑ /Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java -Djava.library.path=/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/lib/ -Djava.class.path=. Main  
↓ 0 valor inserido no índice 0 é: 23  
↺ 0 valor inserido no índice 3 é: 12  
⇓ Process finished with exit code 0
```



```
1 package com.company;
2
3 public class Main {
4     public static void main(String[] args) {
5
6         // Declarando primeiro
7         int numbers[] = new int[5];
8
9         // Inserindo os valores
10        numbers[0] = 23;
11        numbers[1] = 56;
12        numbers[2] = 34;
13        numbers[3] = 12;
14        numbers[4] = 78;
15
16        System.out.printf("0 valor inserido no índice 0 é: %d \n", numbers[0]);
17        System.out.printf("0 valor inserido no índice 3 é: %d", numbers[3]);
18
19    }
20 }
```

Run: Main ×

```
/Library/Java/JavaVirtualMachines/ac
0 valor inserido no índice 0 é: 23
0 valor inserido no índice 3 é: 12
Process finished with exit code 0
```

```
1 package com.company;
2
3 public class Main {
4     public static void main(String[] args) {
5
6         int numbers[] = {23, 56, 34, 12, 78};
7
8         // Utilizando os metodos do objeto
9         System.out.printf("0 tamanho desse vetor é: %d \n", numbers.length);
10    }
11 }
```

Run: Main

↑ /Library/Java/JavaVirtualMachines/
↓ 0 tamanho desse vetor é: 5


```
3 ▶ public class Main {  
4 ▶     public static void main(String[] args) {  
5  
6         int numbers[] = {23, 56, 34, 12, 78};  
7  
8         // Percorrendo o vetor e imprimindo os valores  
9         for (int count = 0; count <= 4; count++){  
10             System.out.printf("No indice %d temo o valor %d \n", count, numbers[count]);  
11         }  
12     }  
13 }
```

Run: Main ×

▶	↑	/Library/Java/JavaVirtualMachi
■	↓	No indice 0 temo o valor 23
📷	↺	No indice 1 temo o valor 56
🔍	⇅	No indice 2 temo o valor 34
📄	🖨	No indice 3 temo o valor 12
🗑	🗑	No indice 4 temo o valor 78
☰		

```
3 ▶ public class Main {  
4 ▶     public static void main(String[] args) {  
5  
6         int numbers[] = {23, 56, 34, 12, 78};  
7  
8         // Percorrendo o vetor e imprimindo os valores  
9         for (int count = 0; count <= numbers.length - 1; count++){  
10             System.out.printf("No indice %d temo o valor %d \n", count, numbers[count]);  
11         }  
12     }  
13 }
```

Run: Main x

▶	↑	/Library/Java/JavaVirtualMachi
■	↓	No indice 0 temo o valor 23
📷	↺	No indice 1 temo o valor 56
⚙️	⬇️	No indice 2 temo o valor 34
📄	🖨️	No indice 3 temo o valor 12
🗑️	🗑️	No indice 4 temo o valor 78
☰		



***MÃO NA MASSA
COM VOCÊ***

PRÁTICA 01

Declare os seguintes vetores e faça um programa para que seja informado quantos dia tem cada mês.

Ex. "O mês de Janeiro tem 31 dias"

mes[]	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
tot[]	31	28	31	30	31	30	31	31	30	31	30	31
	0	1	2	3	4	5	6	7	8	9	10	11

```
public class Main {
```

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

```
        String mes[] = {"Janeiro", "Fevereiro", "Março", "Abril",  
                        "Maio", "Junho", "Julho", "Agosto", "Setembro",  
                        "Outubro", "Novembro", "Dezembro"};
```

```
        int dias[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

```
        int count;
```

```
        int tamanhoMes = mes.length;
```

```
        // Percorrendo o vetor
```

```
        for (count = 0; count < tamanhoMes; count++){
```

```
            System.out.printf("0 mes de %s tem %d dias \n", mes[count], dias[count]);
```

```
        }
```

Run: Main

/Library/Java/JavaVirtualMachines,

0 mes de Janeiro tem 31 dias

0 mes de Fevereiro tem 28 dias

0 mes de Março tem 31 dias

0 mes de Abril tem 30 dias

0 mes de Maio tem 31 dias

0 mes de Junho tem 30 dias

0 mes de Julho tem 31 dias

0 mes de Agosto tem 31 dias

0 mes de Setembro tem 30 dias

0 mes de Outubro tem 31 dias

0 mes de Novembro tem 30 dias

0 mes de Dezembro tem 31 dias

MELHORANDO NOSSO FOR

ForEach





MÃO NA MASSA

```
3 ▶ public class Main {  
4 ▶     public static void main(String[] args) {  
5  
6         int numbers[] = {23, 56, 34, 12, 78};  
7  
8         // Percorrendo o vetor com forEach  
9         for (int value: numbers){  
10             System.out.printf("O valor é %d \n", value);  
11         }  
12     }  
13 }
```

Run: Main ×

/Library/Java/

0 valor é 23
0 valor é 56
0 valor é 34
0 valor é 12
0 valor é 78

```
5 ▶ public class Main {  
6 ▶     public static void main(String[] args) {  
7  
8         int numbers[] = {23, 56, 34, 12, 78};  
9  
10        //Ordenando o vetor  
11        Arrays.sort(numbers);  
12  
13        // Percorrendo o vetor com forEach  
14        for (int value: numbers){  
15            System.out.printf("0 valor é %d \n", value);  
16        }  
17    }  
18 }
```

Run: Main ×

/Library/Java/Java

0 valor é 12

0 valor é 23

0 valor é 34













0 valor é 56

0 valor é 78

VETORES



Arrays.

 asList (T... a)	List<T>
 binarySearch (int[] a, int key)	int
 binarySearch (byte[] a, byte key)	int
 binarySearch (char[] a, char key)	int
 binarySearch (long[] a, long key)	int
 binarySearch (float[] a, float key)	int
 binarySearch (short[] a, short key)	int
 binarySearch (double[] a, double key)	int
 binarySearch (Object[] a, Object key)	int
 binarySearch (T[] a, T key, Comparator<? super T>...	int
 binarySearch (int[] a, int fromIndex, int toIndex...	int
 binarySearch (byte[] a, int fromIndex, int toIndex...	int

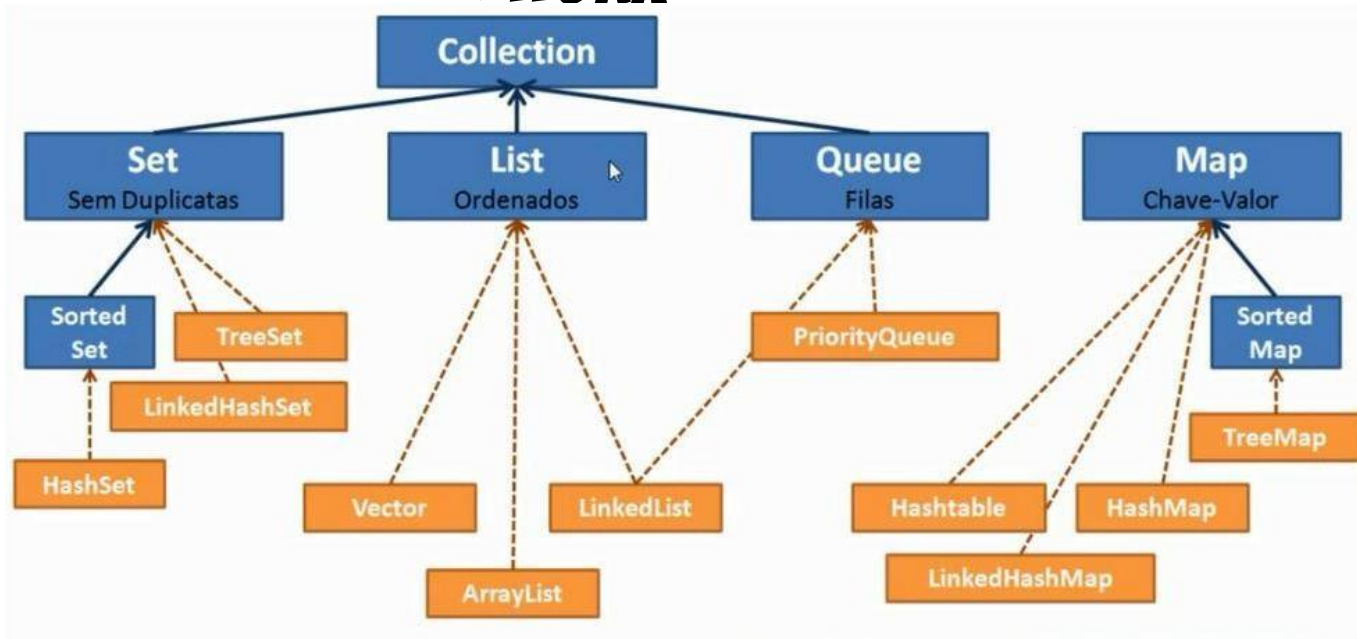
Press ^, to choose the selected (or first) suggestion and insert a dot afterwards [Next Tip](#)  

/Java/javav11tools/binaries/adoptopenjdk-11-jdk/content/home/bin/

COLLECTIONS FRAMEWORK

Collections Framework é um conjunto bem definido de interfaces e classes para representar e tratar grupos de dados como uma única unidade.

COLLECTIONS FRAMEWORK



JAVA COLEÇÕES



COLLECTIONS FRAMEWORK

Ordenada

Classificadas ou sortidas

LIST

List é uma interface e o ArrayList é sua implementação



MÃO NA MASSA

```
6 public class Main {
7     public static void main(String[] args) {
8
9         List names = new ArrayList();
10
11         //Adicionando valores na lista
12         names.add("Tainá");
13         names.add("Ana");
14         names.add("Maria");
15
16         for (Object name: names) {
17             System.out.println("0 nome da vez é " + name);
18         }
19     }
20 }
```

Run: Main x



/Library/Java/JavaVirtua



0 nome da vez é Tainá



0 nome da vez é Ana



0 nome da vez é Maria

```
6 ▶ public class Main {
7 ▶   public static void main(String[] args) {
8     //Definindo o tipo da coleção
9     List<String> names = new ArrayList<>();
10
11     //Adicionando valores na lista
12     names.add("Tainá");
13     names.add("Ana");
14     names.add("Maria");
15
16     for (Object name: names) {
17         System.out.println("O nome da vez é " + name);
18     }
19 }
20 }
```



```
6 ▶ public class Main {
7 ▶   public static void main(String[] args) {
8       //Definindo o tipo da coleção
9       List<String> names = new ArrayList<>();
10
11       //Adicionando valores na lista
12       names.add("Tainá");
13       names.add("Ana");
14       names.add("Maria");
15
16       int count;
17
18       for (count=0; count < names.size(); count++){
19           System.out.println("O nome da vez é " + names.get(count));
20       }
21   }
22 }
```

```
6 ▶ public class Main {
7 ▶     public static void main(String[] args) {
8         //Definindo o tipo da coleção
9         List<String> names = new ArrayList<>();
10        List<String> apelidos = new ArrayList<>();
11
12        //Adicionando valores na lista names
13        names.add("Tainá");
14        names.add("Ana");
15        names.add("Maria");
16
17        //Adicionando valores na lista apelidos
18        apelidos.add("Tai");
19
20        // Juntando conteudo da lista apelidos na lista names
21        names.addAll(apelidos);
22
23        for (String name: names){
24            System.out.println("0 nome da vez é " + name);
25        }
26    }
27 }
```

Run: Main ×

```
0 nome da vez é Tainá
0 nome da vez é Ana
0 nome da vez é Maria
0 nome da vez é Tai
```

```
7 ▶ public class Main {  
8 ▶   public static void main(String[] args) {  
9     //Definindo o tipo da coleção  
10    List<String> names = new ArrayList<>();  
11    List<String> apelidos = new ArrayList<>();  
12  
13    //Adicionando valores na lista names  
14    names.add("Tainá");  
15    names.add("Ana");  
16    names.add("Maria");  
17  
18    // Ordenando a lista  
19    Collections.sort(names);  
20  
21    for (String name: names){  
22      System.out.println("0 nome da vez é " + name);  
23    }  
24  }  
25 }
```

Run: Main x

0 nome da vez é Ana
0 nome da vez é Maria
0 nome da vez é Tainá













```
7 ▶ public class Main {
8 ▶   public static void main(String[] args) {
9     //Definindo o tipo da coleção
10    List<String> names = new ArrayList<>();
11
12    //Adicionando valores na lista names
13    names.add("Tainá");
14    names.add("Ana");
15    names.add("Maria");
16
17    // Ordenando a lista
18    Collections.sort(names);
19
20    // Buscando um valor específico
21    int indice = Collections.binarySearch(names, key: "Tainá");
22
23    System.out.println("O índice foi " + indice);
24  }
25 }
```

Run: Main ×

▶ ↑ /Library/Java/Java
▣ ↓ 0 índice foi 2

LIST

names. |

 addAll (Collection<? extends String> c)	boolean
 add (String e)	boolean
 get (int index)	String
 size ()	int
 add (int index, String element)	void
 addAll (int index, Collection<? extends Strin...	boolean
 remove (Object o)	boolean
 remove (int index)	String
 clear ()	void
 contains (Object o)	boolean
 containsAll (Collection<?> c)	boolean
 equals (Object o)	boolean

Press <? to insert, = to replace [Next Tip](#)

SET

Set é a interface e o HashSet é sua implementação

LIST VS SET

List é uma sequência ordenada de elementos

Set é uma lista distinta de elementos que não é ordenada

LIST<E>

Uma coleção ordenada (também conhecida como sequência).

O usuário desta interface tem controle preciso sobre onde na lista cada elemento é inserido.

O usuário pode acessar elementos por seu índice inteiro (posição na lista) e procurar elementos na lista.

SET<E>

Uma coleção que não contém elementos duplicados.

Mais formalmente, os conjuntos não contêm par de elementos e_1 e e_2 , de modo que $e_1.equals(e_2)$ e, no máximo, um elemento nulo.

Como está implícito no nome, essa interface modela a abstração do conjunto matemático.

	List	Set
Duplicates	YES	NO
Order	ORDERED	DEPENDS ON IMPLEMENTATION
Positional Access	YES	NO



MÃO NA MASSA


```
5 ▶ public class Main {  
6 ▶   public static void main(String[] args) {  
7     //Definindo o tipo da coleção  
8     Set<String> names = new HashSet<>();  
9  
10    //Adicionando valores na lista names  
11    names.add("Tainá");  
12    names.add("Ana");  
13    names.add("Maria");  
14  
15    for(String name: names){  
16      System.out.println("0 da vez é: " + name);  
17    }  
18  }  
19 }
```

Run: Main x

▶	↑	/Library/Java/JavaVir
■	↓	0 da vez é: Tainá
📷	↶	0 da vez é: Ana
🔍	↷	0 da vez é: Maria

```

5 public class Main {
6     public static void main(String[] args) {
7         //Definindo o tipo da coleção
8         Set<String> names = new HashSet<>();
9
10        //Adicionando valores na lista names
11        names.add("Tainá");
12        names.add("Ana");
13        names.add("Maria");
14
15        for(String name: names){
16            System.out.println("0 da vez é: " + name);
17        }
18
19        // Adicionando um novo valor na lista
20        names.add("Gisele");
21
22        System.out.println("Nova lista!!!!");
23        for(String name: names){
24            System.out.println("0 da vez é: " + name);
25        }
26    }
27 }

```

Run: Main x

```

/Library/Java/JavaVirtualMachines/
0 da vez é: Tainá
0 da vez é: Ana
0 da vez é: Maria
Nova lista!!!!
0 da vez é: Tainá
0 da vez é: Ana
0 da vez é: Gisele
0 da vez é: Maria

```

```

5 ▶ public class Main {
6 ▶     public static void main(String[] args) {
7         //Definindo o tipo da coleção
8         Set<String> names = new LinkedHashSet<>();
9
10        //Adicionando valores na lista names
11        names.add("Tainá");
12        names.add("Ana");
13        names.add("Maria");
14
15        for(String name: names){
16            System.out.println("0 da vez é: " + name);
17        }
18
19        // Adicionando um novo valor na lista
20        names.add("Gisele");
21
22        System.out.println("Nova lista!!!!");
23        for(String name: names){
24            System.out.println("0 da vez é: " + name);
25        }
26    }
27 }

```

Run: Main ×

```

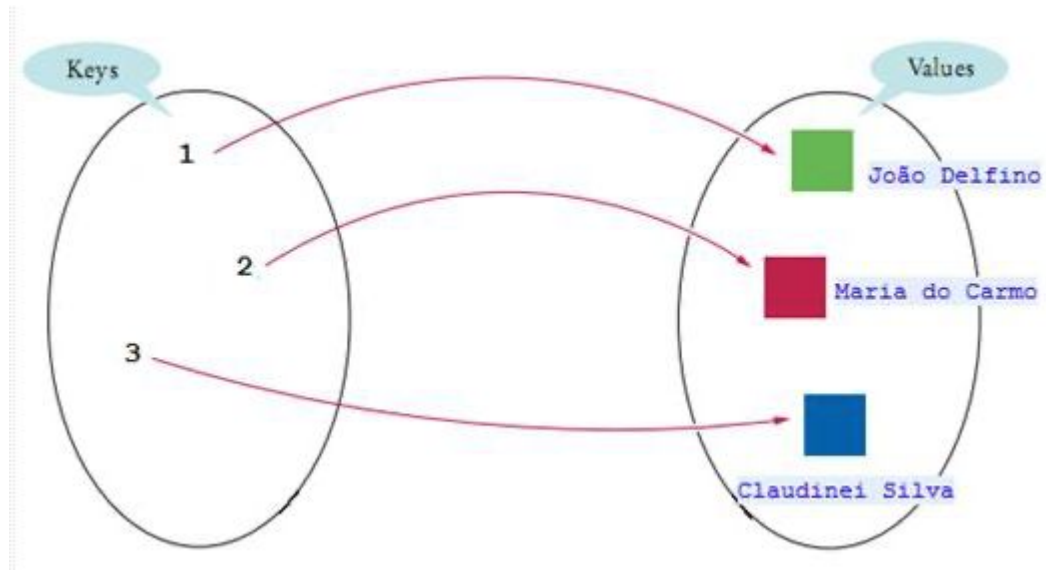
/Library/Java/JavaVirt
0 da vez é: Tainá
0 da vez é: Ana
0 da vez é: Maria
Nova lista!!!!
0 da vez é: Tainá
0 da vez é: Ana
0 da vez é: Maria
0 da vez é: Gisele

```

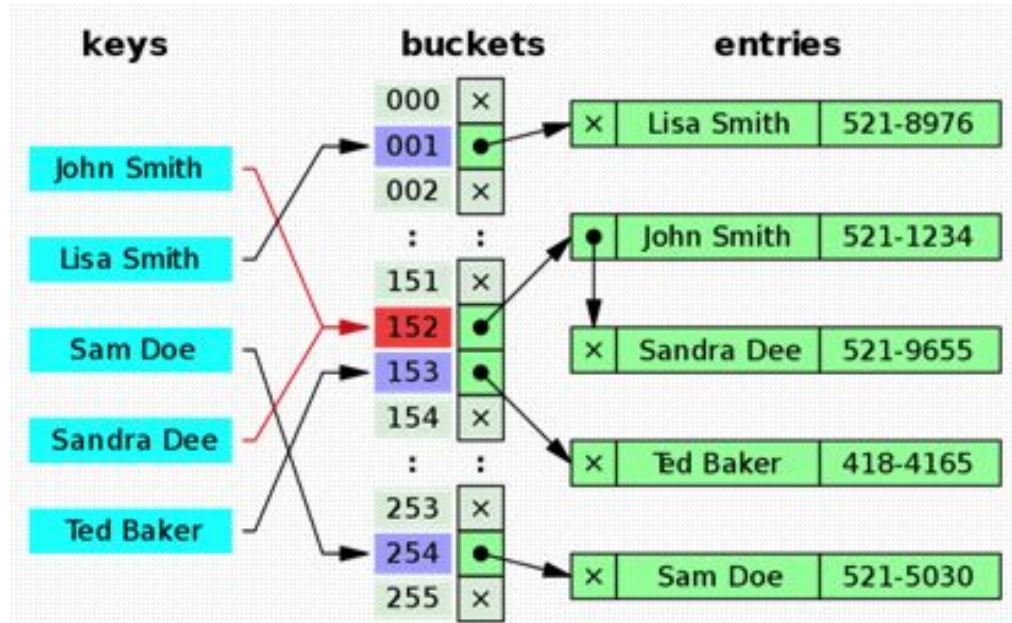
MAP

Essa interface é um objeto que mapeia valores para chaves, ou seja, através da chave consegue ser acessado o valor configurado, sendo que a chave não pode ser repetida ao contrário do valor, mas se caso tiver uma chave repetida é sobrescrito pela última chamada.

MAP



MAP





MÃO NA MASSA

```
5 ▶ public class Main {  
6 ▶   public static void main(String[] args) {  
7       //Definindo o tipo da coleção  
8       Map<String,String> relacao = new HashMap<>();  
9  
10      // Adicionando valores (Key, value)  
11      relacao.put("Mae", "Filha");  
12      relacao.put("Pai", "Filho");  
13      relacao.put("Irmão", "Irmã");  
14  
15      //Percorrer esse map pela key  
16      for (String key : relacao.keySet()){  
17          System.out.println("A chave da vez é: " + key);  
18      }  
19  }  
20 }
```

Run: Main ×

```
▶ /Library/Java/JavaVirtual  
↓ A chave da vez é: Mae  
↺ A chave da vez é: Pai  
⇓ A chave da vez é: Irmão
```

```
5 ▶ public class Main {  
6 ▶     public static void main(String[] args) {  
7         //Definindo o tipo da coleção  
8         Map<String,String> relacao = new HashMap<>();  
9  
10        // Adicionando valores (Key, value)  
11        relacao.put("Mae", "Filha");  
12        relacao.put("Pai", "Filho");  
13        relacao.put("Irmão", "Irmã");  
14  
15        //Percorrer esse map pelo value  
16        for (String key : relacao.values()){  
17            System.out.println("O valor da vez é: " + key);  
18        }  
19    }  
20 }
```

Run: Main x

```
0 valor da vez é: Filha  
0 valor da vez é: Filho  
0 valor da vez é: Irmã
```



```

5 ▶ public class Main {
6 ▶     public static void main(String[] args) {
7         //Definindo o tipo da coleção
8         Map<String,String> relacao = new HashMap<>();
9
10        // Adicionando valores (Key, value)
11        relacao.put("Mae", "Filha");
12        relacao.put("Pai", "Filho");
13        relacao.put("Irmão", "Irmã");
14
15        //Percorrer esse map pelo value
16        for (String key : relacao.values()){
17            System.out.println("0 valor da vez é: " + key);
18        }
19
20        // Add valor um novo elemento com chave existente
21        relacao.put("Irmão", "Primos");
22
23        System.out.println("Nova Lista!!!!");
24        for (String key : relacao.values()){
25            System.out.println("0 valor da vez é: " + key);
26        }
27    }
28 }

```

Run: Main ×

```

/Library/Java/JavaVirtualM
0 valor da vez é: Filha
0 valor da vez é: Filho
0 valor da vez é: Irmã
Nova Lista!!!!
0 valor da vez é: Filha
0 valor da vez é: Filho
0 valor da vez é: Primos

```



```
5 ▶ public class Main {  
6 ▶     public static void main(String[] args) {  
7         //Definindo o tipo da coleção  
8         Map<String,String> relacao = new HashMap<>();  
9  
10        // Adicionando valores (Key, value)  
11        relacao.put("Mae", "Filha");  
12        relacao.put("Pai", "Filho");  
13        relacao.put("Irmão", "Irmã");  
14  
15        //Percorrer esse map pela key e value  
16        for (Map.Entry<String, String> entry : relacao.entrySet()){  
17            System.out.println("A chave e valor da vez é: " + entry.getKey() + " e " + entry.getValue());  
18        }  
19    }  
20 }  
21 }
```

Run: Main ×

▶	↑	/Library/Java/JavaVirtualMachines/adopt
■	↓	A chave e valor da vez é: Mae e Filha
📷	↺	A chave e valor da vez é: Pai e Filho
🔧	⇅	A chave e valor da vez é: Irmão e Irmã

```

3 public class Main {
6     public static void main(String[] args) {
7         //Definindo o tipo da coleção
8         Map<String,String> relacao = new HashMap<>();
9
10        // Adicionando valores (Key, value)
11        relacao.put("Mae", "Filha");
12        relacao.put("Pai", "Filho");
13        relacao.put("Irmão", "Irmã");
14
15        //Percorrer esse map pela key e value
16        for (Map.Entry<String, String> entry : relacao.entrySet()){
17            System.out.println("A chave e valor da vez é: " + entry.getKey() + " e " + entry.getValue());
18        }
19
20        //Adicionando novo valor valido
21        relacao.put("Primas", "Primos");
22
23        System.out.println("Nova LISTA!!!!!!");
24        for (Map.Entry<String, String> entry : relacao.entrySet()){
25            System.out.println("A chave e valor da vez é: " + entry.getKey() + " e " + entry.getValue());
26        }
27    }
28 }
29 }

```

```

Run: Main x
  ↑ /Library/Java/JavaVirtualMachines/adoptopen
  ↓ A chave e valor da vez é: Mae e Filha
  ↕ A chave e valor da vez é: Pai e Filho
  ↕ A chave e valor da vez é: Irmão e Irmã
  ↕ Nova LISTA!!!!!!
  ↕ A chave e valor da vez é: Mae e Filha
  ↕ A chave e valor da vez é: Primas e Primos
  ↕ A chave e valor da vez é: Pai e Filho
  ↕ A chave e valor da vez é: Irmão e Irmã

```

```
//Definindo o tipo da coleção
```

```
Map<String,String> relacao = new LinkedHashMap<>();
```

```
Run: Main x
↑ /Library/Java/JavaVirtualMachines/adoptopen
↓ A chave e valor da vez é: Mae e Filha
↵ A chave e valor da vez é: Pai e Filho
⇓ A chave e valor da vez é: Irmão e Irmã
☐ Nova LISTA!!!!!!
☑ A chave e valor da vez é: Mae e Filha
☒ A chave e valor da vez é: Pai e Filho
☐ A chave e valor da vez é: Irmão e Irmã
☑ A chave e valor da vez é: Primas e Primos
```

QUEUE

É um tipo de coleção para manter uma lista de prioridades.

Com a interface Queue pode-se criar filas e pilhas;

QUEUE

É um tipo de coleção para manter uma lista de prioridades.

Com a interface Queue pode-se criar filas e pilhas;



MÃO NA MASSA

```
5 ▶ public class Main {  
6 ▶   public static void main(String[] args) {  
7     //Definindo o tipo da coleção  
8     Queue<String> names = new PriorityQueue<>();  
9  
10    // Adicionando valores  
11    names.add("Tainá");  
12    names.add("Ana");  
13    names.add("Maria");  
14  
15    //Percorrer a priorityQueue  
16    System.out.println("Essa fila é: " + names);  
17  }  
18 }
```

Run: Main ×

↑ /Library/Java/JavaVirtualMachines/a
↓ Essa fila é: [Ana, Tainá, Maria]
⏏

```
5 ▶ public class Main {  
6 ▶     public static void main(String[] args) {  
7         //Definindo o tipo da coleção  
8         Queue<String> names = new PriorityQueue<>();  
9  
10        // Adicionando valores  
11        names.add("Tainá");  
12        names.add("Ana");  
13        names.add("Maria");  
14  
15        //Percorrer a priorityQueue  
16        System.out.println("Essa fila é: " + names);  
17  
18        //Identificar o elemento da vez da fila  
19        System.out.println("Elemento da vez é: " + names.peek());  
20    }  
21 }
```

Run: Main ×



```
/Library/Java/JavaVirtualMachines/  
Essa fila é: [Ana, Tainá, Maria]  
Elemento da vez é: Ana
```

```

5 ▶ public class Main {
6 ▶   public static void main(String[] args) {
7       //Definindo o tipo da coleção
8       Queue<String> names = new PriorityQueue<>();
9
10      // Adicionando valores
11      names.add("Tainá");
12      names.add("Ana");
13      names.add("Maria");
14
15      //Percorrer a priorityQueue
16      System.out.println("Essa fila é: " + names);
17
18      //Identificar o elemento da vez da fila
19      System.out.println("Elemento da vez é: " + names.poll());
20
21      System.out.println("Essa fila é: " + names);
22  }
23 }

```

Run: Main x

```

Essa fila é: [Ana, Tainá, Maria]
Elemento da vez é: Ana
Essa fila é: [Maria, Tainá]

```

```
5 ▶ public class Main {
6 ▶   public static void main(String[] args) {
7       //Definindo o tipo da coleção
8       Stack<String> names = new Stack<>();
9
10      // Adicionando valores
11      names.push( item: "Tainá");
12      names.push( item: "Ana");
13      names.push( item: "Maria");
14
15      //Percorrer a pilha
16      System.out.println("Essa fila é: " + names);
17
18      //Identificar o elemento da vez da pilha
19      System.out.println("Elemento da vez é: " + names.pop());
20
21      System.out.println("Essa fila é: " + names);
22  }
23 }
```

Run: Main ×



```
/Library/Java/JavaVirtualMachines/a
Essa fila é: [Tainá, Ana, Maria]
Elemento da vez é: Maria
Essa fila é: [Tainá, Ana]
```






PRÁTICA 02

Escreva um programa que adote um ArrayList como estrutura padrão de armazenamento de dados.

O programa deverá cadastrar (sem interação do usuário), 10 valores de qualquer tipo dentro da lista (String, int, char, double, etc).

Ao final, o programa deverá mostrar os valores cadastrados.

PRÁTICA 03

Escreva um programa que receba cinco nomes diferentes do usuário.

Todos os nomes deverão ser armazenados em um ArrayList tipado para Strings.

O programa deverá mostrar os nomes cadastrados em ordem inversa a qual foram cadastrados, ou seja, do último para o primeiro.

PRÁTICA 04

Escreva um programa que contenha uma lista com 5 nomes pré-cadastrados.

O programa deverá dar ao usuário a opção de excluir um único nome da lista, com valores entre 1 e 5, exemplo ao lado.

Feita a escolha do usuário pelo número correspondente ao nome, o programa deverá mostrar na tela os quatro nomes que restaram cadastrados na lista

Qual dos nomes abaixo você deseja excluir da lista?

1. Tainá
2. Stellinha
3. Tequillina
4. Pituzinho
5. <Seu Nome>

PRÁTICA 05

Escreva um programa que cadastra em uma lista numérica de valores inteiros 10 números inteiros aleatórios entre 10 e 100.

Ao final, o programa deverá detectar na lista qual o maior número inteiro sorteado e mostrá-lo ao usuário.

THE END