# Estrutura de Dados e Algoritmos com Java

# FILAS (QUEUES)

<loiane.training />

19

# Estrutura de Dados e Algoritmos com Java

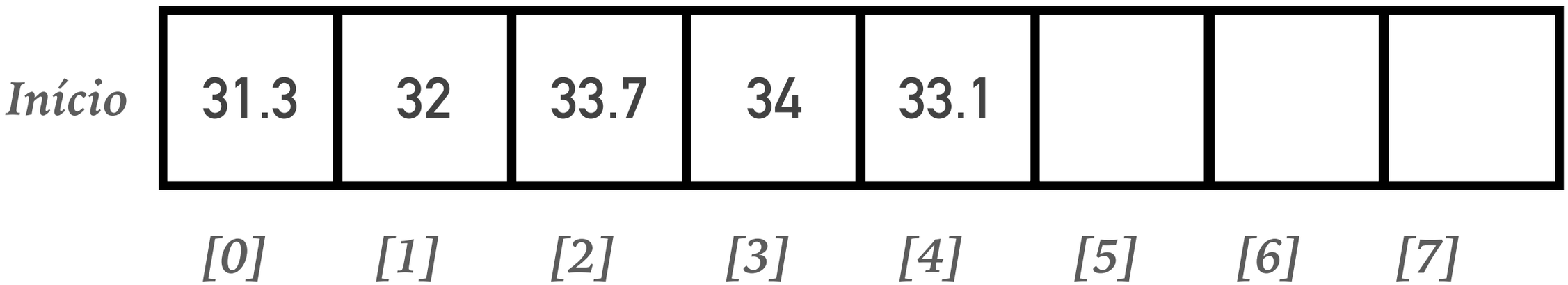
<loiane.training />

# FILAS (QUEUES)

*Introdução*


http://baianolandia.blog.br/wp-content/uploads/2016/03/fila-de-banco.jpg

# *Vetor, Array, Lista*




$tamanho = 5$

# FIFO

# *FIFO*

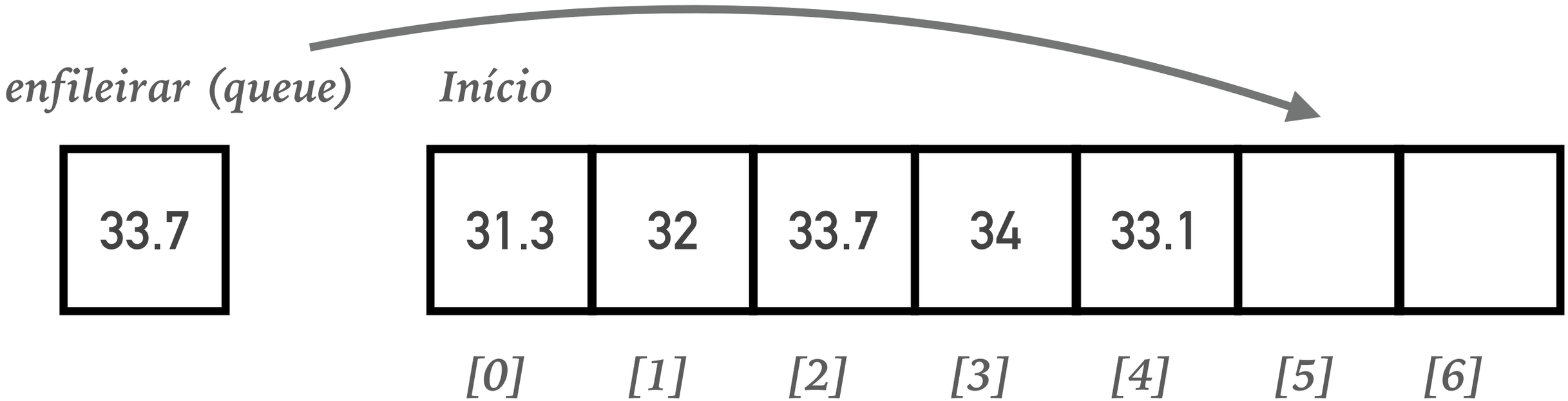
## *First In First Out*

# *FIFO*

## *First In First Out*

*Primeiro a entrar, primeiro a sair*

# *Fila*




$tamanho = 5$

# *Fila*

*último elemento adicionado*

*desenfileirar (dequeue)*

*Início*

| 31.3 | 32 | 33.7 | 34 | 33.1 | 33.7 | |
|---|---|---|---|---|---|---|
| *[0]* | *[1]* | *[2]* | *[3]* | *[4]* | *[5]* | *[6]* |

$$tamanho = 6$$

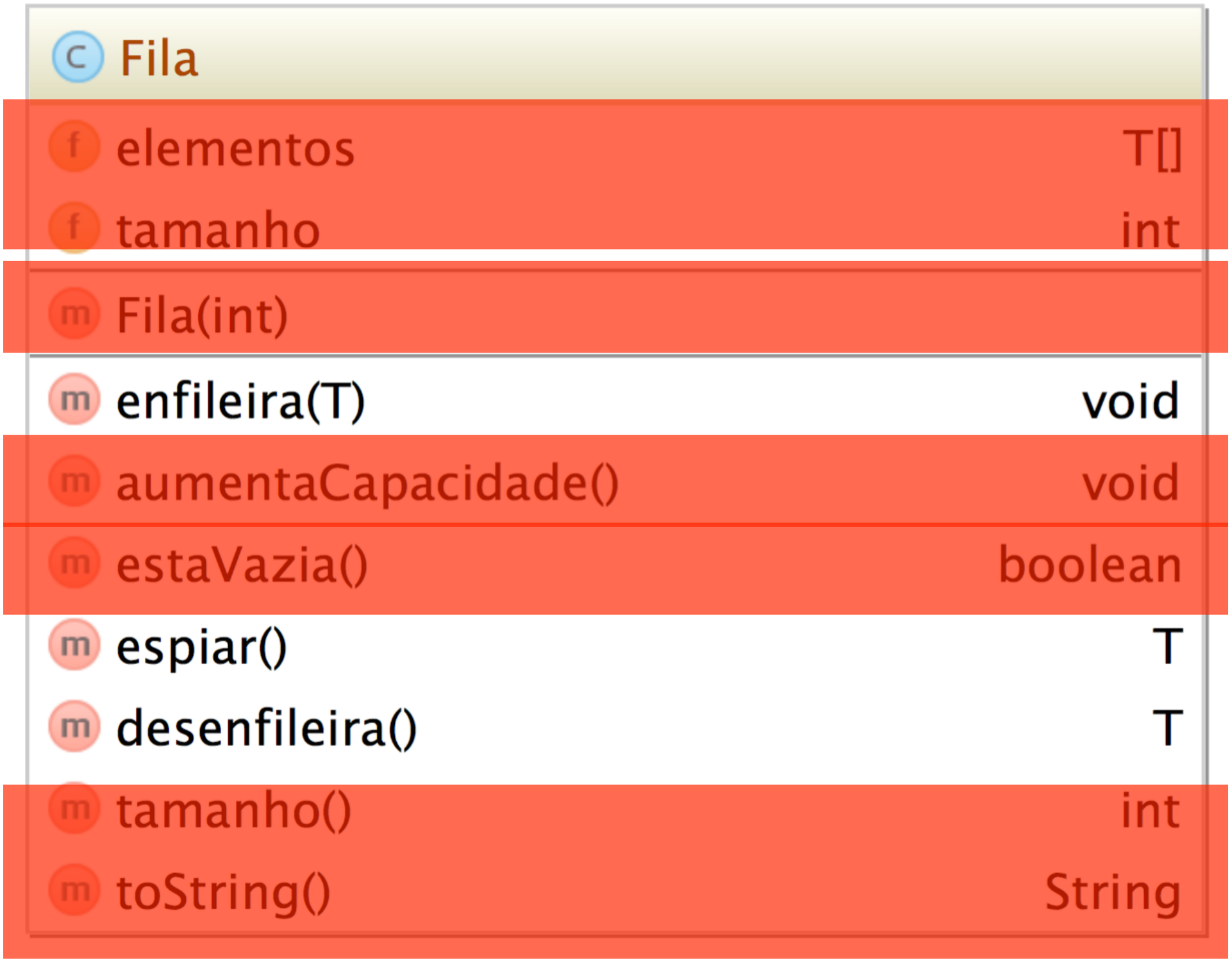
# FILAS (QUEUES): INTRODUÇÃO

- Classe Fila
  - Definição
  - Enfileirar elemento
  - Espiar/Verificar elemento início da pilha
  - Desenfileirar elemento da pilha
  - API Java Queue
  - Filas com prioridade
  - API Java PriorityQueue
  - Exercícios

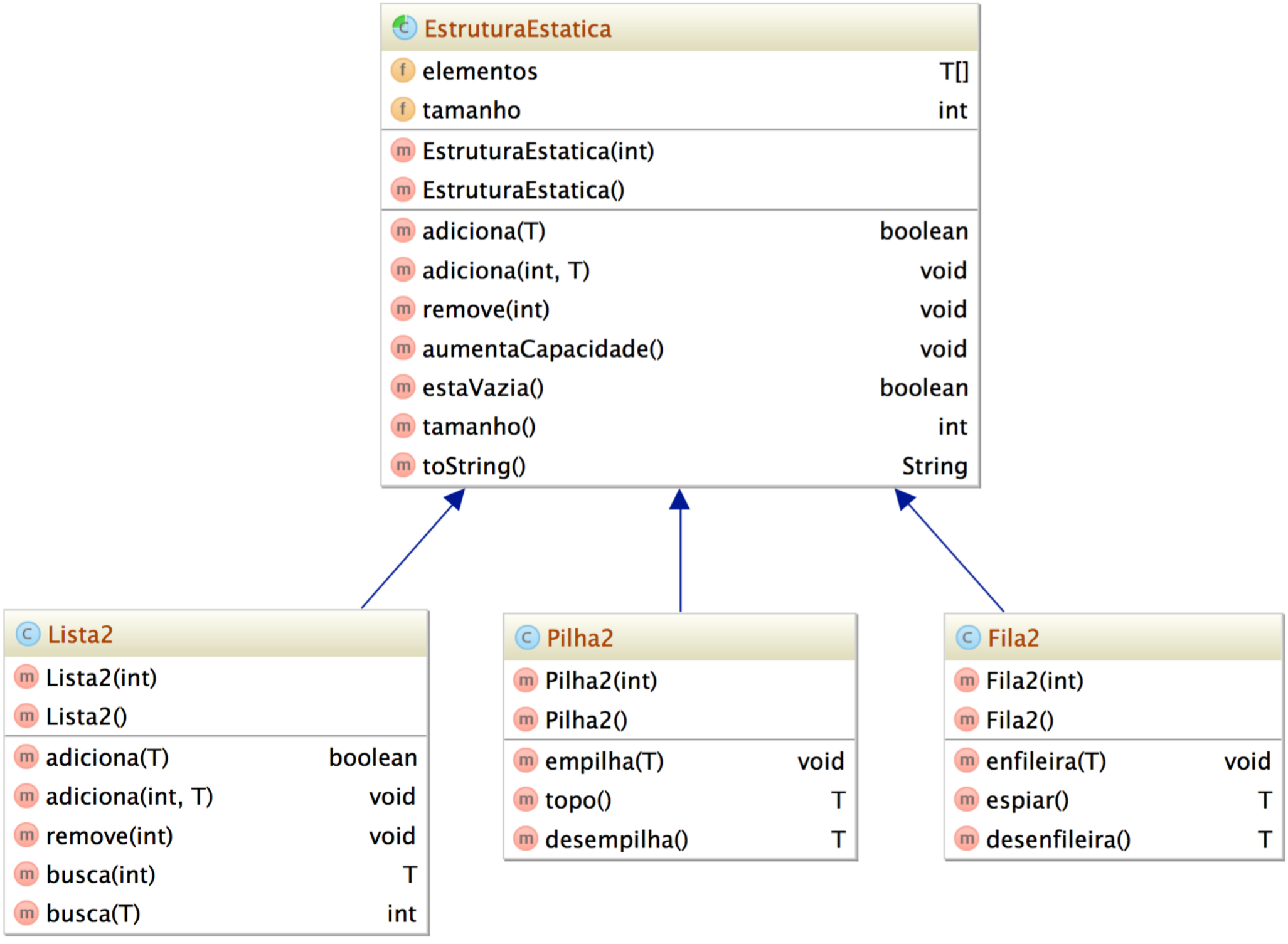
# DESAFIO:
# SABENDO DO COMPORTAMENTO DA PILHA E FILA, TENTAR DESENVOLVER A CLASSE FILA

# DIAGRAMA CLASSE FILA

# DIAGRAMA CLASSE FILA



# USANDO A SUPER CLASSE



# CLASSE FILA

```
public class Fila<T> extends EstruturaEstatica<T>{

  public Fila(int capacidade) {
    super(capacidade);
  }

  public Fila() {
    super();
  }
}
```

# TESTE

```
Fila<Integer> fila = new Fila<>();
System.out.println(fila.estaVazia());
System.out.println(fila.tamanho());
```

# FILAS (QUEUES): INTRODUÇÃO

- Classe Fila
  - Definição ✅
  - Enfileirar elemento
  - Espiar/Verificar elemento início da pilha
  - Desenfileirar elemento da pilha
  - API Java Queue
  - Filas com prioridade
  - API Java PriorityQueue
  - Exercícios

20

# Estrutura de Dados e Algoritmos com Java

<loiane.training />

# FILAS (QUEUES)

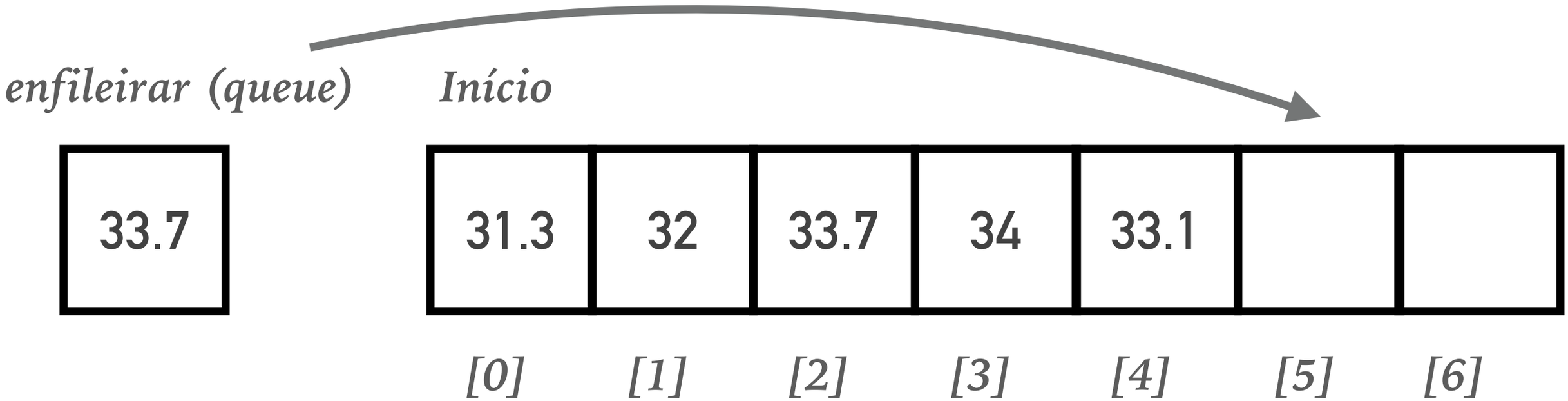
*Enfileirar elemento*

# *FIFO*

## *First In First Out*

*Primeiro a entrar, primeiro a sair*

# *Fila*




$tamanho = 5$

# ENFILEIRAR (ADICIONAR ELEMENTO)

```
public void enfileira(T elemento) {
  aumentaCapacidade();
  elementos[tamanho] = elemento;
  tamanho++;
  //elementos[tamanho++] = elemento;
}
```

# ENFILEIRAR (ADICIONAR ELEMENTO) - REUSANDO CÓDIGO

```
public void enfileira(T elemento) {
  super.adiciona(elemento);
}
```

Mesmo código de add elemento na pilha

# TESTE

```java
Fila<Integer> fila = new Fila<>();

fila.enfileira(3);
fila.enfileira(1);
fila.enfileira(2);

System.out.println(fila.estaVazia()); //false
System.out.println(fila.tamanho()); //3

System.out.println(fila); //[3, 1, 2]
```

# FILAS (QUEUES): INTRODUÇÃO

- Classe Fila
  - Definição ✅
  - Enfileirar elemento ✅
  - Espiar/Verificar elemento início da pilha
  - Desenfileirar elemento da pilha
  - API Java Queue
  - Filas com prioridade
  - API Java PriorityQueue
  - Exercícios

21

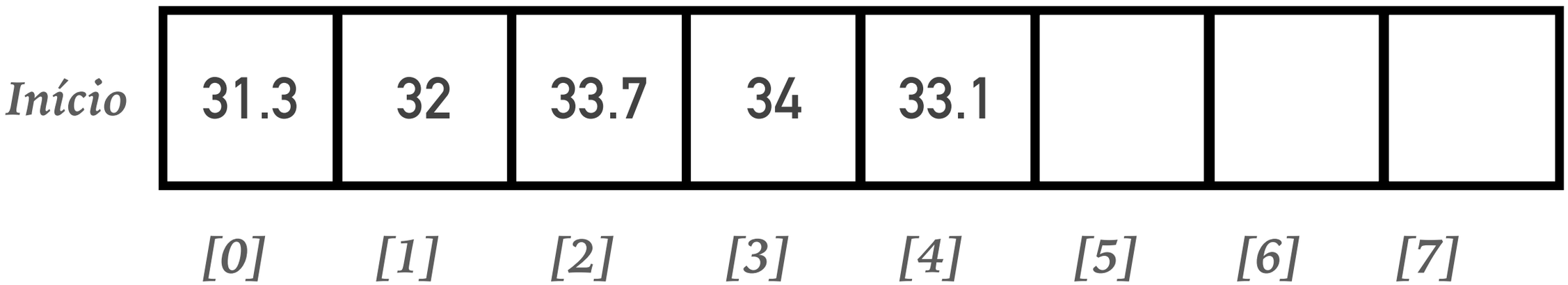
# Estrutura de Dados e Algoritmos com Java

<loiane.training />

## FILAS (QUEUES)

*Espiar início da fila*

# *Fila*




$tamanho = 5$

# ESPIAR INÍCIO DA FILA (PEEK)

```
public T espiar(){
  if (this.estaVazia()){
    return null;
  }
  return elementos[0];
}
```

# TESTE

```java
Fila<Integer> fila = new Fila<>();

fila.enfileira(3);
fila.enfileira(1);
fila.enfileira(2);

System.out.println(fila.espiar()); //3
```

# FILAS (QUEUES): INTRODUÇÃO

- Classe Fila
  - Definição ✅
  - Enfileirar elemento ✅
  - Espiar/Verificar elemento início da fila ✅
  - Desenfileirar elemento da fila
  - API Java Queue
  - Filas com prioridade
  - API Java PriorityQueue
  - Exercícios

22

# Estrutura de Dados e Algoritmos com Java

<loiane.training />

# FILAS (QUEUES)

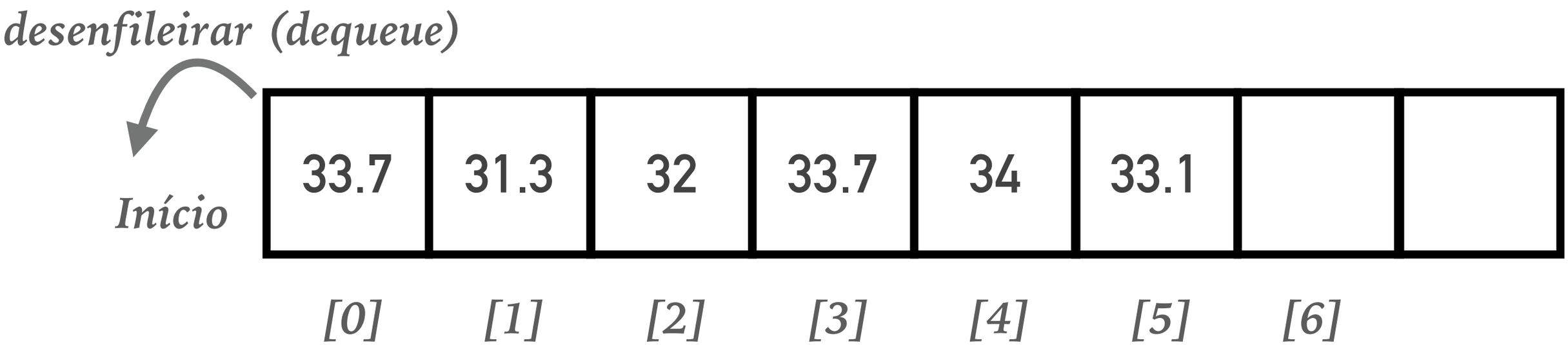
*Desenfileirar Elemento*

# *FIFO*

## *First In First Out*

*Primeiro a entrar, primeiro a sair*

# *Fila*




$tamanho = 6$

# DESENFILEIRAR ELEMENTO (DEQUEUE)

```
public T desenfileira(){
  if (this.estaVazia()){
    return null;
  }
  T objetoRemovido = this.elementos[0];
  this.remove(0);
  return objetoRemovido;
}
```

# TESTE

```java
Fila<Integer> fila = new Fila<>();

fila.enfileira(3);
fila.enfileira(1);
fila.enfileira(2);

System.out.println(fila.desenfileira()); //3

System.out.println(fila.desenfileira()); //1
```

# FILAS (QUEUES): INTRODUÇÃO

- Classe Fila
  - Definição ✅
  - Enfileirar elemento ✅
  - Espiar/Verificar elemento início da fila ✅
  - Desenfileirar elemento da fila ✅
  - API Java Queue
  - Filas com prioridade
  - API Java PriorityQueue
  - Exercícios

23

# Estrutura de Dados e Algoritmos com Java

<loiane.training />

# FILAS (QUEUES)

*API Java Queue + LinkedList*

# TESTE

```
Queue<Integer> fila = new LinkedList<>();

fila.add(3);
fila.add(2);
fila.add(1);

System.out.println(fila);

fila.remove();

System.out.println(fila);

System.out.println(fila.peek());
```

# FILAS (QUEUES): INTRODUÇÃO

- Classe Fila
  - Definição ✅
  - Enfileirar elemento ✅
  - Espiar/Verificar elemento início da fila ✅
  - Desenfileirar elemento da fila ✅
  - API Java Queue ✅
  - Filas com prioridade
  - API Java PriorityQueue
  - Exercícios

24

# Estrutura de Dados e Algoritmos com Java

<loiane.training />

# FILAS (QUEUES)

*Filas com prioridade*

# FILA COM PRIORIDADE

```
public class FilaComPrioridade<T> extends Fila<T>{

  public void enfileira(T elemento) {

    Comparable<T> chave = (Comparable<T>) elemento;
    int i;
    for (i=0; i<this.tamanho; i++){
      if (chave.compareTo(this.elementos[i]) < 0) {
        break;
      }
    }

    super.adiciona(i, elemento);
  }
}
```

# TESTE

```
public class Pessoa implements Comparable<Pessoa>{

   private String nome;
   private int prioridade;

   @Override
   public int compareTo(Pessoa o) {
      if (prioridade > o.prioridade){
         return 1;
      } else if (prioridade < o.prioridade){
         return -1;
      }

      return 0;
   }

}
```

# TESTE

```java
FilaComPrioridade<Pessoa> fila = new FilaComPrioridade<>();

fila.enfileira(new Pessoa("C", 3));
fila.enfileira(new Pessoa("D", 7));

System.out.println(fila);

fila.enfileira(new Pessoa("B", 2));
System.out.println(fila);

fila.enfileira(new Pessoa("A", 1));
System.out.println(fila);

fila.enfileira(new Pessoa("F", 8));
System.out.println(fila);
```

# FILAS (QUEUES): INTRODUÇÃO

- Classe Fila
  - Definição ✅
  - Enfileirar elemento ✅
  - Espiar/Verificar elemento início da fila ✅
  - Desenfileirar elemento da fila ✅
  - API Java Queue ✅
  - Filas com prioridade ✅
  - API Java PriorityQueue
  - Exercícios

25

# Estrutura de Dados e Algoritmos com Java

<loiane.training />

# FILAS (QUEUES)

*API Java PriorityQueue*

# TESTE

```
Queue<Pessoa> fila = new PriorityQueue<>(new Comparator<Pessoa>() {
    public int compare(Pessoa p1, Pessoa p2) {
          return Integer.valueOf(p1.getPrioridade()).compareTo(p2.getPrioridade());
      }
});

fila.add(new Pessoa("C", 3));
fila.add(new Pessoa("D", 7));

System.out.println(fila);

fila.add(new Pessoa("B", 2));
System.out.println(fila);

fila.add(new Pessoa("A", 1));
System.out.println(fila);

fila.add(new Pessoa("F", 8));
System.out.println(fila);
```

# FILAS (QUEUES): INTRODUÇÃO

- Classe Fila
  - Definição ✅
  - Enfileirar elemento ✅
  - Espiar/Verificar elemento início da fila ✅
  - Desenfileirar elemento da fila ✅
  - API Java Queue ✅
  - Filas com prioridade ✅
  - API Java PriorityQueue ✅
  - Exercícios

# *EXERCÍCIOS*

*http://goo.gl/fFQCDD*

https://github.com/loiane/estrutura-dados-algoritmos-java


Estrutura de Dados

# Estrutura de Dados e Algoritmos com Java

GRÁTIS

INICIAR CURSO

HOME

GRÁTIS

* REQUER JAVA BÁSICO

ACESSO ILIMITADO

CERTIFICADO DO CURSO

## O QUE VAMOS APRENDER NESSE CURSO?

- Vetores (Arrays)
- Pilhas (Stacks)
- Filas (Queues)
- Listas Encadeadas (Linked Lists)
- Listas Duplamente Encadeadas (Doubly-Linked Lists)
- Conjuntos (Sets)
- Tabelas de Hashing (HashTables)
- Árvores (Trees)
- Grafos (Graphs)
- Algoritmos de Ordenação:
  - Bolha (Bubble Sort)

# Download código fonte e certificado Cadastro em:

# http://loiane.training

Obrigada

http://loiane.training


http://loiane.com

facebook.com/loianegroner

twitter.com/loiane

https://github.com/loiane


youtube.com/loianegroner