



# Tipos Abstratos em Java

## Programação O. O. II

Prof. Ronaldo Serpa da Rosa

[ronaldo.rosa@bento.ifrs.edu.br](mailto:ronaldo.rosa@bento.ifrs.edu.br)



**INSTITUTO FEDERAL**  
Rio Grande do Sul  
Campus Bento Gonçalves

# ArrayList

```
import java.util.ArrayList;
```

```
ArrayList<String> arrayList = new ArrayList<String>();  
arrayList.add("A");//Adicionar elemento  
arrayList.add("B");//Adicionar elemento  
arrayList.add(1,"C");//Adicionar em um posição específica  
boolean existe = arrayList.contains("A");//busca um valor  
int pos = arrayList.indexOf("B");//Retornar a posição do elemento  
String elemento = arrayList.get(2);  
arrayList.remove(2);//Remove elemento por posição  
arrayList.remove("B");//Remove elemento pelo objeto  
int tam = arrayList.size();//Retorna o tamanho do array
```

# Stack

```
import java.util.Stack;
```

```
Stack<Integer> pilha = new Stack<Integer>();
```

```
pilha.push(1);//Adiciona elementos a pilha
```

```
pilha.push(2);//Adiciona elementos a pilha
```

```
pilha.push(3);//Adiciona elementos a pilha
```

```
boolean vazia = pilha.isEmpty();//Verifica se a pilha esta vazia
```

```
int tam = pilha.size();//Retorna o tamanho da pilha
```

```
Integer elemtopo = pilha.peek();//Mostra o elemento do topo da pilha
```

```
pilha.pop();//Remove um elemento da pilha (desempilha)
```

```
}
```

# Queue

```
import java.util.LinkedList;
```

```
import java.util.Queue;
```

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

```
fila.add(1);//adiciona um elemento (enfileirar)
```

```
fila.add(2);//adiciona um elemento (enfileirar)
```

```
Integer elem1 = fila.peek();//mostra o primeiro elemento da fila
```

```
fila.remove();//remove um elemento (desenfileirar)
```

# Queue

```
import java.util.LinkedList;
```

```
import java.util.Queue;
```

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

```
fila.add(1);//adiciona um elemento (enfileirar)
```

```
fila.add(2);//adiciona um elemento (enfileirar)
```

```
Integer elem1 = fila.peek();//mostra o primeiro elemento da fila
```

```
fila.remove();//remove um elemento (desenfileirar)
```



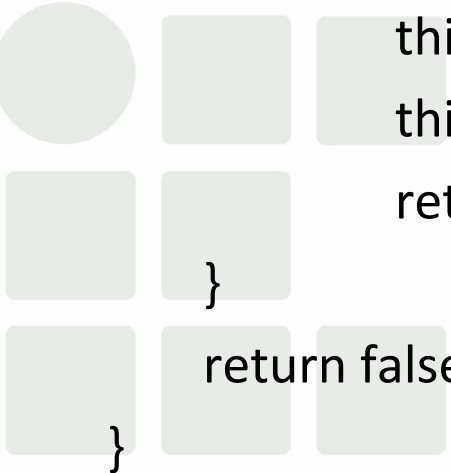
# Generics <Type>

```
public class Vetor<T> {  
    private T [] elementos;  
    private int tamanho;  
    public int getTamanho() {  
        return tamanho;}  
    public Vetor(int capacidade) {  
        this.elementos = (T[]) new Object[capacidade];  
        this.tamanho = 0;}  
}
```



# Generics <Type>

```
public boolean adicionar(T elemento) {  
    if(this.tamanho<this.elementos.length) {  
        this.elementos[this.tamanho] = elemento;  
        this.tamanho++;  
        return true;  
    }  
    return false;  
}
```



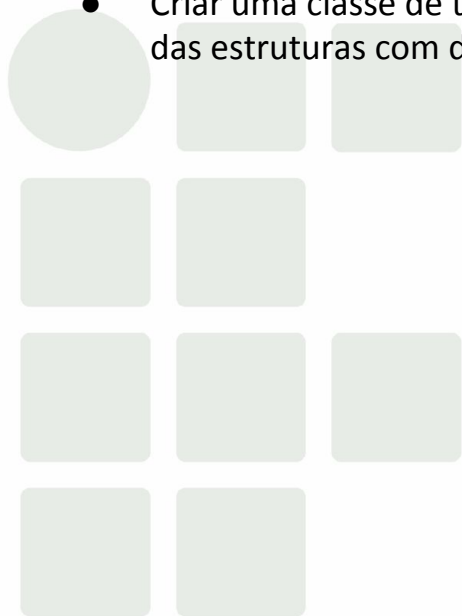
# Generics <Type>

```
public T busca(int posicao) {  
    if(!(posicao >=0 && posicao < tamanho)) {  
        throw new IllegalArgumentException("Posicao invalida");  
    }  
    return elementos[posicao];  
}
```

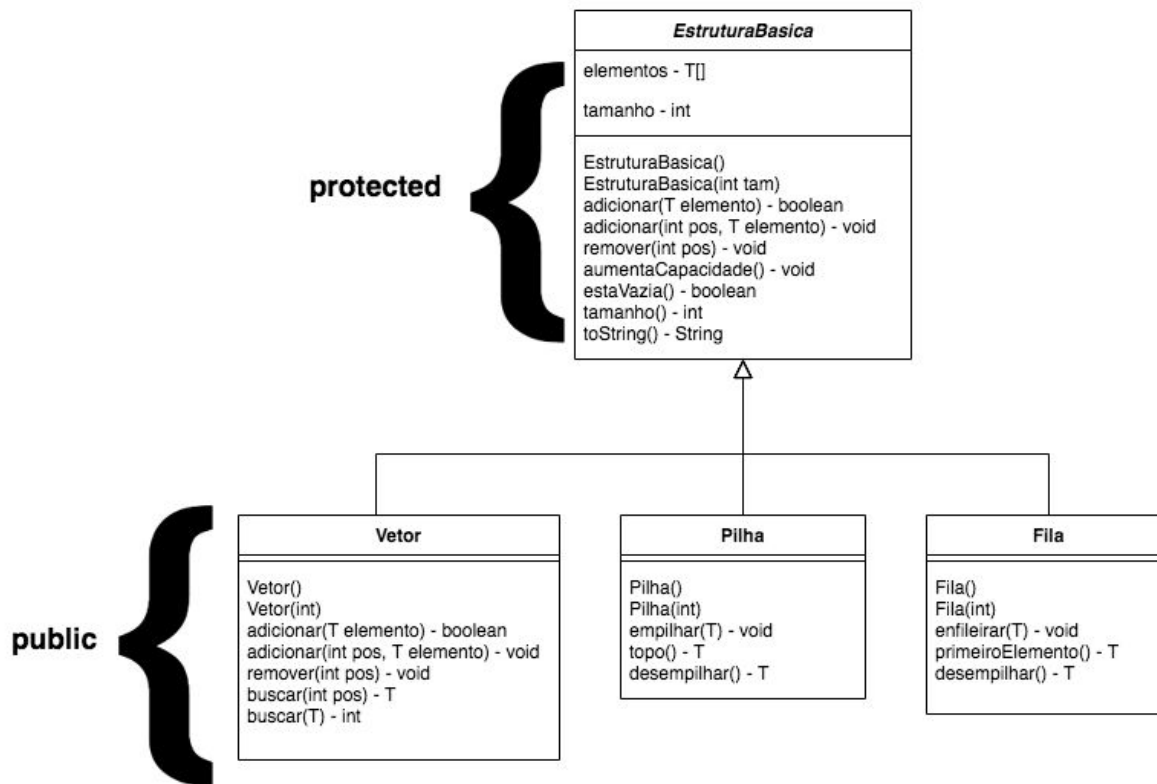


# Atividade 4 - Aplicar Generics

- Aplicar o generics nas classes criadas nos exercícios anteriores (Vetor, Pilha, Fila), possibilitando que possam ser criadas estruturas de qualquer tipo de dado.
- Criar uma classe de teste e realizar as operações (criar, adicionar, mostrar, remover) com cada uma das estruturas com diferentes tipos de dados;

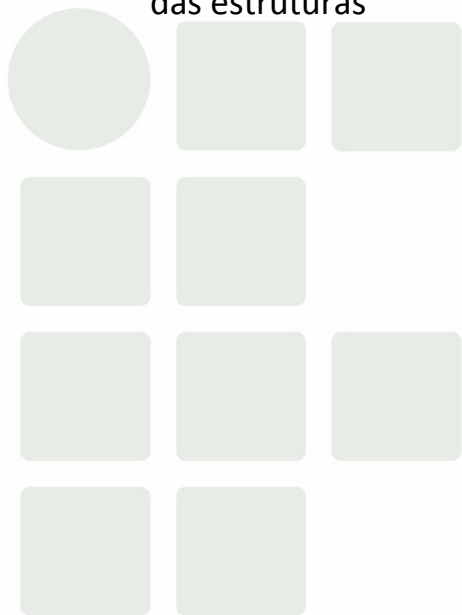


# Avaliação 1 - Refatorando as estruturas



# Avaliação 1 - Refatorando as estruturas

- Aplicar o refactoring nas classes, de acordo com o diagrama do slide anterior anterior;
- Criar uma classe de teste e realizar as operações (criar, adicionar, mostrar, remover) com cada uma das estruturas



# Referências

- GOODRICH, Michael T.; TAMASSIA, Roberto. Estruturas de dados e algoritmos em Java. 5. ed. Porto Alegre: Bookman, 2013
- <https://www.youtube.com/playlist?list=PLGxZ4Rq3BOBrgumpzz-l8kFMw2DLERdxi>