EXERCÍCIO HERANÇA

Árvore Binária

01

Classe base

Implemente uma classe que representa uma árvore de busca binária, sem balanceamento.

02

Implementação

Essa classe deve ser implementada usando um array de Strings, no formato de um heap.

03

Heap

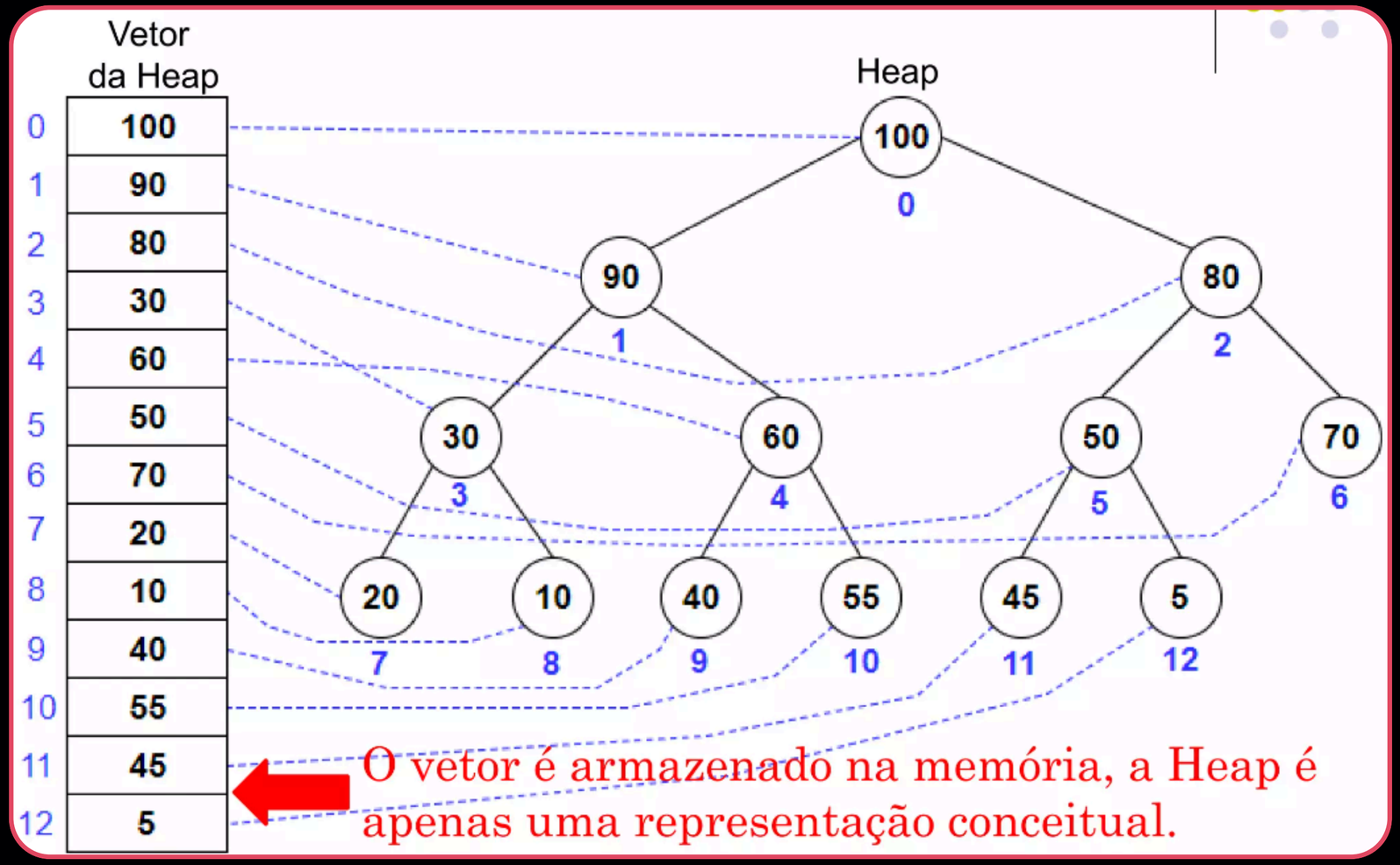
Posição 0 do array é a raiz. Filho esquerdo do nó i está na posição (2 * i + 1). Filho direito na posição (2* i + 2).

04

Interface pública

Inserir, remover, procurar, tamanho, toString.

Heap


Vetor
da Heap
Heap
0 100
1 90
2 80
3 30
4 60
5 50
6 70
7 20
8 10
9 40
10 55
11 45
12 5
100
0
90
1
80
2
30
3
60
4
50
5
70
6
20
7
10
8
40
9
55
10
45
11
5
12
O vetor é armazenado na memória, a Heap é apenas uma representação conceitual.

Obrigatório

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
boolean	**find**(String v)	Verifica se o elemento está presente na árvore.
void	**insert**(String value)	Insere um string na árvore
int	**len**()	Reorna o número de elementos presentes na árvore
boolean	**remove**(String v)	Remove um elemento da árvore.
String	**toString**()	Retorna um string no formato de um grafo da ferramenta graphviz.

Obrigatório

Constructor Summary

Constructors

Constructor	Description
`ArvBin(int len)`	Construtor gera um aárvore vazia.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
boolean	`find(String v)`	Verifica se o elemento está presente na árvore.
void	`insert(String value)`	Insere um string na árvore
int	`len()`	Reorna o número de elementos presentes na árvore
boolean	`remove(String v)`	Remove um elemento da árvore.
`String`	`toString()`	Retorna um string no formato de um grafo da ferramenta graphviz.

Árvore Binária Balanceada

01

Subclasses

Implemente uma subclasse que representa uma árvore de busca binária, com balanceamento perfeito.

02

Subclasses

Implemente uma classe que representa uma árvore AVL.

03

Heap

Serão necessários métodos protegidos na superclasse para manipular o heap.

04

Interface pública

Inserir, remover, procurar, tamanho, toString.

Subclasses

Class ArvAVL

java.lang.Object
    ArvBin
        ArvAVL

```
public class ArvAVL
extends ArvBin
```

Class ArvBal

java.lang.Object
    ArvBin
        ArvBal

```
public class ArvBal
extends ArvBin
```

Protegidos - superclasse

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
protected int	countNodes(int i)	Conta o número de nós em um subárvore.
boolean	find(String v)	Verifica se o elemento está presente na árvore.
protected String	getNode(int i)	Retorna o valor de um dado nó.
void	insert(String value)	Insere um string na árvore
protected boolean	isBalance()	Verifica se a árvore está balanceada, de acordo com o critério de cada tipo de árvore.
int	len()	Reorna o número de elementos presentes na árvore
protected int	nodeLeft(int i)	Indica qual é o número do filho à esquerda da posição i.
protected int	nodeRight(int i)	Indica qual é o número do filho à direita da posição i.
boolean	remove(String v)	Remove um elemento da árvore.
protected void	setNode(int i, String v)	Atribui um valor para um determinado nó.
String	toString()	Retorna um string no formato de um grafo da ferramenta graphviz.

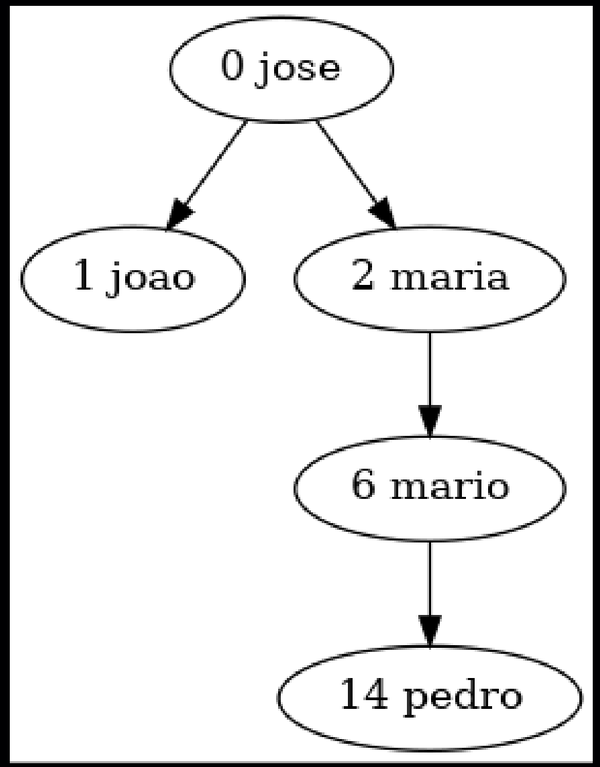
toString()

```
digraph {
"0 jose" ->"1 joao"
"0 jose" ->"2 maria"
"2 maria" ->"6 mario"
"6 mario" ->"14 pedro"
}
```

toString()

```
digraph {
"0 jose" ->"1 joao"
"0 jose" ->"2 maria"
"2 maria" ->"6 mario"
"6 mario" ->"14 pedro"
}
```

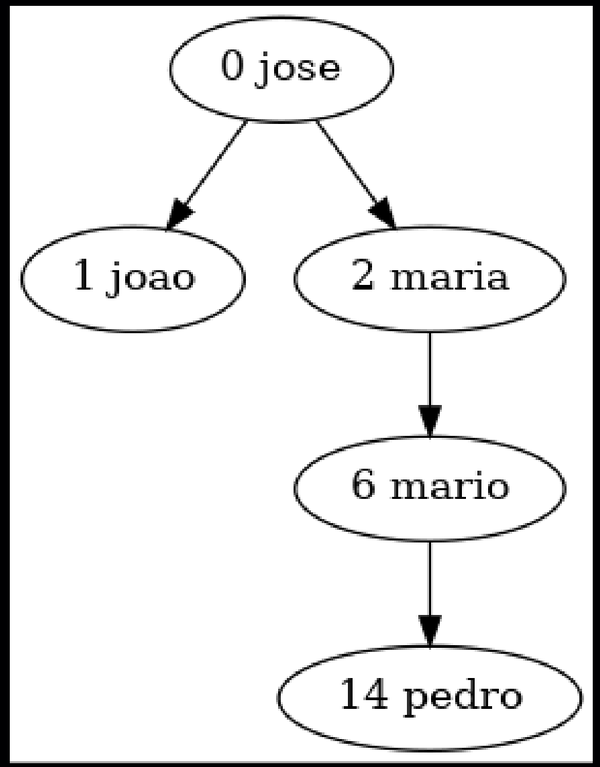
graphviz



toString()

```
digraph {
"0 jose" ->"1 joao"
"0 jose" ->"2 maria"
"2 maria" ->"6 mario"
"6 mario" ->"14 pedro"
}
```

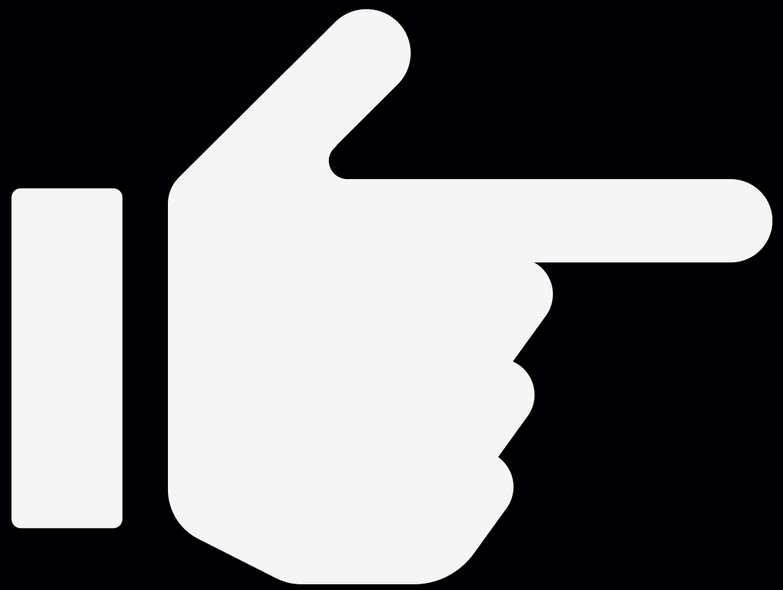
graphviz



>> dot -Tpng -o arq.png arq.dot

Entrada/Saída

```
i jose
i joao
i maria
i jose
d joao
i mario
```



toString() para cada uma das árvores

Dicas

- Cada vez que um elemento é inserido ou removido, é preciso verificar o balanceamento da árvore
- Se ela estiver desbalanceado, aplica algoritmo de rebalanceamento
- APB: número de elementos das sub-árvores difere no máximo em 1 unidade
- AVL: altura das sub-árvores difere no máximo em 1 unidade

Dicas

- APB: “jogar fora” a árvore atual e re-inserir os elementos novamente, de forma que ela fique perfeitamente balanceada
- AVL: fazer as rotações neessárias