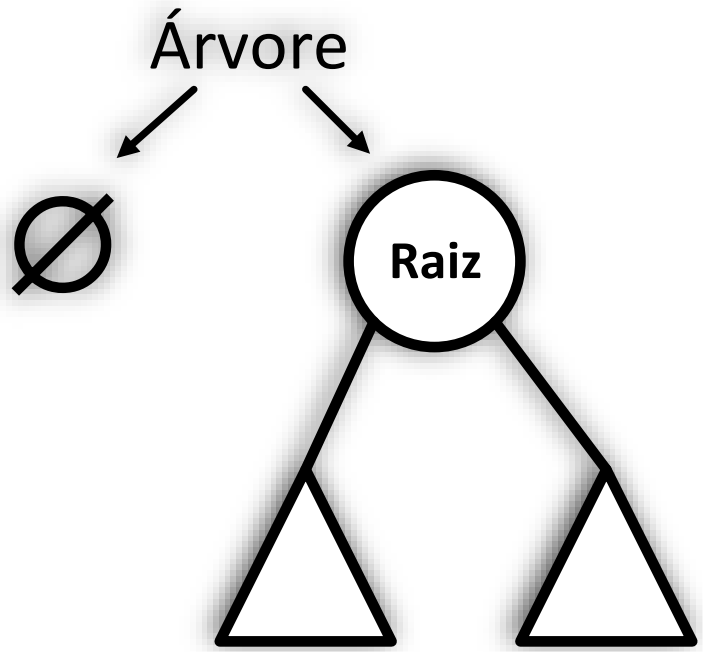
Árvores binárias

FURB
UNIVERSIDADE DE BLUMENAU

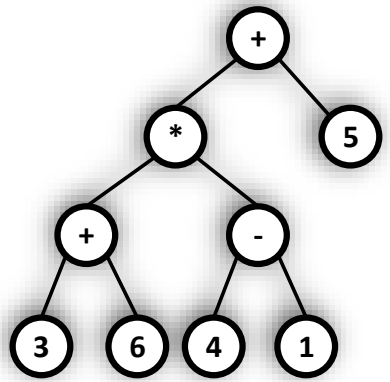
Árvores binárias

- São árvores em que qualquer nó tem no máximo 2 filhos
- Recursivamente, uma árvore binária é:
  - Uma árvore vazia ou
  - Um nó raiz com duas sub-arvores:
    - A sub-árvore da direita (*sad*)
    - A sub-árvore da esquerda (*sae*)



Árvores binárias

- Exemplo: expressões aritméticas
  - Nós folhas representam operandos
  - Nós internos representam operadores
  - Exemplo: (3+6) * (4-1) + 5

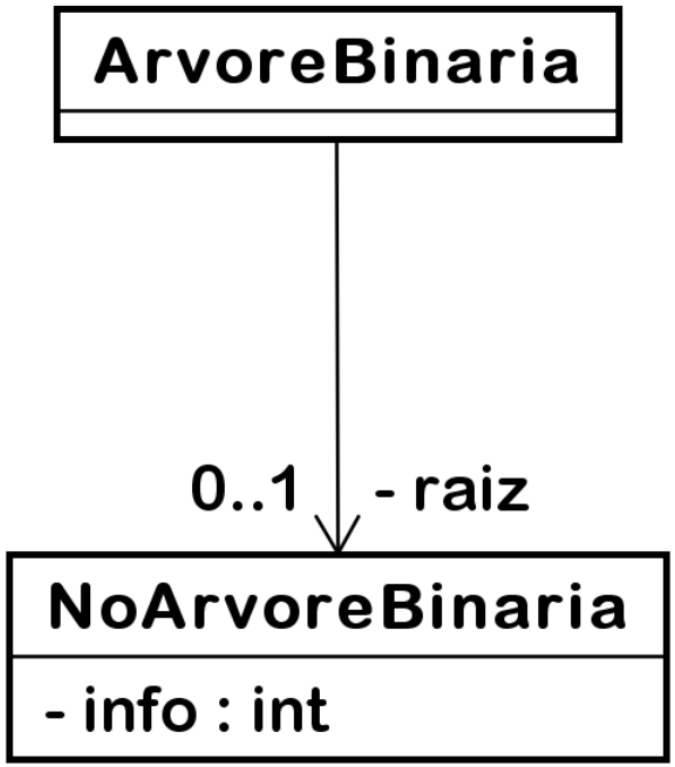


- Árvores binárias de busca são especialmente uteis para realizar buscas com O(log N).

Implementação

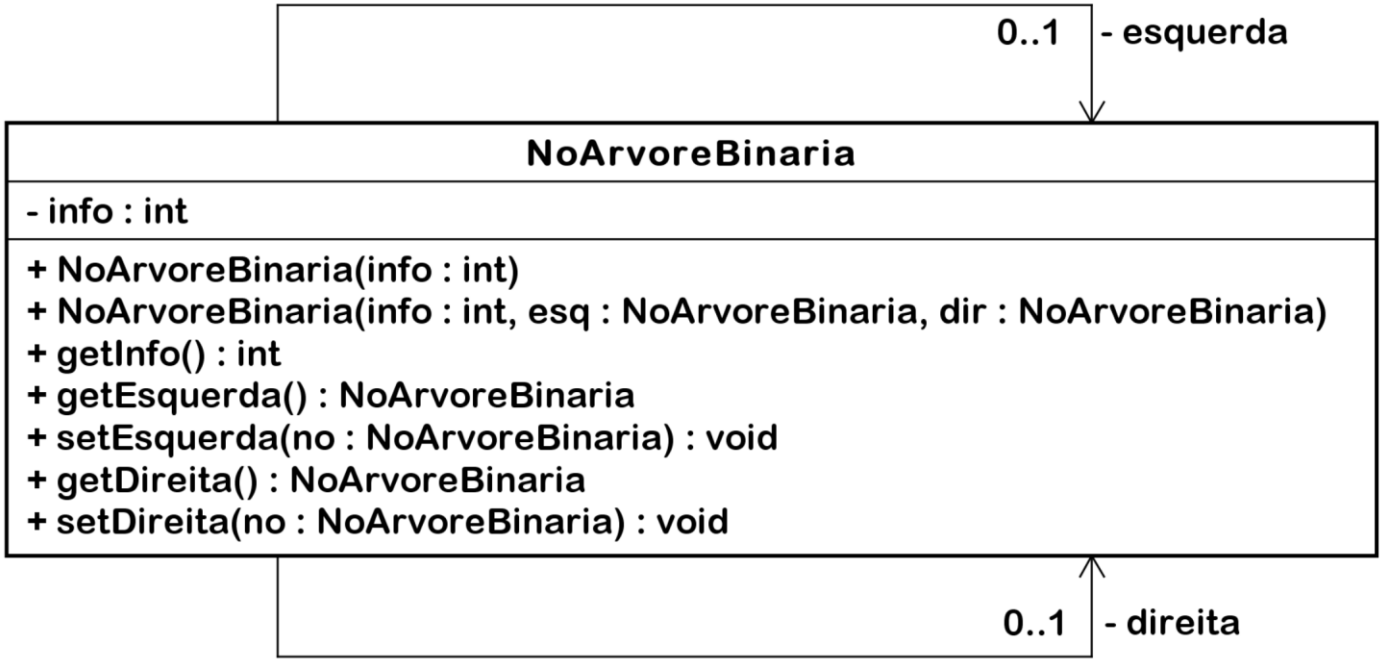
- Os nós são armazenados em objetos da classe **`NoArvoreBinaria`**

- Um objeto da classe **`ArvoreBinaria`** referencia o nó raiz da árvore.

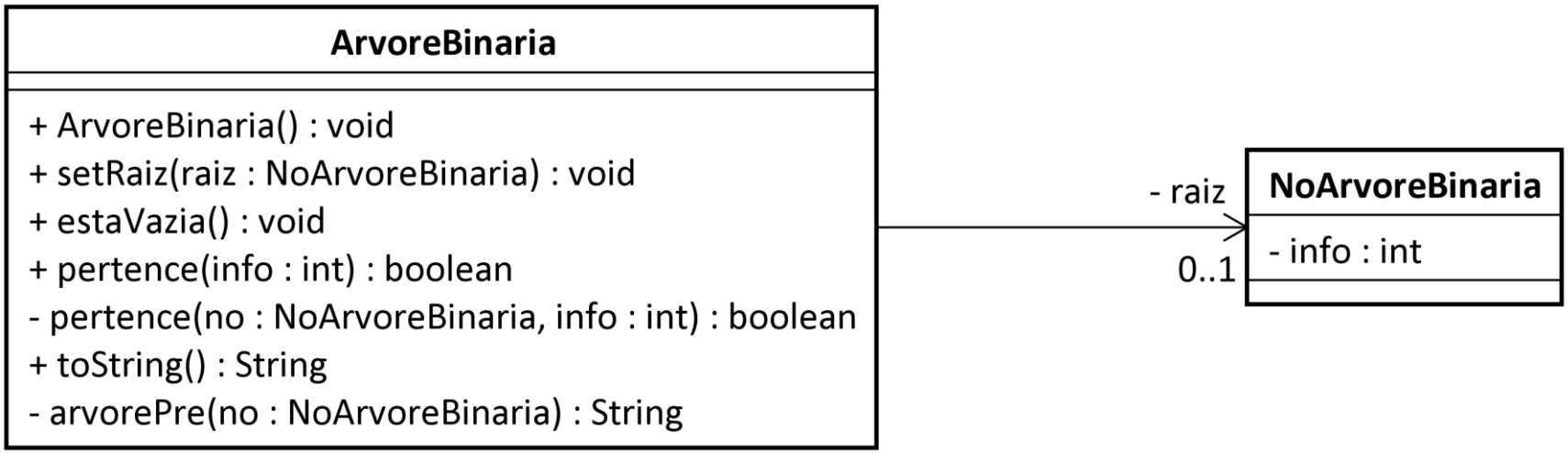


Implementação

- Cada nó é representado pela classe:



Classe Árvore Binária

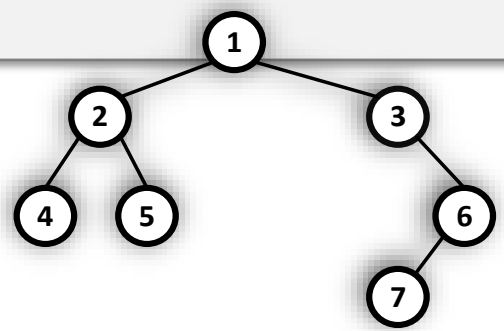


Inclusão de nós na árvore

- Criar objetos **`NoArvoreBinaria`**, estabelecendo a hierarquia entre eles
- Definir a raiz da árvore utilizado o método **`setRaiz()`**

Exemplo:

```
 1 NoArvoreBinaria<Integer> no4 = new NoArvoreBinaria<>(4);
 2 NoArvoreBinaria<Integer> no5 = new NoArvoreBinaria<>(5);
 3 NoArvoreBinaria<Integer> no2 = new NoArvoreBinaria<>(2, no4, no5);
 4
 5 NoArvoreBinaria<Integer> no7 = new NoArvoreBinaria<>(7);
 6 NoArvoreBinaria<Integer> no6 = new NoArvoreBinaria<>(6, no7, null);
 7
 8 NoArvoreBinaria<Integer> no3 = new NoArvoreBinaria<>(3, null, no6);
 9
10 NoArvoreBinaria<Integer> no1 = new NoArvoreBinaria<>(1, no2, no3);
11
12 ArvoreBinaria<Integer> arvore = new ArvoreBinaria<>();
13 arvore.setRaiz(no1);
```



Construtor da classe

- Cria uma árvore sem raiz

Algoritmo: **Criar ArvoreBinaria()**

raiz ← null;

Criar um nó

- Criar um nó sem filhos

```
Algoritmo: Criar NoArvoreBinaria(int info)

this.info ← info;
esquerda ← null;
direita ← null;
```

- Criar nó com filhos

```
Algoritmo: Criar NoArvoreBinaria(int info, NoArvoreBinaria esq, NoArvoreBinaria dir)

this.info ← info;
esquerda ← esq;
direita ← dir;
```

Método estaVazia()

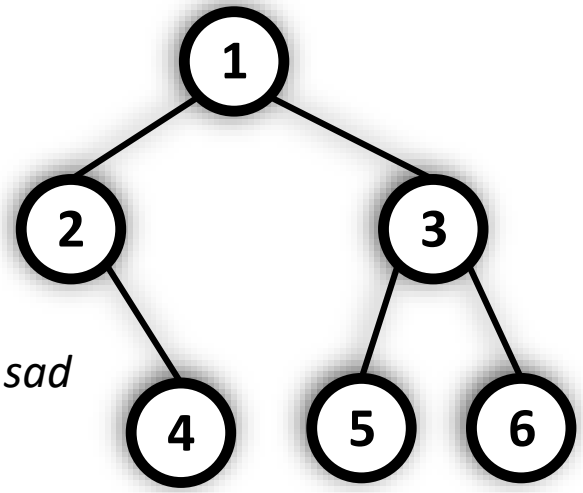
- O método **`estaVazia()`** retorna **`true`** se a árvore estiver vazia

Algoritmo**: estaVazia()**

Se (raiz = null) então
  retornar verdadeiro
senão
  retornar falso;

Ordem de percurso

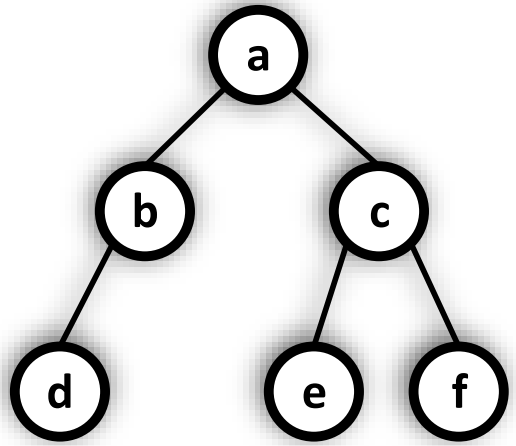
- Sendo:
  - *sae* = sub-árvore à esquerda e
  - *sad* = sub-árvore à direita
- Pré-ordem:
  - Trata raiz, percorre *sae*, percorre *sad*
  - Exemplo: 1 2 4 3 5 6
- Ordem simétrica
  - Percorre *sae*, trata raiz, percorre *sad*
  - Exemplo: 2 4 1 5 3 6
- Pós-ordem
  - Percorre *sae*, percorre *sad*, trata raiz
  - Exemplo: 4 2 5 6 3 1



Notação textual

- Árvore vazia é representada por: <>
- Árvores não vazias por: <raiz sae sad>

- Exemplo:



<a <b <d<><>> <> ><c <e<><>><f<><>>>>

Método pertence()

- Este método deve verificar a existência de um nó que contenha o valor fornecido como argumento

- Utiliza um método privado auxiliar recursivo para atingir este objetivo

Método pertence()

Algoritmo**: pertence(int info)**

retornar pertence(raiz, info);

Algoritmo**: pertence(NoArvoreBinaria no, int info)**

```
se (no = null) então
   retornar falso;
senão
   retornar (no.info = info)
            ou pertence(no.esq, info)
            ou pertence(no.dir, info);
```

Método privado