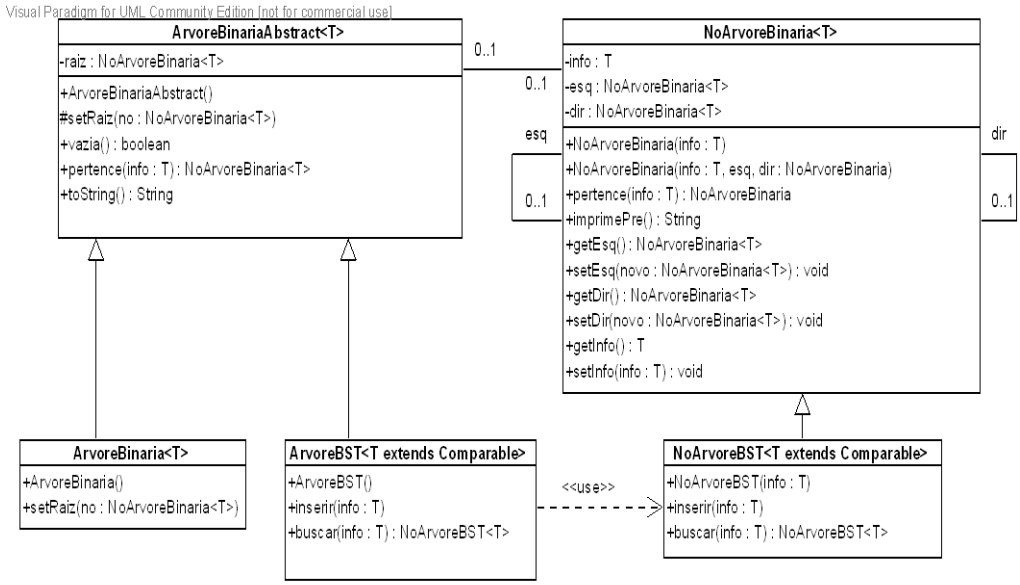
O objetivo desta atividade prática é realizar a implementação de **árvores binárias de busca**, de acordo com o diagrama de classes da figura seguinte.



Para construir a solução, recomenda-se aproveitar a implementação da Lista5, como descrito nestas etapas:

a) Fazer um backup das classes **`ArvoreBinaria`** e **`NoArvoreBinaria`** (*nunca se sabe...*);
b) Renomear a classe **`ArvoreBinaria`** para **`ArvoreBinariaAbstract`**;
c) Tornar a nova classe **`ArvoreBinariaAbstract`** uma classe abstrata;
d) Criar uma nova classe **`ArvoreBinaria`** estendendo-a da classe **`ArvoreBinariaAbstract`**;
e) Tornar o método **`setRaiz()`**, da classe **`ArvoreBinariaAbstract`**, protegido;
f) Inserir o método **`public setRaiz()`** na classe **`ArvoreBinaria`** e implementar seu código para utilizar a implementação do método **`setRaiz()`** da super classe.
g) A classe **`NoArvoreBinaria`** deste exercício difere do projeto da Lista5, pois deve-se acrescentar *setters e getters* para todos os atributos.

Os métodos novos para serem implementados na classe **`ArvoreBST`** são:

a) **`inserir()`**: este método deve inserir o dado, fornecido como argumento, na árvore binária de busca.
b) **`buscar()`**: este método deve buscar o dado fornecido como argumento, na árvore binária, retornando o nó que o armazena.

Os métodos novos para serem implementados na classe **`NoArvoreBST`** são:

c) **`inserir()`**: este método apoia o método inserir() de **`ArvoreBST`**. De maneira recursiva, identifica a posição correta e insere o novo valor.
d) **`buscar()`**: este método apoia o método buscar() de **`ArvoreBST`**. De maneira recursiva e de forma ordenada, localiza o nó que possui o valor informado. Retorna *null* caso não encontre.

Após implementar a árvore binária de busca, crie uma classe contendo o método **`main()`** para testar e demonstrar o funcionamento da nova árvore implementada (**`ArvoreBST`**) e o funcionamento da árvore que já existia (**`ArvoreBinaria`**).