|  |  |
| --- | --- |
| Đại học khoa học tự nhiên | |
| BÁO CÁO ĐỒ ÁN | |
| BINARY SEARCH TREE FRAMEWORK | |
| **MÔN HỌC: MẪU THIẾT KẾ HƯỚNG ĐỐI TƯỢNG VÀ ỨNG DỤNG** | |
| 1212148 NGUYỄN ĐẠT NHẬT HUY  1212230 NGUYỄN HẢI MIÊN  1212443 PHẠM MINH TRÍ  1212090 VÕ VĂN THÀNH ĐÔ |
| **6/8/2016** | |

|  |
| --- |
|  |

Contents

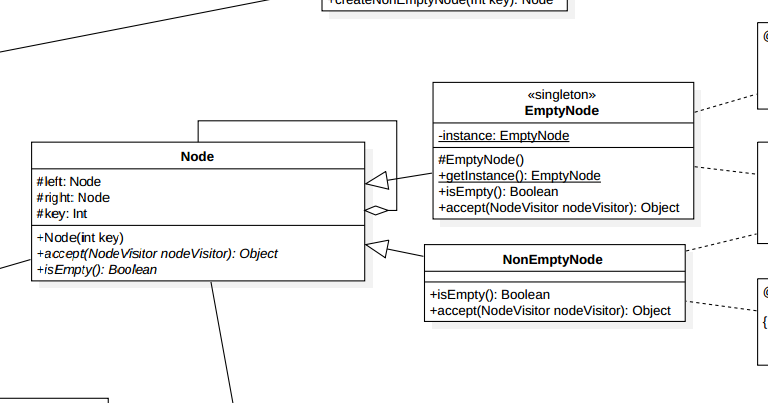
[A. SƠ ĐỒ LỚP 2](#_Toc453156834)

# SƠ ĐỒ LỚPC:\Users\PCPRO\Desktop\BSTDesignPng.png

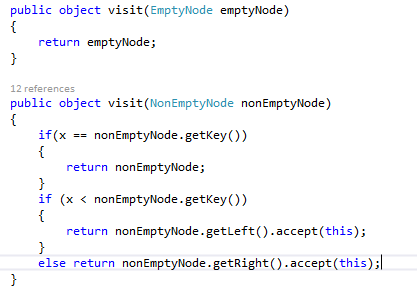
* **Lớp BSTree**: là đối tượng cây nhị phân tìm kiếm có 1 thuộc tính là 1 đối tượng Node là root(nút gốc), và các hàm xử lí dựa vào root.
* **Lớp Node**: là đối tượng nút trong cây nhị phân, là lớp abstract định nghĩa một số phương thức cho lớp con kế thừa.
* **Lớp EmptyNode**: kế thừa lớp Node, là nút rỗng. Khi 1 nút có một con thì con còn lại là nút rỗng.
* **Lớp NonEmptyNode**: kế thứa lớp Node, chứa 2 đối tượng Node khác.
* **Lớp NodeFactory**: interface sử dụng cho mẫu AbstractFactory, dùng để khởi tạo Node (EmptyNode và NonEmptyNode)
* **Lớp NodeFactoryImpl**: lớp thể hiện của interface NodeFactory, thể hiện một factory cụ thể sản xuất ra một loại Node cụ thể.
* **Lớp BSTTraversal**: lớp interface định nghĩa phương thức duyệt cây cho các lớp con kế thừa.
* **Lớp TraversalAction:** interface định nghĩa method run(), method này sẽ chạy khi visit từng node trong lúc duyệt cây.
* **Lớp PostOrderTraversal, InOrderTraversal, PreOrderTraversal**: các lớp con của lớp BSTTraversal viết lại phương thức duyệt cây đại diện cho 3 cách duyệt khác nhau.
* **Lớp BSTBuilder**: lớp khởi tạo cây nhị phân tìm kiếm thông qua mảng được truyền vào.
* **Lớp NodeVisitor**: interface định nghĩa lại phương thức duyệt qua từng nút(visit).
* **Lớp NodeDeletionVisitor**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor, để duyệt qua từng nút tìm nút cần xóa và xóa.
* **Lớp NodeMinimumSearchingVisitor**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor và trả về Node có giá trị nhỏ nhất.
* **Lớp NodeMaximumSearchingVisitor**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor và trả về Node có giá trị lớn nhất.
* **Lớp NodeSearchingVisitor**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor, trả về Node có giá trị bằng với giá trị thuộc tính x của lớp ta sẽ truyền vào.
* **Lớp NodeInsertVisitor**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor và trả về Node mới thêm vào.
* **Lớp NodeCoutingAtLevel**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor để đếm số Node tại mỗi mức của cây nhị phân.
* **Lớp PathLenghtFindingVisitor**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor và trả về độ dài đường đi từ Node root đến Node có giá trị x.
* **Lớp NodeHightFindingVisitor**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor và trả về độ cao của cây.
* **Lớp LeafCountingVisitor**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor và trả về số lượng Node lá.
* **Lớp NodeCoutingVisitor**: viết lại phương thức duyệt(visit) kế thừa của lớp NodeVisitor và trả về số lượng Node theo yêu cầu(đếm số Node có 2 con, 1 con….)
* **Lớp NodeCountingCondition**: lớp kiểm tra điều kiện được truyền vào(node 1 con, node 2 con, node có con trái….) trả về giá trị true hoặc false.
* **Lớp NodeState**: lớp enum định nghĩa các trạng thái của node (có 1 con, 2 con, có con trái, con phải…).
* **Lớp NodeChecker**: lớp sử dụng các phương thức static public để kiểm tra các trạng thái của Node.

# CÁC MẪU THIẾT KẾ ĐƯỢC ÁP DỤNG

## Composite



**NodeSearchingVisitor.cs**



Ở đây lớp component của mẫu composite là class Node. Ở đây lớp composite là lớp NonEmptyNode kế thừa lớp Node và có thể chứa 2 đối tượng Node khác, lớp EmptyNode thể hiện node rỗng, còn đối với nút lá của cây nhị phân thì vẫn là NonEmptyNode và chứa 2 đối tượng EmptyNode.