UML Practice on Algorithms (2)

Today's practice is about using UML diagrams to implement two algorithms involved in a "Tree". To finish "Practice 3" and "Practice 4", you need the file "BinaryTree.asta", please download the file in Manaba +R.

Please submit your answer file "UML_Practice3.png" to "Final Practice 3" and your answer file "UML_Practice4.png" to "Final Practice 4" under "Assignments" tab in Manaba +R.

The deadline for "Practice 3" is by **July 21st., 2022 12:10** and the deadline for "Practice 4 is by **July 22nd., 2022 12:00.** The maximum points for "Practice 3" are **4P** and for "Practice 4" is **5p**.

- **Tree:**

A Tree is a non-linear data structure where **data objects (node)** are generally organized in terms of hierarchical relationship. Non-linear structure means that **data objects (node)** in a tree is not organized linearly like Linked Lists.

In a Tree data structure, the topmost node is known as a **root node**. Each node contains "data" (the data can be any type) and "reference(s)" to other nodes. Fig. 1 shows the theory of a Binary Tree.

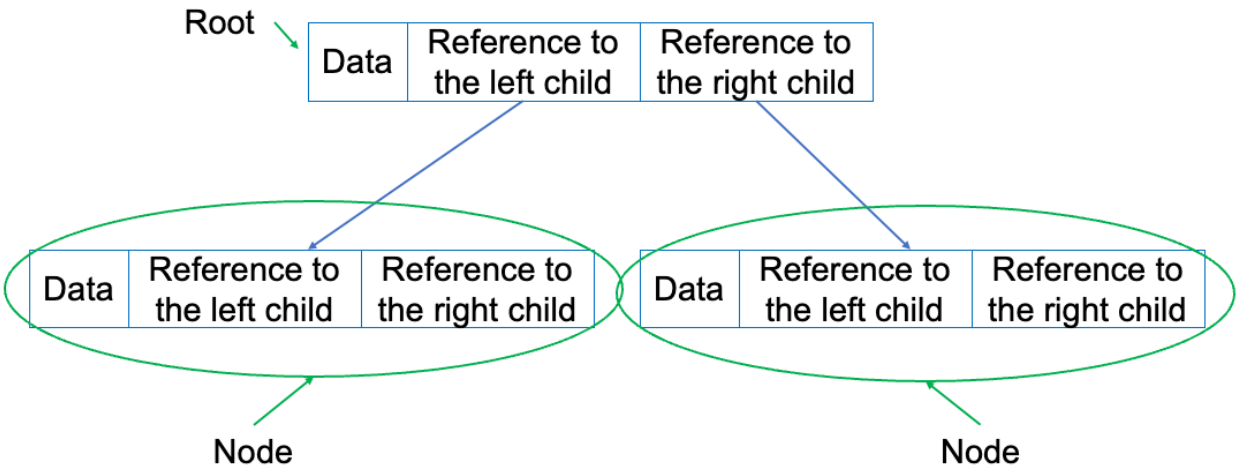

Fig. 1: The illustration of a Binary Tree

- **Binary Tree:**

A Binary Tree is a recursive Tree data structure where each node has zero or two children. In a Binary Tree, each node contains {“Data”, “Reference to the left child”, “Reference to the right child”}

- **Describe a Binary Tree using a Class Diagram:**

A Binary Tree can be represented as a class and a Node as a separate class. The Binary Tree class contains a reference to the topmost node (root). Fig. 2 shows the UML notations of a Binary Tree class and a Node class.

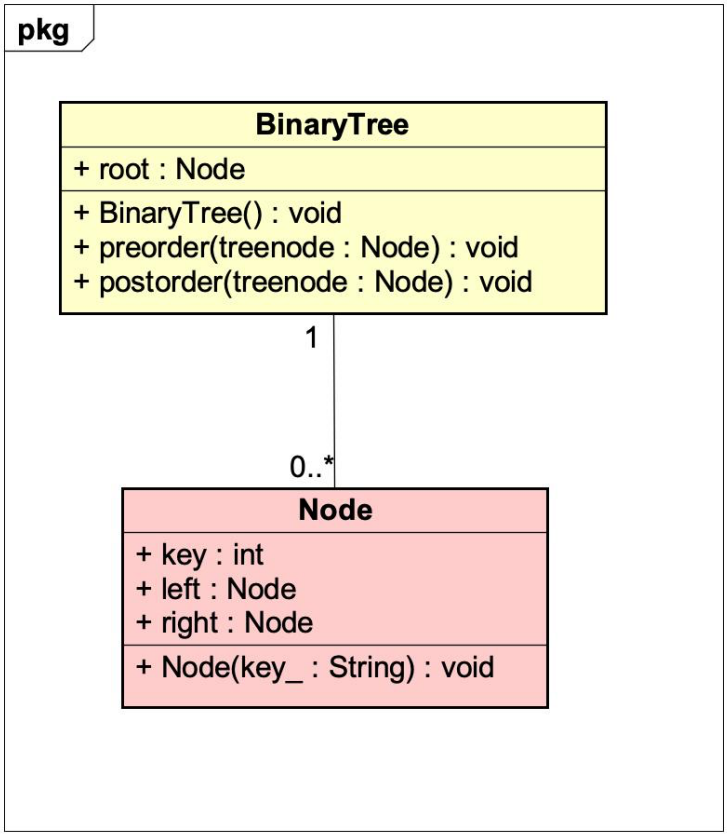

Fig. 2: A UML class diagram notation for a Binary Tree and a Node

- **Describe the process to build a Binary Tree using an Activity Diagram:**

Fig 3 shows a Binary Tree which we are aiming to build. The whole process includes the following six steps. **In addition, the balloons next the node means the name of the node. For example, for the topmost node, its name is "new_tree" and the "data" of the "new_tree" is "R". Another example, the left child of "new_tree", the node's name is "A_tree" and the "data" of the "A_tree" is "A".**

1. Allocate a BinaryTree object (the object name is "new_tree")
2. Create the Root of the "new_tree" by initializing a new Node object with "R" as its "data".
3. Create the left child of "new_tree" by initializing a new Node object with "A" as its "data".
4. Create the right child of "new_tree" by initializing a new Node object with "B" as its "data"
5. Create the left child of "A_tree" by initializing a new Node object with "C" as its "data".
6. Create the right child of "A_tree" by initializing a new Node object with "D" as its "data".

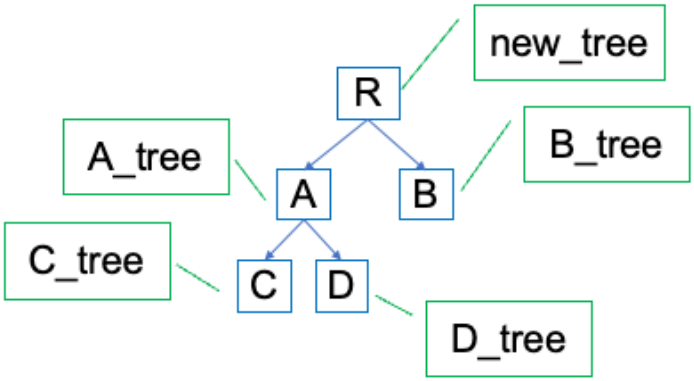

Fig. 3: An illustration of a Binary Tree

Fig. 4 shows an activity diagram for describing the above six steps of building the Binary Tree in Fig. 3.

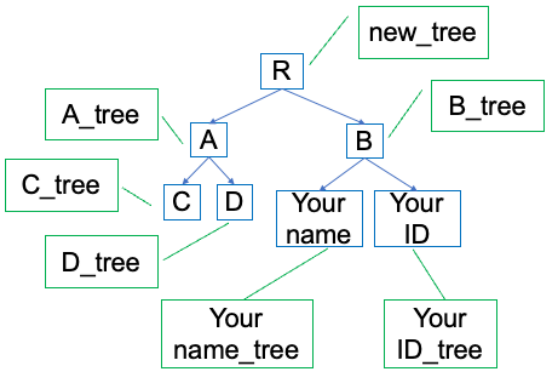

Fig. 5: An illustration of a Binary Tree for Practice 3

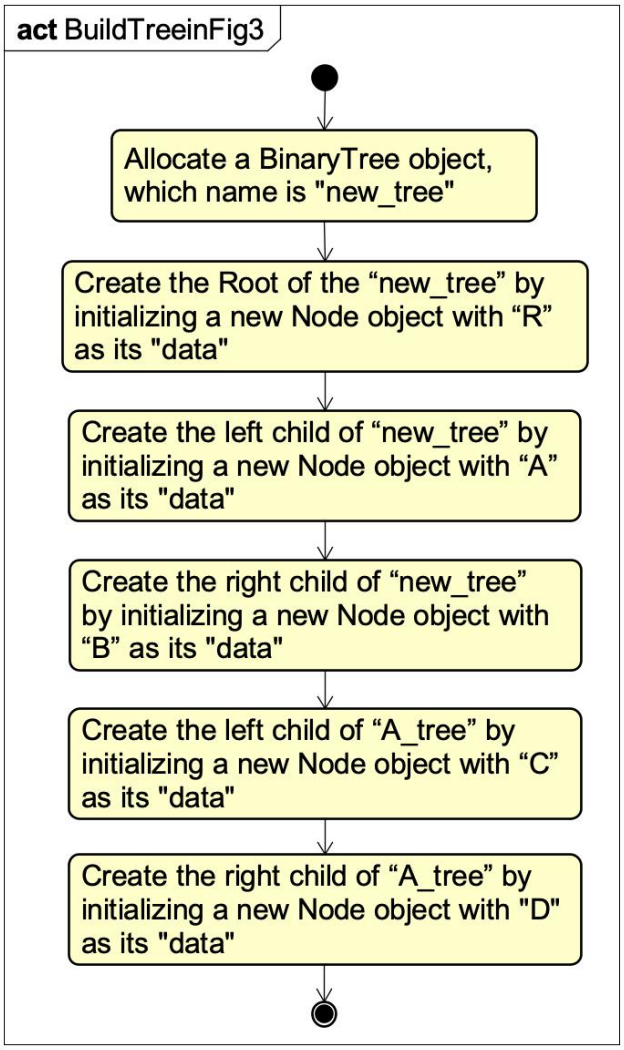

Fig. 4: Describing the process to build the Binary Tree in Fig. 3 using an Activity Diagram

- **Practice 3:** Please follow the following requirements and add one left child and one right child of the “B_tree” in the tree in Fig. 3.
  - **Requirements for Practice 3**:

1. The Binary Tree with added children of “B_tree” is shown in Fig 5.
2. Add two more steps (Step 7 and Step 8) for adding the children of the “B_tree”.
3. The “data” of the left child of the “B_tree” should be your name.
4. The “data” of the right child of the “B_tree” should be your ID.
5. Add the steps in the diagram “BuildTreeinFig3” and export the diagram as “UML_Practice3.jpg”. **Color the added step blocks with different color.**
6. The diagram “BuildTreeinFig3” is in the given “.astah” file “BinaryTree.astah”.

- **Describe the method “preorder()” using and Activity Diagram:**

The “preorder()” method in the BinaryTree class is for displaying the “data” of a visited node in a preorder traversal **in which a node is visited before its descendants**. The algorithm for the “preorder()” method is described in the following steps:

1. Give a BinaryTree object.
2. If the given tree object is NOT null, do following steps:

2-1: Display the “data” of the given tree object.

2-2: If the left child of the given tree object is NOT null, then assign the left child as the given tree object. Repeat Step #2.

2-3: If the right child of the given tree object is NOT null, then assign the right child as the given tree object. Repeat Step #2.

Fig. 5 shows an activity diagram for describing the “preorder()” method functions for the Binary Tree in Fig. 3. The given tree object at the beginning is the “new_tree”. The output after the preorder traversal for the Binary Tree in Fig. 3 is “R->A->C->D->B”.

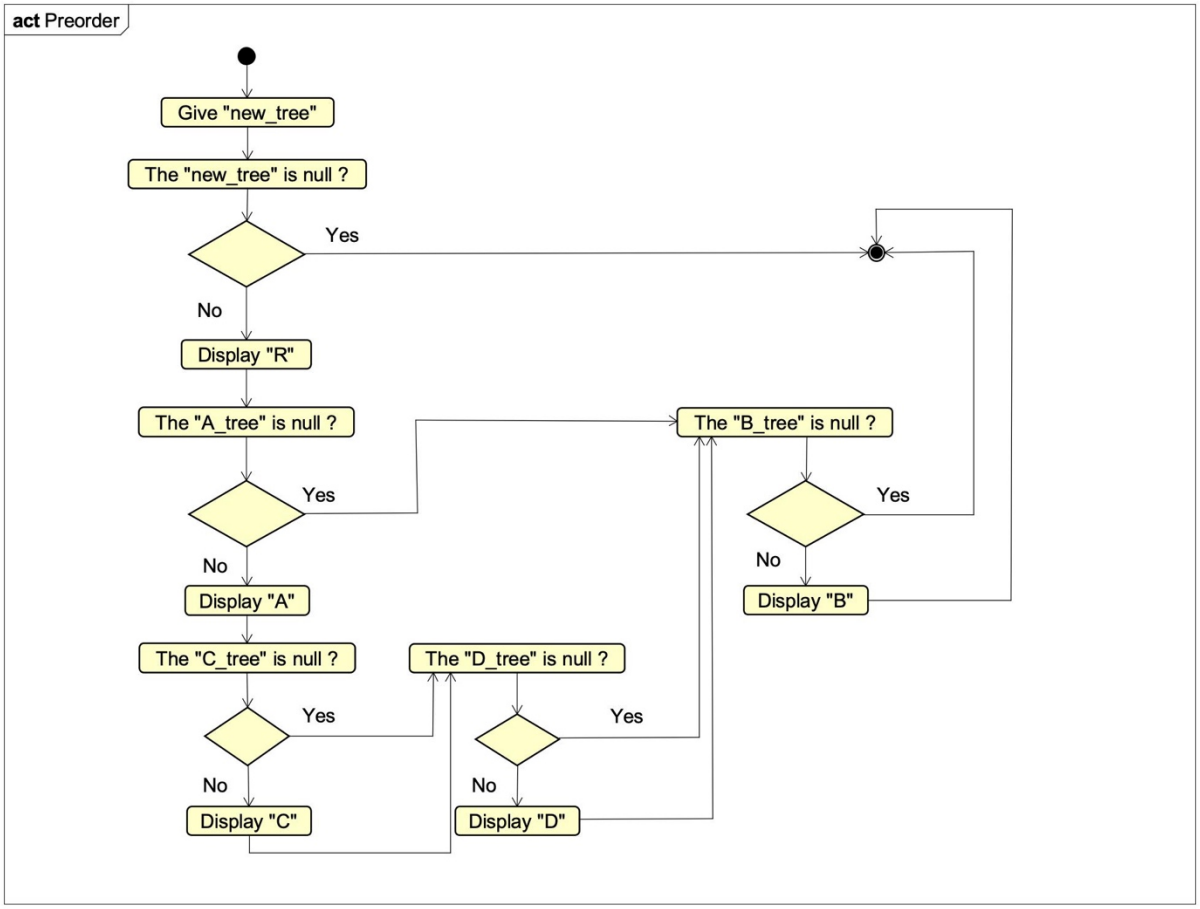

Fig. 5: Describing the “preorder()” method for the Binary Tree in Fig. 3 using an Activity Diagram

- **Practice 4:**

The “postorder()” method in the BinaryTree class is for displaying the “data” of a visited node in a postorder traversal in **which a node is visited after its descendants**. The algorithm for the “postorder()” method can be described in the following steps:

1. Give a BinaryTree object.
2. If the given tree object is NOT null, do following steps:

2-1: If the left child of the given tree object is NOT null, then assign the left child as the given tree object. Repeat Step # 2.

2-2: If the right child of the given tree object is NOT null, then assign the right child as the given BinaryTree object. Repeat Step #2.

2-3: Display the “data” of the given tree object.

Please follow the following requirements and describe the “postorder()” method in an activity diagram.

- **Requirements for Practice 4**:

1. Please use the tree that you made **in Practice 3 shown in Fig 5.**

2. The diagram “Postorder” has already finished the postorder traversal for the left child of “new_tree”.
3. Finish the diagram “Postorder” by finishing the postorder traversal for the right child of the “new_tree” and export the diagram as “UML_Practice4.jpg”. **Color the added step blocks with different color.**

4. The diagram “Postorder” is in the given “.astah” file “BinaryTree.astah**”.**

5. The output after the postorder traversal for the Binary Tree in Fig. 5 is “C->D->A->your name -> your ID -> B -> R”.