Laporan Praktikum

Struktur Data Non Linear DP

Modul 3

Dosen Pengampu

JB. Budi Darmawan S.T., M.Sc.

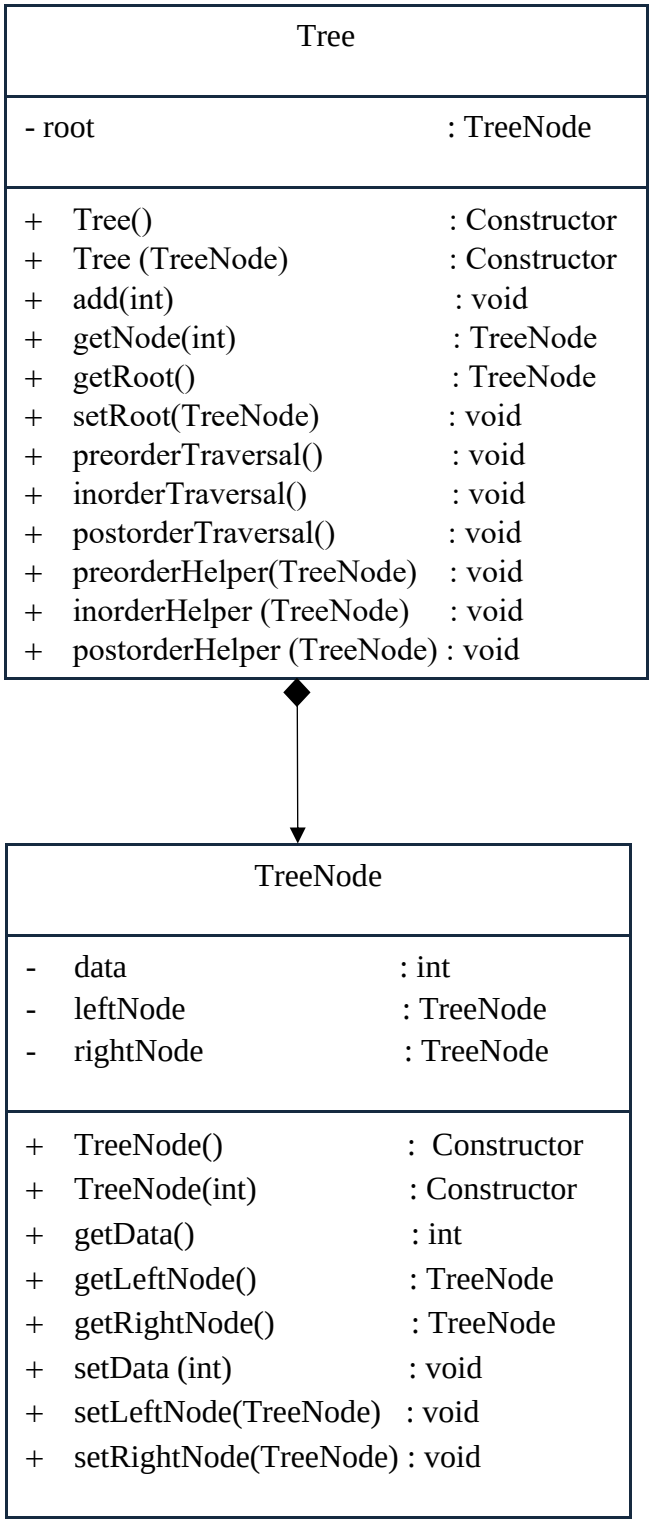
Oleh :

Nama : Maria Gresia Plena Br Purba

NIM : 235314094

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA
2024

1. Diagram UML



2. Source Code

preorderTraversal()

```
74      public void preorderTraversal() {
75          preorderHelper(localRoot: root);
76      }
77
```

inorderTraversal()

```
78      public void inorderTraversal() {
79          inorderHelper(localRoot: root);
80      }
81
```

postorderTraversal()

```
82      public void postorderTraversal() {
83          postorderHelper(localRoot: root);
84      }
85
```

preorderHelper()

```
86      public void preorderHelper(TreeNode localRoot) {
87          if (localRoot != null) {
88              System.out.print(localRoot.getData() + " ");
89              preorderHelper(localRoot: localRoot.leftNode);
90              preorderHelper(localRoot: localRoot.rightNode);
91          }
92
93      }
94
```

inorderHelper()

```
95      public void inorderHelper(TreeNode localRoot) {
96          if (localRoot != null) {
97              inorderHelper(localRoot: localRoot.leftNode);
98              System.out.print(localRoot.getData() + " ");
99              inorderHelper(localRoot: localRoot.rightNode);
100         }
101     }
```

postorderHelper()

```
103     public void postorderHelper(TreeNode localRoot) {
104         if (localRoot != null) {
105             postorderHelper(localRoot: localRoot.leftNode);
106             postorderHelper(localRoot: localRoot.rightNode);
107             System.out.print(localRoot.getData() + " ");
108
109         }
110     }
```

TreeMain

```java
package modul3b;

import java.util.Scanner;

public class TreeMain {

    public static void main(String[] args) {
        Scanner in = new Scanner(source: System.in);
        Tree data = new Tree();

        data.add(x: 42);
        data.add(x: 21);
        data.add(x: 38);
        data.add(x: 27);
        data.add(x: 71);
        data.add(x: 82);
        data.add(x: 55);
        data.add(x: 63);
        data.add(x: 6);
        data.add(x: 2);
        data.add(x: 40);
        data.add(x: 12);

        System.out.println(x: "data yang ditambahkan = 42, 21, 38, 27, 71, 82, 55, 63, 6, 2, 40 dan 12");

        System.out.print(x: "Preorder = ");
        data.preorderTraversal();

        System.out.print(x: "\nInorder = ");
        data.inorderTraversal();

        System.out.print(x: "\nPostorder = ");
        data.postorderTraversal();
    }
}
```

Output

```
run:
data yang ditambahkan = 42, 21, 38, 27, 71, 82, 55, 63, 6, 2, 40 dan 12
Preorder = 42 21 6 2 12 38 27 40 71 55 63 82
Inorder = 2 6 12 21 27 38 40 42 55 63 71 82
Postorder = 2 12 6 27 40 38 21 63 55 82 71 42 BUILD SUCCESSFUL (total time: 7 seconds)
```

3. Analisa

Tree

Code	Penjelasan

public void preorderTraversal() {
preorderHelper(root);
}
	Method ini berfungsi untuk memanggil method preorderHelper dengan isi parameter root.

public void inorderTraversal() {
inorderHelper(root);
}
	Method ini berfungsi untuk memanggil method inorderHelper dengan isi parameter root.

public void postorderTraversal() {
postorderHelper(root);
}
	Method ini berfungsi untuk memanggil method postorderHelper dengan isi parameter root.

public void preorderHelper(TreeNode localRoot) {
if (localRoot != null) {
System.out.print(localRoot.getData() + " ");
preorderHelper(localRoot.leftNode);
preorderHelper(localRoot.rightNode);
}
}
	Method ini berfungsi untuk memeriksa apakah localRoot tidak sama dengan null. Jika tidak, maka akan mencetak nilai dari node localRoot. Kemudian, method ini dipanggil secara rekursif dari localRoot untuk leftNode dan rightNode.

public void inorderHelper(TreeNode localRoot) {
if (localRoot != null) {
inorderHelper(localRoot.leftNode);
System.out.print(localRoot.getData() + " ");
	Method ini berfungsi untuk memeriksa apakah localRoot tidak sama dengan null. Jika tidak, maka method ini dipanggil secara rekursif dari localRoot untuk leftNode. Kemudian, mencetak nilai dari node localRoot. Setelah itu, method ini dipanggil secara rekursif dari localRoot untuk rightNode.

inorderHelper(localRoot.rightNode);
}
}	

public void postorderHelper(TreeNode localRoot) {
if (localRoot != null) {
postorderHelper(localRoot.leftNode);
postorderHelper(localRoot.rightNode);
System.out.print(localRoot.getData() + " ");
}
}
	Method ini berfungsi untuk memeriksa apakah localRoot tidak sama dengan null. Jika tidak, maka method ini dipanggil secara rekursif dari localRoot untuk leftNode dan rightNode. Lalu, mencetak nilai dari node localRoot.

TreeNode

Code	Penjelasan
private int data;	Mendeklarasikan atribut data dengan tipe data integer yang bersifat private.
private TreeNode leftNode;	Mendeklarasikan atribut leftNode dengan tipe data TreeNode yang bersifat private.
private TreeNode rightNode;	Mendeklarasikan atribut rightNode dengan tipe data TreeNode yang bersifat private.

public TreeNode() {
this(0);
}
	Berfungsi untuk memanggil constructor yang lain. Lalu, menginisialisasikan data dengan nilai 0.

public TreeNode(int data) {
this.data = data;
leftNode = null;
rightNode = null;
}
	Menginisialisasi data dengan nilai yang ada. Menginisialisasi leftNode, serta rightNode dengan null.

public int getData() {
return data;
}
	Method ini berfungsi untuk mengembalikan nilai dari data.

public TreeNode getLeftNode() {
return leftNode;
}
	Method in berfungsi untuk mengembalikan nilai dari leftNode.

public TreeNode getRightNode() {
return rightNode;
}
	Method in berfungsi untuk mengembalikan nilai dari rightNode.

public void setData(int data) {
this.data = data;
}
	Method ini berfungsi untuk mengubah nilai data saat ini dengan data baru yang ingin diinput.

public void setLeftNode(TreeNode lefftNode) {
this.leftNode = leftNode;
}
	Method ini berfungsi untuk mengubah nilai leftNode saat ini dengan leftNode baru yang ingin diinput.

public void setRightNode(TreeNode rightNode) {
this.rightNode = rightNode;
}
	Method ini berfungsi untuk mengubah nilai rightNode saat ini dengan rightNode baru yang ingin diinput.

4. Debug inorderTraversal()

a. Dengan objek data memanggil method inOrderTraversal()

```
System.out.print(s: "\nInorder = ");
data.inorderTraversal();
```

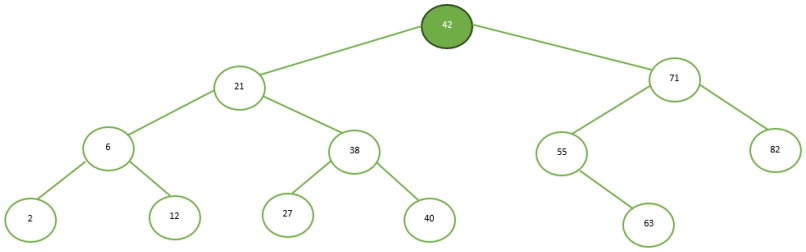
b. Method inOrderTraversal memanggil inOrderHelper() dengan parameter root agar user tidak perlu mengisi root dengan manual

```
debug:
data yang ditambahkan = 42, 21, 38, 27, 71, 82, 55, 63, 6, 2, 40 dan 12
Preorder = 42 21 6 2 12 38 27 40 71 55 63 82
Inorder =
```

```
77      public void inorderTraversal() {
            inorderHelper(localRoot: root);
79      }
```

c. Memeriksa apakah node sekarang null atau tidak

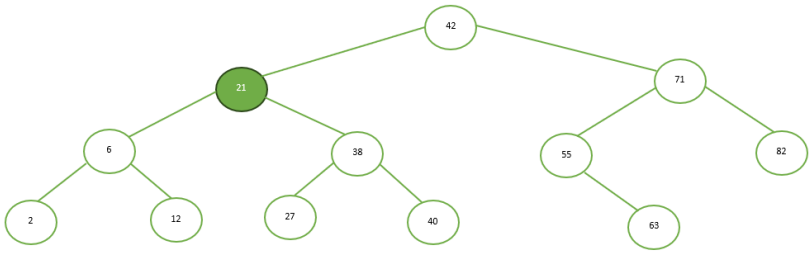
```
    public void inorderHelper(TreeNode localRoot) {
        if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
            System.out.print(localRoot.getData() + " ");
            inorderHelper(localRoot: localRoot.rightNode);
        }
    }
```



d. Melakukan operasi rekursif dengan memanggil method inorderHelper dengan parameter localRoot dari leftNode sekarang

```
    public void inorderHelper(TreeNode localRoot) {
        if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
            System.out.print(localRoot.getData() + " ");
            inorderHelper(localRoot: localRoot.rightNode);
        }
    }
```

```
    public void inorderHelper(TreeNode localRoot) {
        if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
            System.out.print(localRoot.getData() + " ");
            inorderHelper(localRoot: localRoot.rightNode);
        }
    }
```



e. Memeriksa apakah node sekarang null atau tidak. Node tidak null, sehingga melanjutkan ke step selanjutnya

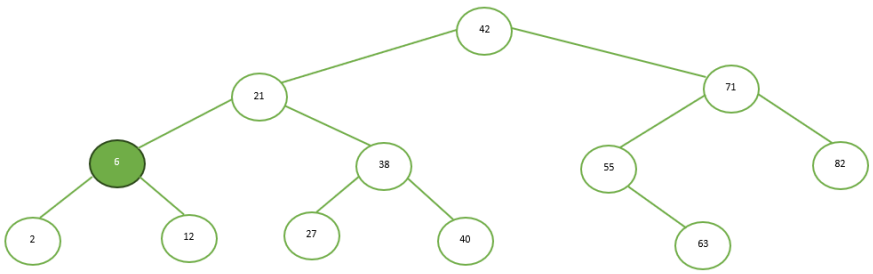
```
    public void inorderHelper(TreeNode localRoot) {
        if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
97          System.out.print(localRoot.getData() + " ");
98          inorderHelper(localRoot: localRoot.rightNode);
99      }
100 }
101
```

f. Memanggil method inorderHelper dengan parameter localRoot dari leftNode sekarang

```
    public void inorderHelper(TreeNode localRoot) {
95      if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
97          System.out.print(localRoot.getData() + " ");
98          inorderHelper(localRoot: localRoot.rightNode);
99      }
100 }
101
```

g. Melakukan operasi rekursif dengan memanggil method inorderHelper dengan parameter localRoot dari leftNode sekarang.

```
    public void inorderHelper(TreeNode localRoot) {
95      if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
97          System.out.print(localRoot.getData() + " ");
98          inorderHelper(localRoot: localRoot.rightNode);
99      }
100 }
101
```



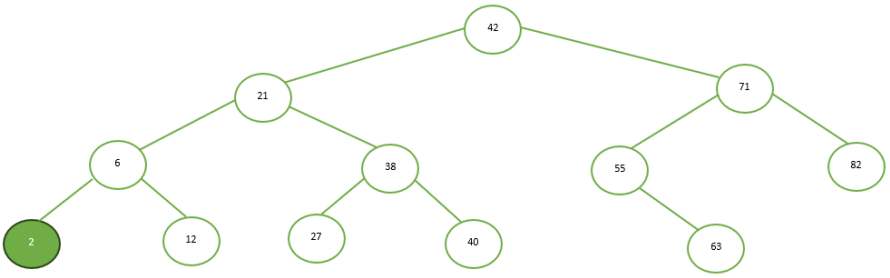
h. Memeriksa apakah node sekarang null atau tidak

```
    public void inorderHelper(TreeNode localRoot) {
        if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
            System.out.print(localRoot.getData() + " ");
            inorderHelper(localRoot: localRoot.rightNode);
        }
    }
```

i. Melakukan operasi rekursif dengan memanggil method inorderHelper dengan parameter localRoot dari leftNode sekarang

```
    public void inorderHelper(TreeNode localRoot) {
        if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
            System.out.print(localRoot.getData() + " ");
            inorderHelper(localRoot: localRoot.rightNode);
        }
    }
```

```
    public void inorderHelper(TreeNode localRoot) {
        if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
            System.out.print(localRoot.getData() + " ");
            inorderHelper(localRoot: localRoot.rightNode);
        }
    }
```



j. Memeriksa apakah node sekarang null atau tidak

```
    public void inorderHelper(TreeNode localRoot) {
        if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
            System.out.print(localRoot.getData() + " ");
            inorderHelper(localRoot: localRoot.rightNode);
        }
    }
```

k. Melakukan operasi rekursif dengan memanggil method inorderHelper dengan parameter localRoot dari leftNode sekarang

```
95      public void inorderHelper(TreeNode localRoot) {
96          if (localRoot != null) {
97              inorderHelper(localRoot: localRoot.leftNode);
98              System.out.print(localRoot.getData() + " ");
99              inorderHelper(localRoot: localRoot.rightNode);
100         }
101     }
```

```
95      public void inorderHelper(TreeNode localRoot) {
96          if (localRoot != null) {
97              inorderHelper(localRoot: localRoot.leftNode);
98              System.out.print(localRoot.getData() + " ");
99              inorderHelper(localRoot: localRoot.rightNode);
100         }
101     }
```

l. Memeriksa apakah node sekarang null atau tidak. leftNode dari 2 tidak ada (null)

```
95      public void inorderHelper(TreeNode localRoot) {
96          if (localRoot != null) {
97              inorderHelper(localRoot: localRoot.leftNode);
98              System.out.print(localRoot.getData() + " ");
99              inorderHelper(localRoot: localRoot.rightNode);
100         }
101     }
```

m. Perintah selesa. Lalu, kembali ke node 2

```
95      public void inorderHelper(TreeNode localRoot) {
96          if (localRoot != null) {
97              inorderHelper(localRoot: localRoot.leftNode);
98              System.out.print(localRoot.getData() + " ");
99              inorderHelper(localRoot: localRoot.rightNode);
100         }
101     }
```

n. Selanjutnya mencetak node yang sekarang

```
95      public void inorderHelper(TreeNode localRoot) {
96          if (localRoot != null) {
97              inorderHelper(localRoot: localRoot.leftNode);
98              System.out.print(localRoot.getData() + " ");
99              inorderHelper(localRoot: localRoot.rightNode);
100         }
101     }
```

o. Mengambil data dari localRoot(node sekarang) untuk dicetak

```
95      public void inorderHelper(TreeNode localRoot) {
96          if (localRoot != null) {
97              inorderHelper(localRoot: localRoot.leftNode);
98              System.out.print(localRoot.getData() + " ");
99              inorderHelper(localRoot: localRoot.rightNode);
100         }
101     }
```

p. Mencetak data localRoot yang telah diambil

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
                System.out.print(localRoot.getData() + " ");
98              inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
101
```

```
debug:
data yang ditambahkan = 42, 21, 38, 27, 71, 82, 55, 63, 6, 2, 40 dan 12
Preorder = 42 21 6 2 12 38 27 40 71 55 63 82
Inorder = 2
```

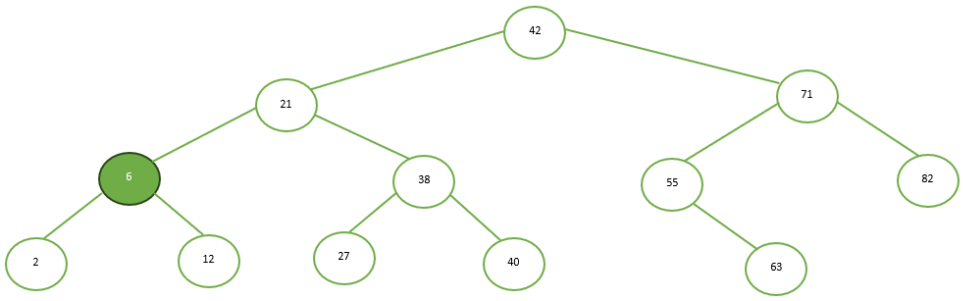
q. Memeriksa apakah node sekarang null atau tidak. leftNode dari 2 tidak ada (null)

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
97              System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
101
```

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
97              System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
101
```

r. Method untuk node 2 selesai

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
97              System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
        }
101
```



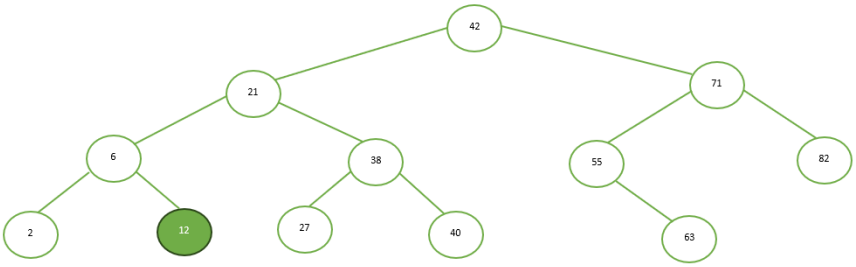
s. Mengambil data dari localRoot(node sekarang) untuk dicetak

```
    public void inorderHelper(TreeNode localRoot) {
95      if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
            System.out.print(localRoot.getData() + " ");
98          inorderHelper(localRoot: localRoot.rightNode);
99      }
100 }
```

t. Mencetak data localRoot yang telah diambil

```
    public void inorderHelper(TreeNode localRoot) {
95      if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
            System.out.print(localRoot.getData() + " ");
98          inorderHelper(localRoot: localRoot.rightNode);
99      }
100 }
101
    public void inorderHelper(TreeNode localRoot) {
95      if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
97          System.out.print(localRoot.getData() + " ");
            inorderHelper(localRoot: localRoot.rightNode);
99      }
100 }
101
```

```
debug:
data yang ditambahkan = 42, 21, 38, 27, 71, 82, 55, 63, 6, 2, 40 dan 12
Preorder = 42 21 6 2 12 38 27 40 71 55 63 82
Inorder = 2 6
```



u. Memeriksa apakah node sekarang null atau tidak

```
    public void inorderHelper(TreeNode localRoot) {
        if (localRoot != null) {
            inorderHelper(localRoot: localRoot.leftNode);
97          System.out.print(localRoot.getData() + " ");
            inorderHelper(localRoot: localRoot.rightNode);
99      }
100 }
```

v. Melakukan operasi rekursif dengan memanggil method inorderHelper dengan parameter localRoot dari leftNode sekarang

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
97              System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
101
```

w. Memeriksa apakah node sekarang null atau tidak. leftNode dari node 12 null

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
97              System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
```

```
        public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
97              System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
101
```

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
97              System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
101
```

x. Mencetak node yang sekarang

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
                System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
```

y. Mengambil data dari localRoot(node sekarang) untuk dicetak

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
                System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
101
```

z. Mencetak data localRoot yang telah diambil

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
                inorderHelper(localRoot: localRoot.leftNode);
                System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
101
```

```
debug:
data yang ditambahkan = 42, 21, 38, 27, 71, 82, 55, 63, 6, 2, 40 dan 12
Preorder = 42 21 6 2 12 38 27 40 71 55 63 82
Inorder = 2 6 12
```

```
95      public void inorderHelper(TreeNode localRoot) {
            if (localRoot != null) {
97              inorderHelper(localRoot: localRoot.leftNode);
                System.out.print(localRoot.getData() + " ");
                inorderHelper(localRoot: localRoot.rightNode);
99          }
100     }
```