Kelas BinaryTree2:

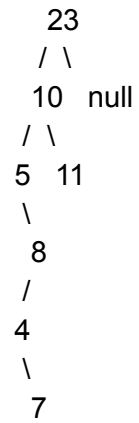
1. `private BTNode2<Integer> root;`: Deklarasi variabel root yang merupakan akar dari pohon biner.
2. `public BinaryTree2() { root = null; }`: Konstruktor kelas BinaryTree2 yang menginisialisasi root menjadi null.
3. `public boolean isEmpty() { return root == null; }`: Fungsi untuk memeriksa apakah pohon biner kosong atau tidak.
4. `public void insert(Integer data) { root = insert(root, data); }`: Fungsi untuk menyisipkan elemen baru ke dalam pohon biner.
5. `private BTNode2<Integer> insert(BTNode2<Integer> node, Integer data) { ... }`: Fungsi rekursif untuk menyisipkan elemen baru ke dalam pohon biner.
6. `public int countNodes() { return countNodes(root); }`: Fungsi untuk menghitung jumlah node dalam pohon biner.
7. `private int countNodes(BTNode2<Integer> r) { ... }`: Fungsi rekursif untuk menghitung jumlah node dalam pohon biner.
8. `public boolean search(Integer val) { return search(root, val); }`: Fungsi untuk mencari elemen dalam pohon biner.
9. `private boolean search(BTNode2<Integer> r, Integer val) { ... }`: Fungsi rekursif untuk mencari elemen dalam pohon biner.
10. `public void inorder() { inorder(root); }`: Fungsi untuk melakukan traversal inorder pada pohon biner.
11. `private void inorder(BTNode2<Integer> r) { ... }`: Fungsi rekursif untuk traversal inorder.
12. `public void preorder() { preorder(root); }`: Fungsi untuk melakukan traversal preorder pada pohon biner.
13. `private void preorder(BTNode2<Integer> r) { ... }`: Fungsi rekursif untuk traversal preorder.
14. `public void postorder() { postorder(root); }`: Fungsi untuk melakukan traversal postorder pada pohon biner.
15. `private void postorder(BTNode2<Integer> r) { ... }`: Fungsi rekursif untuk traversal postorder.

Kelas BTNode2:

1. `private BTNode2<Integer> left, right;`: Variabel left dan right yang merupakan anak kiri dan anak kanan dari node.
2. `private Integer data;`: Variabel data yang menyimpan nilai dari node.
3. `public BTNode2() { ... }`: Konstruktor tanpa parameter yang menginisialisasi left, right, dan data menjadi null.
4. `public BTNode2(Integer item) { ... }`: Konstruktor dengan parameter yang menginisialisasi data dengan nilai dari parameter dan left serta right menjadi null.
5. Method setter dan getter untuk left, right, dan data.

Jelaskan Pohon yang Terbentuk:

Dengan inputan data 23 – 10 – 5 – 8 – 11 – 4 – 7, pohon biner yang terbentuk akan memiliki struktur seperti berikut:



Penjelasan Traversal:

1. **Preorder Traversal**: Menelusuri pohon dari root, lalu ke kiri, dan kemudian ke kanan. Jadi urutannya adalah root, kiri, kanan. Dalam contoh pohon di atas, preorder traversal akan menghasilkan urutan: 23 - 10 - 5 - 8 - 4 - 7 - 11.
2. **Inorder Traversal**: Menelusuri pohon dari kiri, root, dan kemudian ke kanan. Jadi urutannya adalah kiri, root, kanan. Dalam contoh pohon di atas, inorder traversal akan menghasilkan urutan: 5 - 8 - 4 - 7 - 10 - 11 - 23.
3. **Postorder Traversal**: Menelusuri pohon dari kiri, kanan, dan kemudian ke root. Jadi urutannya adalah kiri, kanan, root. Dalam contoh pohon di atas, postorder traversal akan menghasilkan urutan: 7 - 4 - 8 - 5 - 11 - 10 - 23.