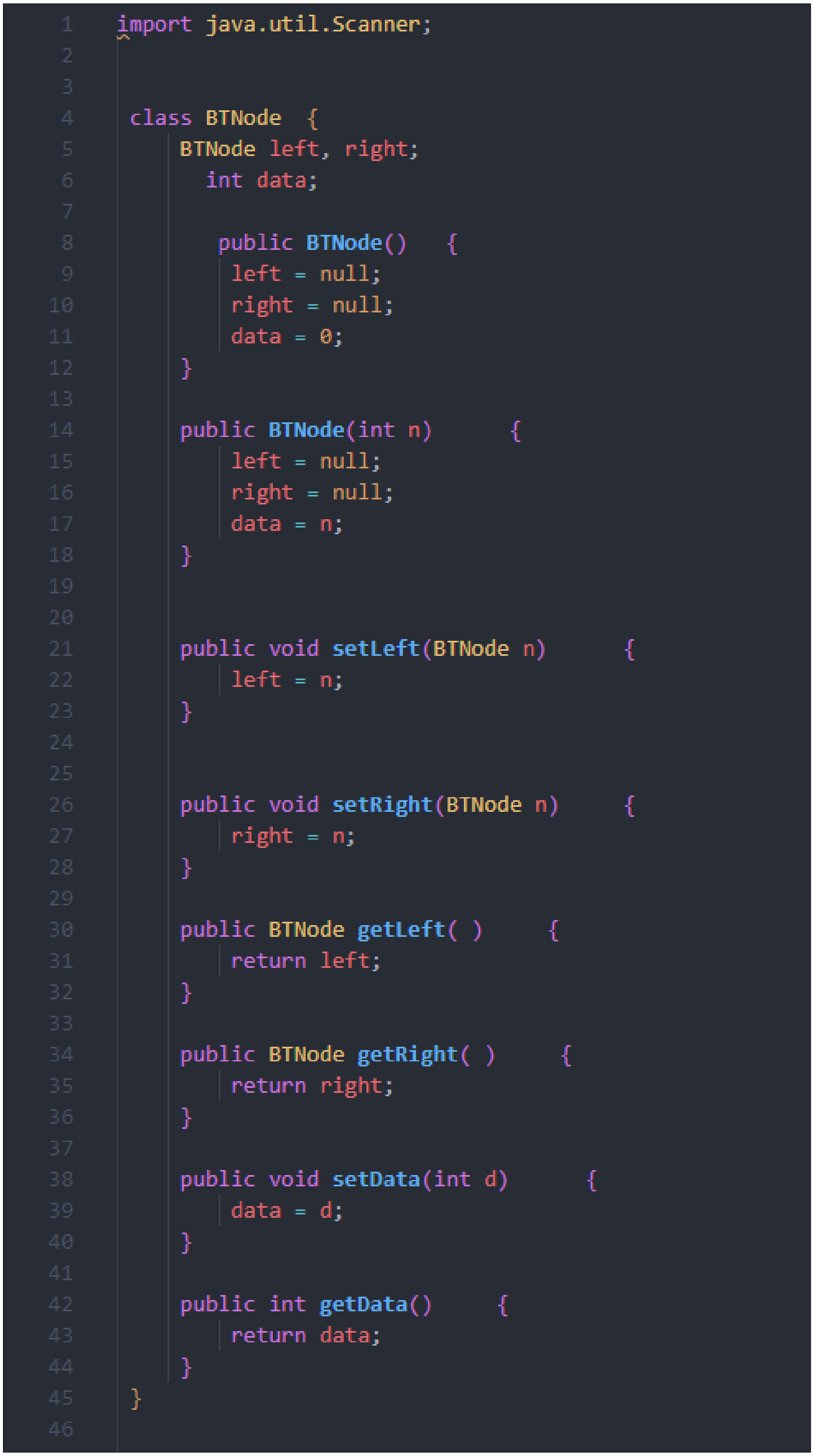
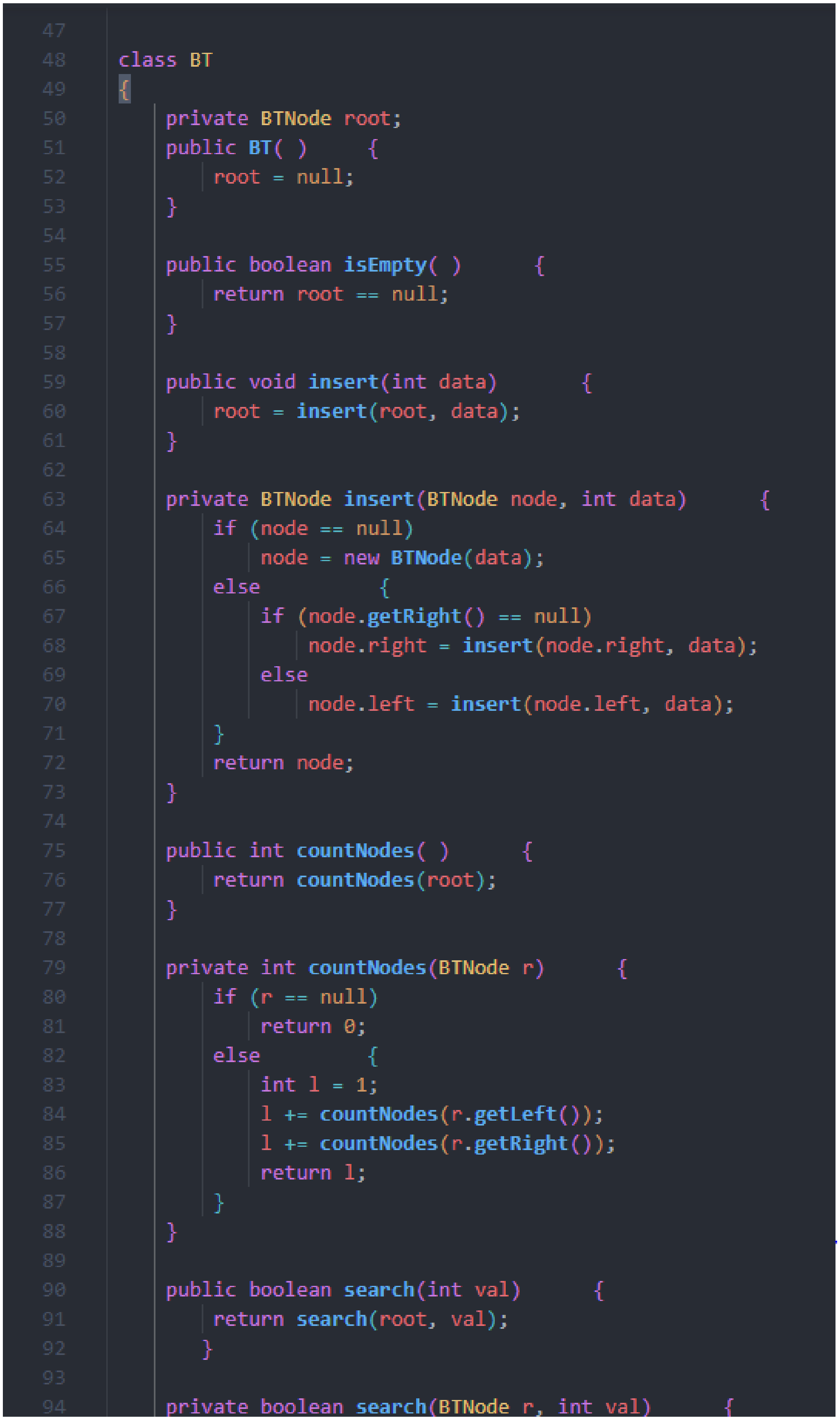
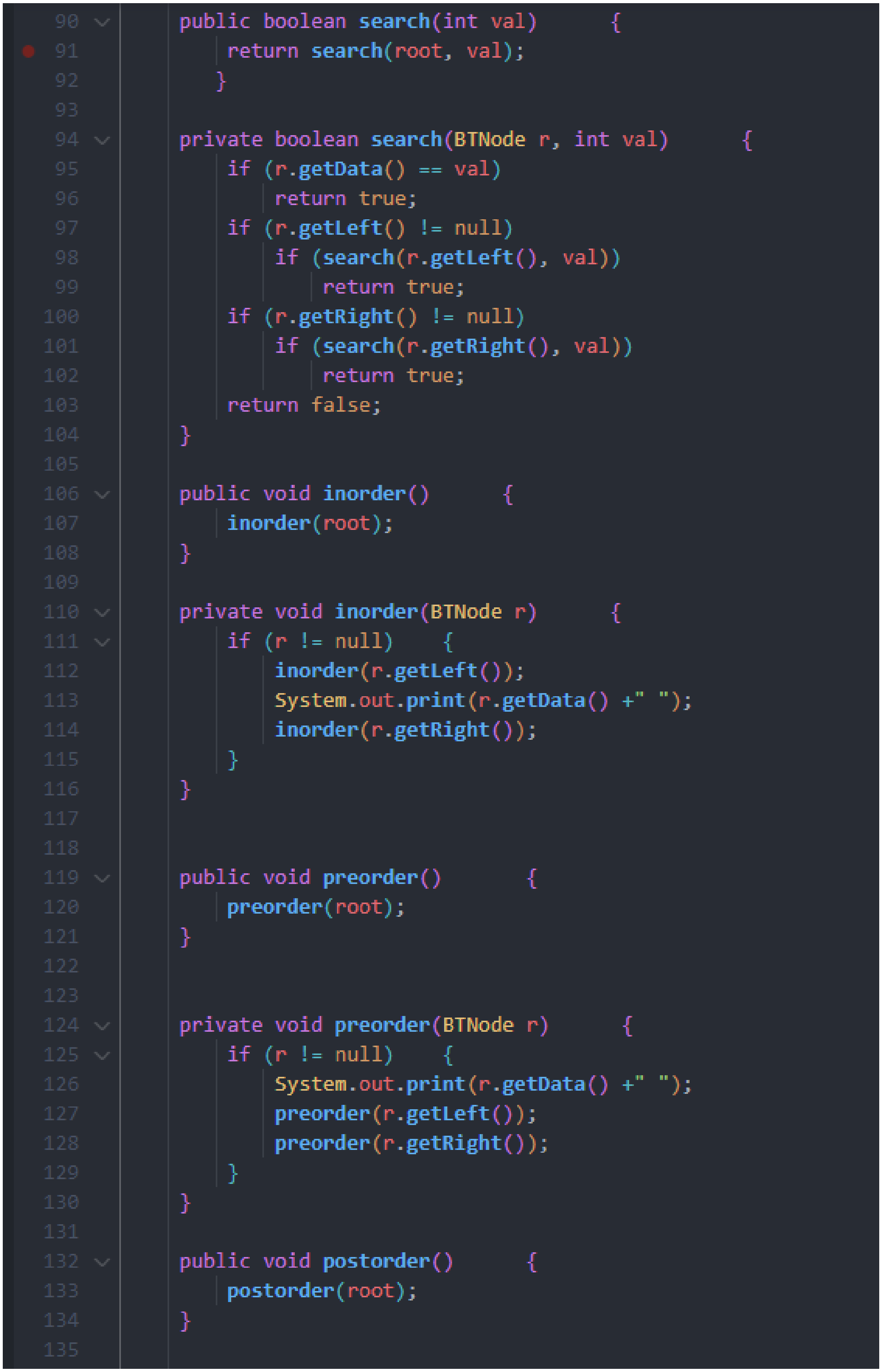


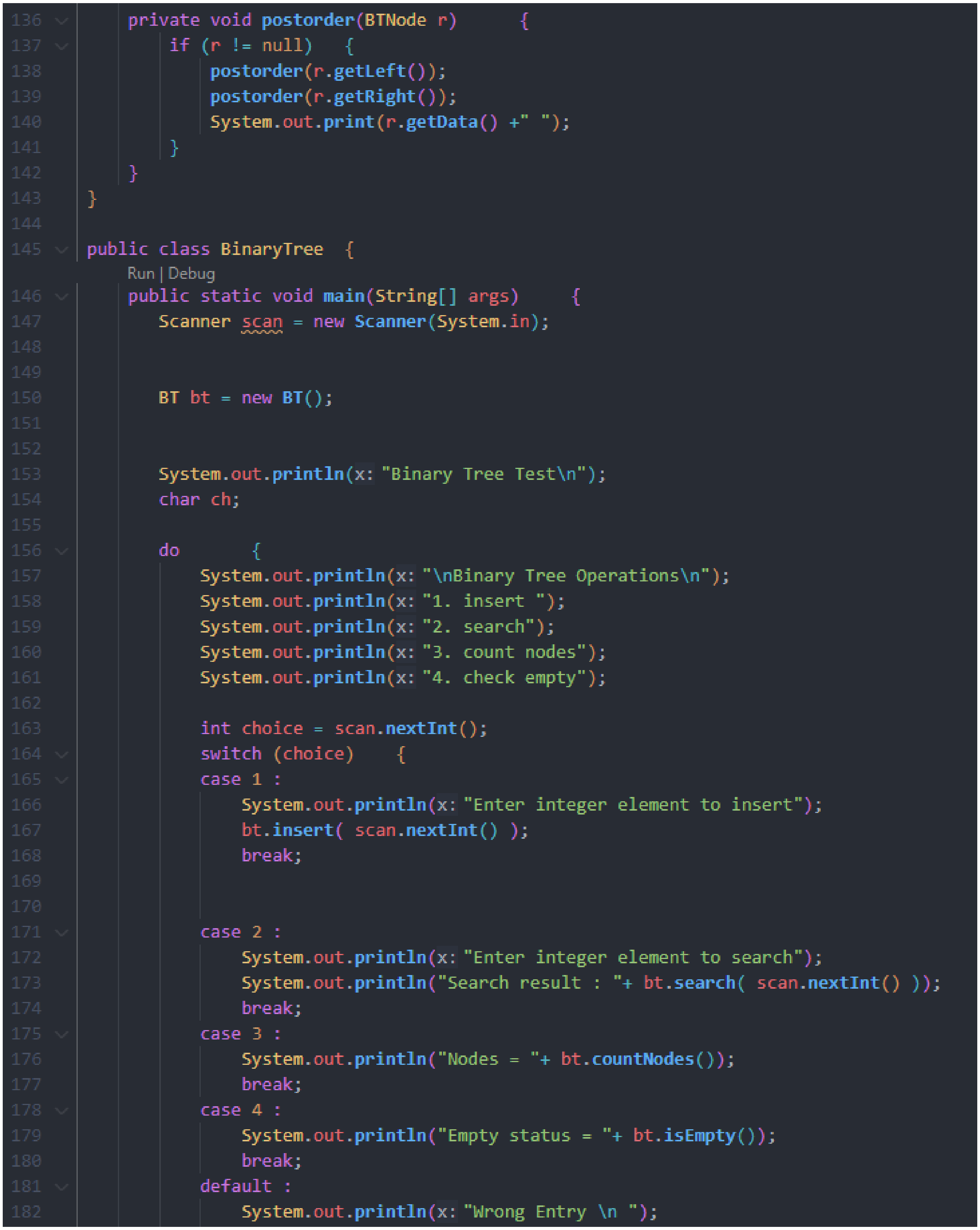
```java
import java.util.Scanner;


 class BTNode  {
     BTNode left, right;
       int data;

        public BTNode()   {
         left = null;
         right = null;
         data = 0;
     }

     public BTNode(int n)      {
         left = null;
         right = null;
         data = n;
     }


     public void setLeft(BTNode n)      {
         left = n;
     }


     public void setRight(BTNode n)     {
         right = n;
     }

     public BTNode getLeft( )     {
         return left;
     }

     public BTNode getRight( )     {
         return right;
     }

     public void setData(int d)      {
         data = d;
     }

     public int getData()     {
         return data;
     }
 }

```



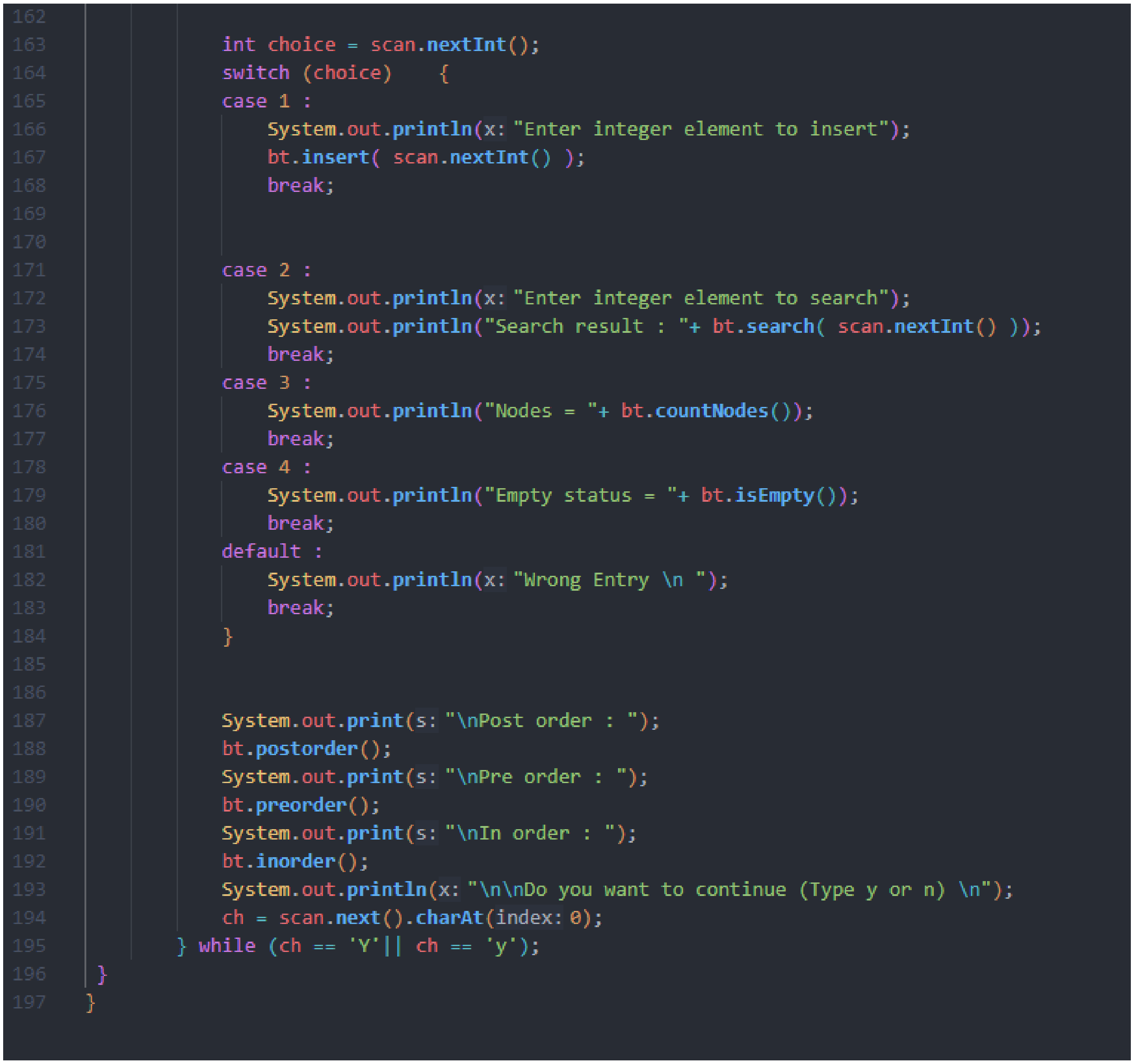
```
class BT
{
    private BTNode root;
    public BT( )     {
        root = null;
    }

    public boolean isEmpty( )      {
        return root == null;
    }

    public void insert(int data)       {
        root = insert(root, data);
    }

    private BTNode insert(BTNode node, int data)      {
        if (node == null)
            node = new BTNode(data);
        else          {
            if (node.getRight() == null)
                node.right = insert(node.right, data);
            else
                node.left = insert(node.left, data);
        }
        return node;
    }

    public int countNodes( )      {
        return countNodes(root);
    }

    private int countNodes(BTNode r)      {
        if (r == null)
            return 0;
        else         {
            int l = 1;
            l += countNodes(r.getLeft());
            l += countNodes(r.getRight());
            return l;
        }
    }

    public boolean search(int val)      {
        return search(root, val);
       }

    private boolean search(BTNode r, int val)      {
```



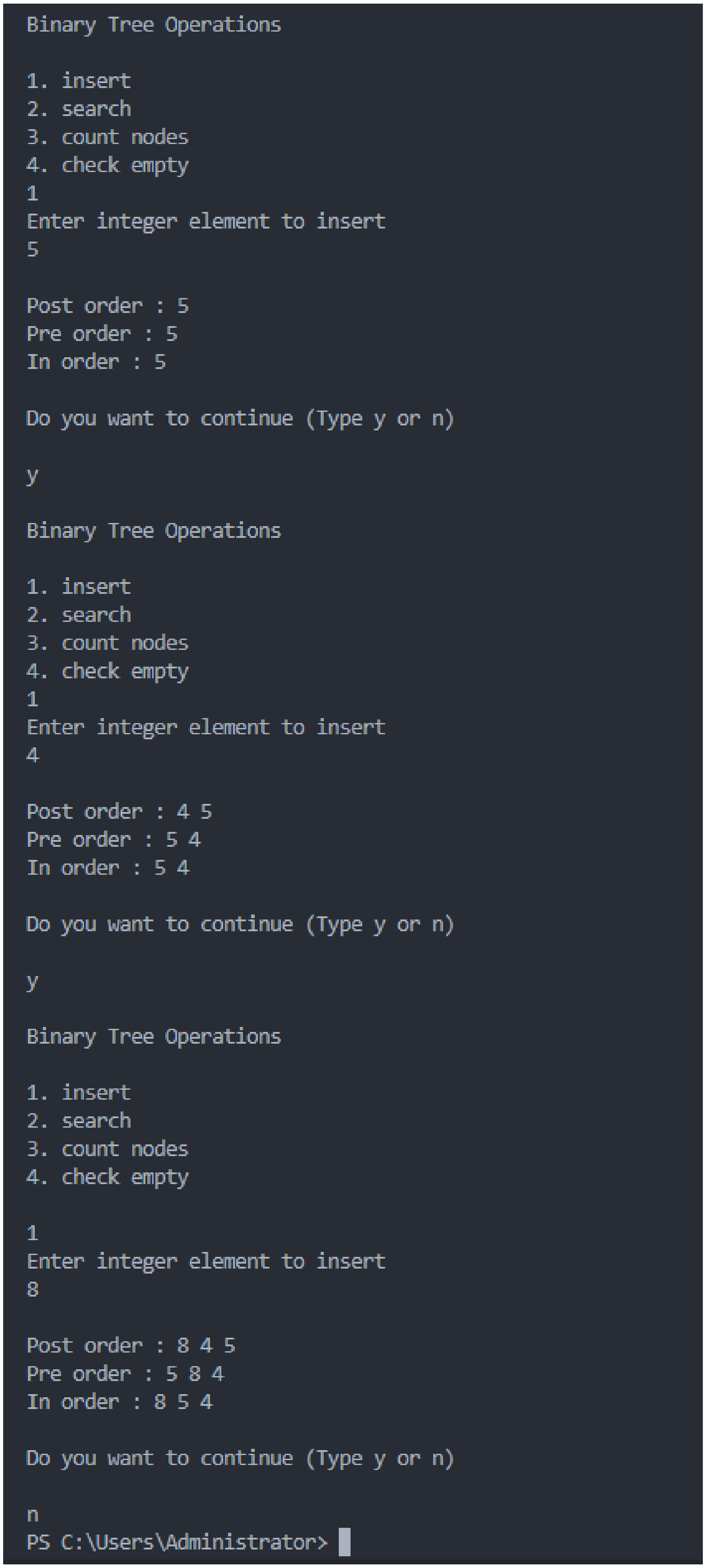
```java
public boolean search(int val)      {
    return search(root, val);
   }

private boolean search(BTNode r, int val)      {
    if (r.getData() == val)
        return true;
    if (r.getLeft() != null)
        if (search(r.getLeft(), val))
            return true;
    if (r.getRight() != null)
        if (search(r.getRight(), val))
            return true;
    return false;
}

public void inorder()      {
    inorder(root);
}

private void inorder(BTNode r)      {
    if (r != null)    {
        inorder(r.getLeft());
        System.out.print(r.getData() +" ");
        inorder(r.getRight());
    }
}


public void preorder()      {
    preorder(root);
}


private void preorder(BTNode r)      {
    if (r != null)    {
        System.out.print(r.getData() +" ");
        preorder(r.getLeft());
        preorder(r.getRight());
    }
}

public void postorder()      {
    postorder(root);
}

```



```
        private void postorder(BTNode r)      {
            if (r != null)   {
                postorder(r.getLeft());
                postorder(r.getRight());
                System.out.print(r.getData() +" ");
            }
        }
    }

    public class BinaryTree  {
        Run | Debug
        public static void main(String[] args)     {
           Scanner scan = new Scanner(System.in);


           BT bt = new BT();


           System.out.println(x: "Binary Tree Test\n");
           char ch;

           do       {
               System.out.println(x: "\nBinary Tree Operations\n");
               System.out.println(x: "1. insert ");
               System.out.println(x: "2. search");
               System.out.println(x: "3. count nodes");
               System.out.println(x: "4. check empty");

               int choice = scan.nextInt();
               switch (choice)    {
               case 1 :
                   System.out.println(x: "Enter integer element to insert");
                   bt.insert( scan.nextInt() );
                   break;


               case 2 :
                   System.out.println(x: "Enter integer element to search");
                   System.out.println("Search result : "+ bt.search( scan.nextInt() ));
                   break;
               case 3 :
                   System.out.println("Nodes = "+ bt.countNodes());
                   break;
               case 4 :
                   System.out.println("Empty status = "+ bt.isEmpty());
                   break;
               default :
                   System.out.println(x: "Wrong Entry \n ");
```



```
            int choice = scan.nextInt();
            switch (choice)    {
            case 1 :
                System.out.println(x: "Enter integer element to insert");
                bt.insert( scan.nextInt() );
                break;


            case 2 :
                System.out.println(x: "Enter integer element to search");
                System.out.println("Search result : "+ bt.search( scan.nextInt() ));
                break;
            case 3 :
                System.out.println("Nodes = "+ bt.countNodes());
                break;
            case 4 :
                System.out.println("Empty status = "+ bt.isEmpty());
                break;
            default :
                System.out.println(x: "Wrong Entry \n ");
                break;
            }


            System.out.print(s: "\nPost order : ");
            bt.postorder();
            System.out.print(s: "\nPre order : ");
            bt.preorder();
            System.out.print(s: "\nIn order : ");
            bt.inorder();
            System.out.println(x: "\n\nDo you want to continue (Type y or n) \n");
            ch = scan.next().charAt(index: 0);
        } while (ch == 'Y'|| ch == 'y');
 }
}
```



```
Binary Tree Operations

1. insert
2. search
3. count nodes
4. check empty
1
Enter integer element to insert
5

Post order : 5
Pre order : 5
In order : 5

Do you want to continue (Type y or n)

y

Binary Tree Operations

1. insert
2. search
3. count nodes
4. check empty
1
Enter integer element to insert
4

Post order : 4 5
Pre order : 5 4
In order : 5 4

Do you want to continue (Type y or n)

y

Binary Tree Operations

1. insert
2. search
3. count nodes
4. check empty

1
Enter integer element to insert
8

Post order : 8 4 5
Pre order : 5 8 4
In order : 8 5 4

Do you want to continue (Type y or n)

n
PS C:\Users\Administrator> █
```