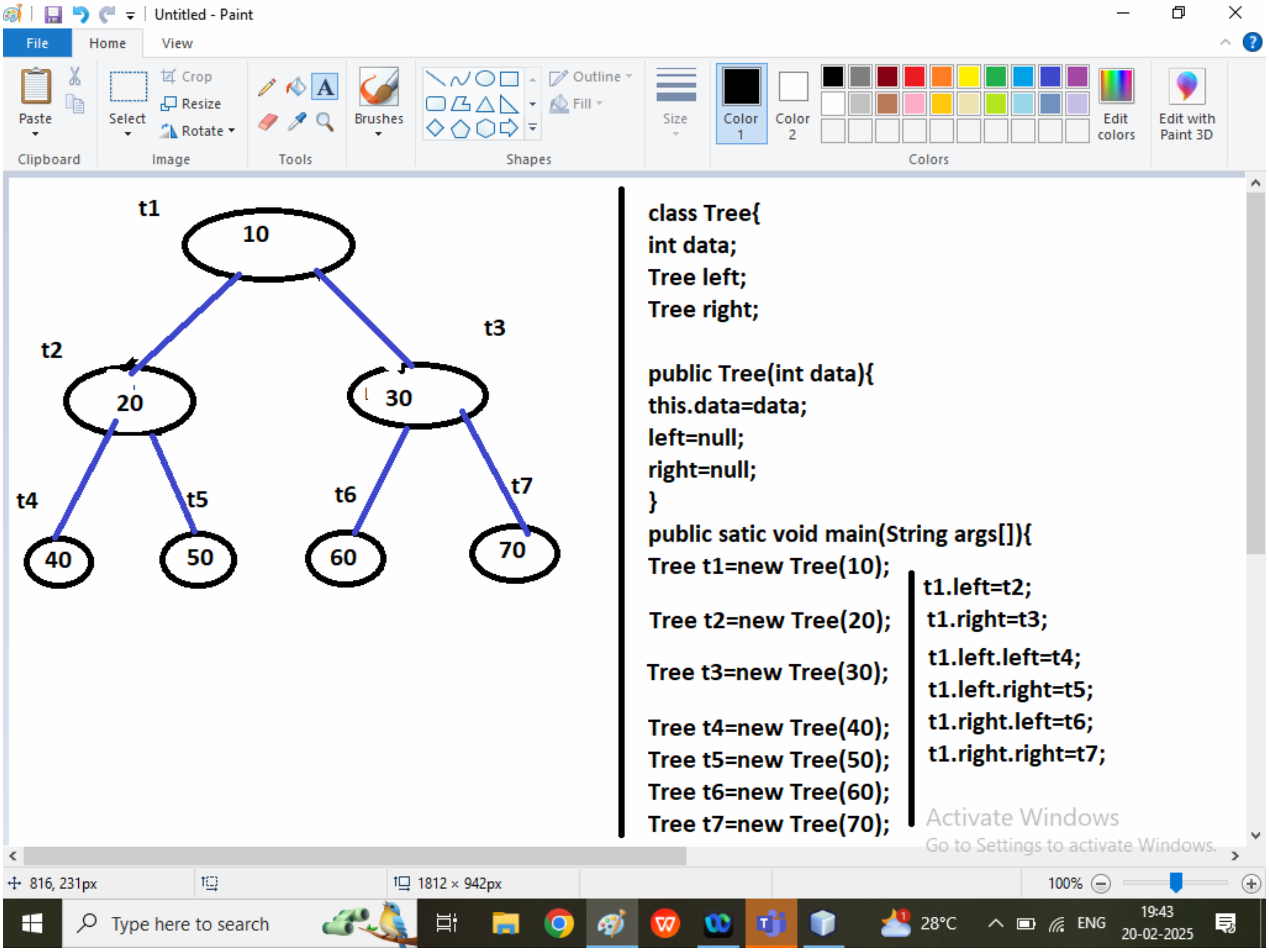
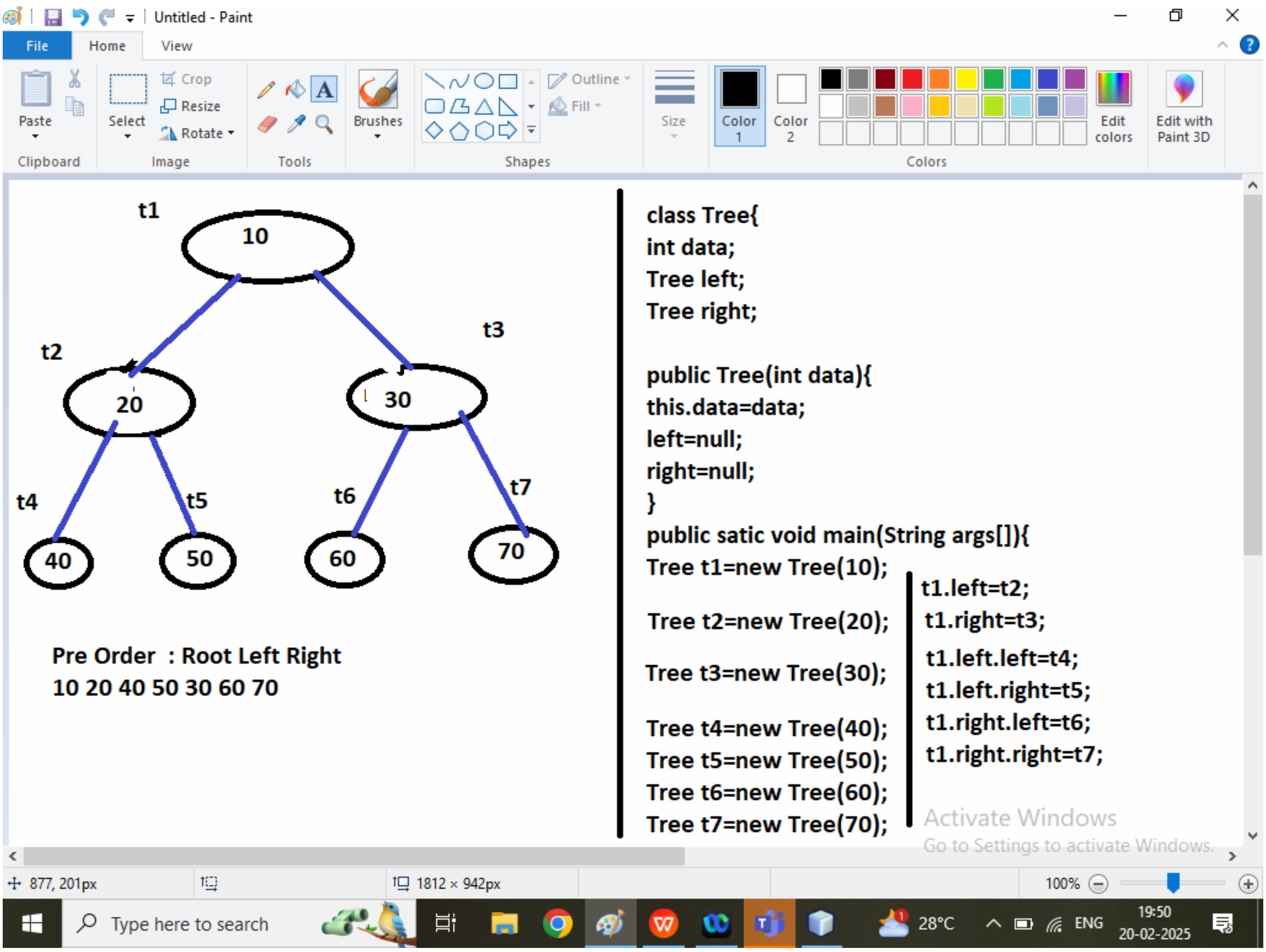
Q1. Write a java Program to constructor a tree and Traverse the element of tree using Pre Order, In Order and Post Order



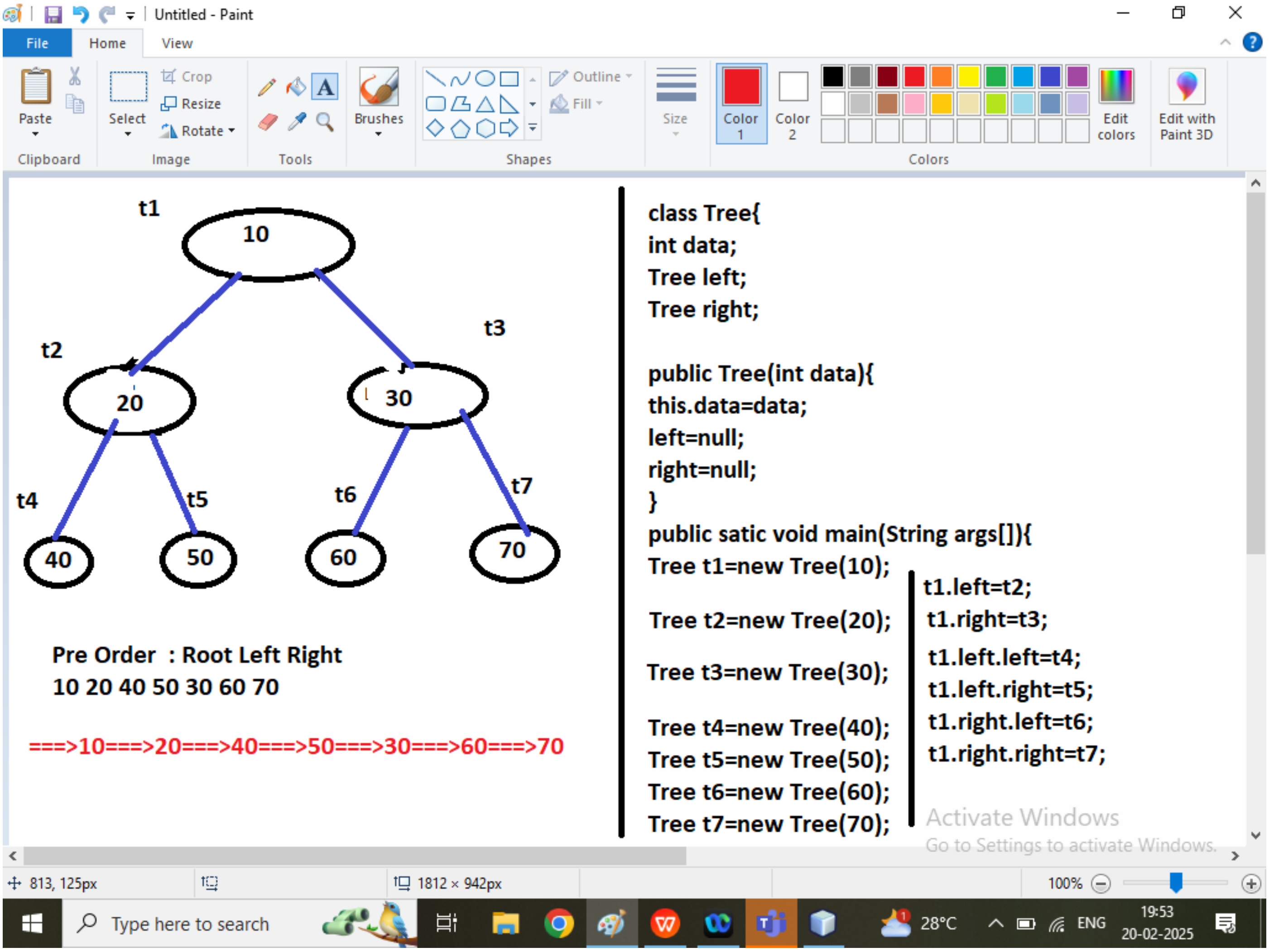
```
class Tree{
int data;
Tree left;
Tree right;

public Tree(int data){
this.data=data;
left=null;
right=null;
}
public satic void main(String args[]){
Tree t1=new Tree(10);

Tree t2=new Tree(20);

Tree t3=new Tree(30);

Tree t4=new Tree(40);
Tree t5=new Tree(50);
Tree t6=new Tree(60);
Tree t7=new Tree(70);
t1.left=t2;
t1.right=t3;

t1.left.left=t4;
t1.left.right=t5;
t1.right.left=t6;
t1.right.right=t7;
```



Pre Order : Root Left Right
10 20 40 50 30 60 70

```
class Tree{
int data;
Tree left;
Tree right;

public Tree(int data){
this.data=data;
left=null;
right=null;
}
public satic void main(String args[]){
Tree t1=new Tree(10);

Tree t2=new Tree(20);

Tree t3=new Tree(30);

Tree t4=new Tree(40);
Tree t5=new Tree(50);
Tree t6=new Tree(60);
Tree t7=new Tree(70);
```

```
t1.left=t2;
t1.right=t3;
t1.left.left=t4;
t1.left.right=t5;
t1.right.left=t6;
t1.right.right=t7;
```

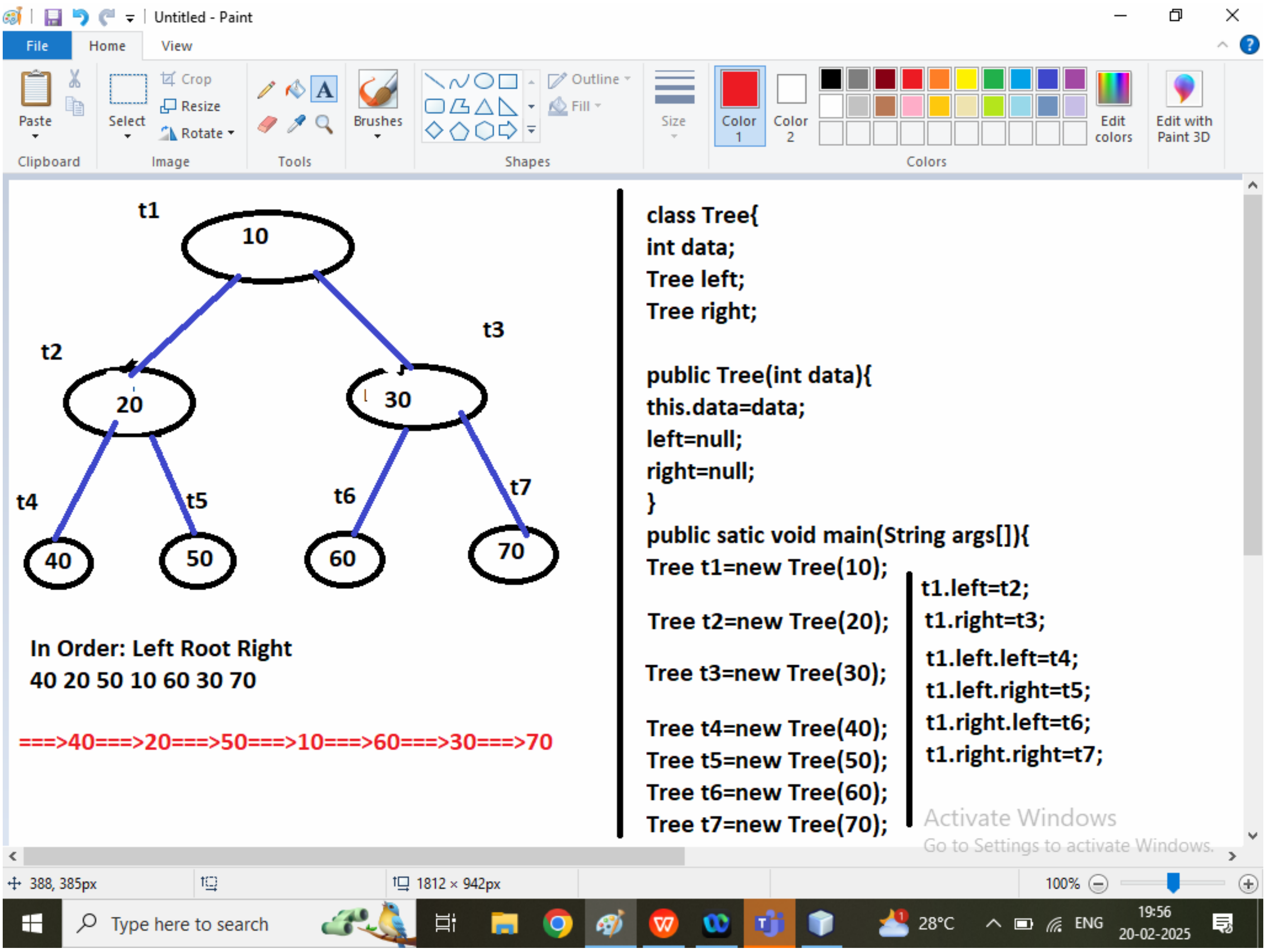


Pre Order : Root Left Right
10 20 40 50 30 60 70

===>10===>20===>40===>50===>30===>60===>70

```
class Tree{
int data;
Tree left;
Tree right;

public Tree(int data){
this.data=data;
left=null;
right=null;
}
public satic void main(String args[]){
Tree t1=new Tree(10);

Tree t2=new Tree(20);

Tree t3=new Tree(30);

Tree t4=new Tree(40);
Tree t5=new Tree(50);
Tree t6=new Tree(60);
Tree t7=new Tree(70);
```

```
t1.left=t2;
t1.right=t3;
t1.left.left=t4;
t1.left.right=t5;
t1.right.left=t6;
t1.right.right=t7;
```



In Order: Left Root Right
40 20 50 10 60 30 70

===>40===>20===>50===>10===>60===>30===>70

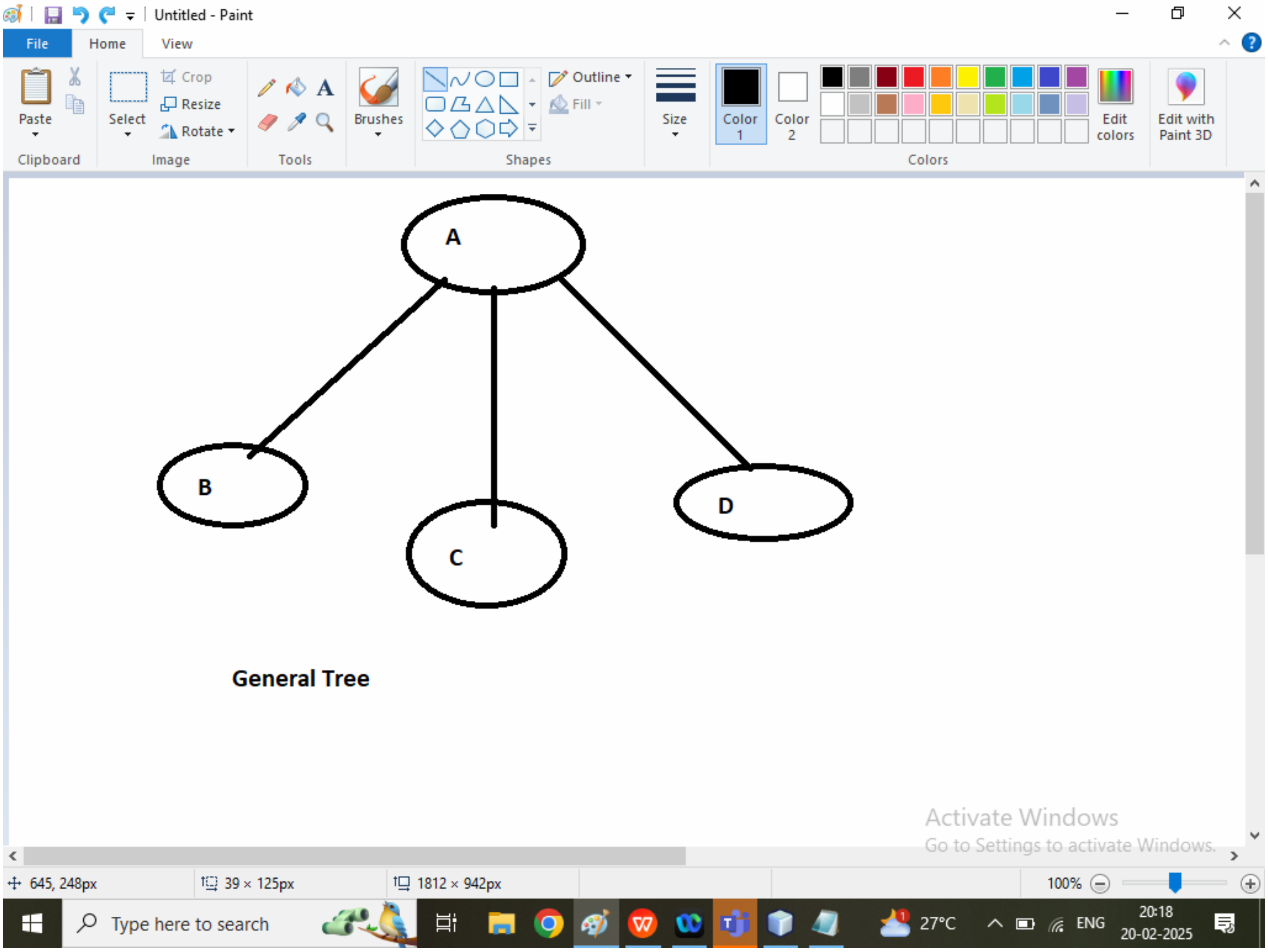
```
class Tree{
int data;
Tree left;
Tree right;

public Tree(int data){
this.data=data;
left=null;
right=null;
}
public satic void main(String args[]){
Tree t1=new Tree(10);

Tree t2=new Tree(20);

Tree t3=new Tree(30);

Tree t4=new Tree(40);
Tree t5=new Tree(50);
Tree t6=new Tree(60);
Tree t7=new Tree(70);
```

```
t1.left=t2;
t1.right=t3;
t1.left.left=t4;
t1.left.right=t5;
t1.right.left=t6;
t1.right.right=t7;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dsaj6;

/**
 *
 * @author Admin
 */
public class Tree {

    public int data;
    Tree left;
    Tree right;

    public Tree(int data) {
        this.data = data;
        left = null;
        right = null;
    }

    public static void preOrder(Tree root) {
        if (root == null) {
            return;
        }
        //ROOT LEFT RIGHT
        System.out.print("===>" + root.data);//ROOT
        preOrder(root.left);//for Left Sub Tree
        preOrder(root.right);

    }

     public static void inOrder(Tree root) {
        if (root == null) {
            return;
        }
        // LEFT ROOT RIGHT

        inOrder(root.left);//for Left Sub Tree
         System.out.print("===>" + root.data);//ROOT
        inOrder(root.right);//Right Sub Tree

    }

     public static void postOrder(Tree root) {
        if (root == null) {
            return;
        }
        // LEFT RIGHT  ROOT

         postOrder(root.left);//for Left Sub Tree

        postOrder(root.right);//Right Sub Tree
         System.out.print("===>" + root.data);//ROOT

    }
    public static void main(String[] args) {
        Tree t1 = new Tree(10);
        Tree t2 = new Tree(20);
        t1.left = t2;
        Tree t3 = new Tree(30);
        t1.right = t3;
        Tree t4 = new Tree(40);
        t1.left.left = t4;
        Tree t5 = new Tree(50);
        t1.left.right = t5;
        Tree t6 = new Tree(60);
        t1.right.left = t6;
        Tree t7 = new Tree(70);
        t1.right.right = t7;
        System.out.println("Print Data of Tree Using Pre order Tree Trevarsal ");
        preOrder(t1);
     System.out.println("\nPrint Data of Tree Using In order Tree Trevarsal ");
        inOrder(t1);
            System.out.println("\nPrint Data of Tree Using Post order Tree Trevarsal ");
        postOrder(t1);

    }
}
```

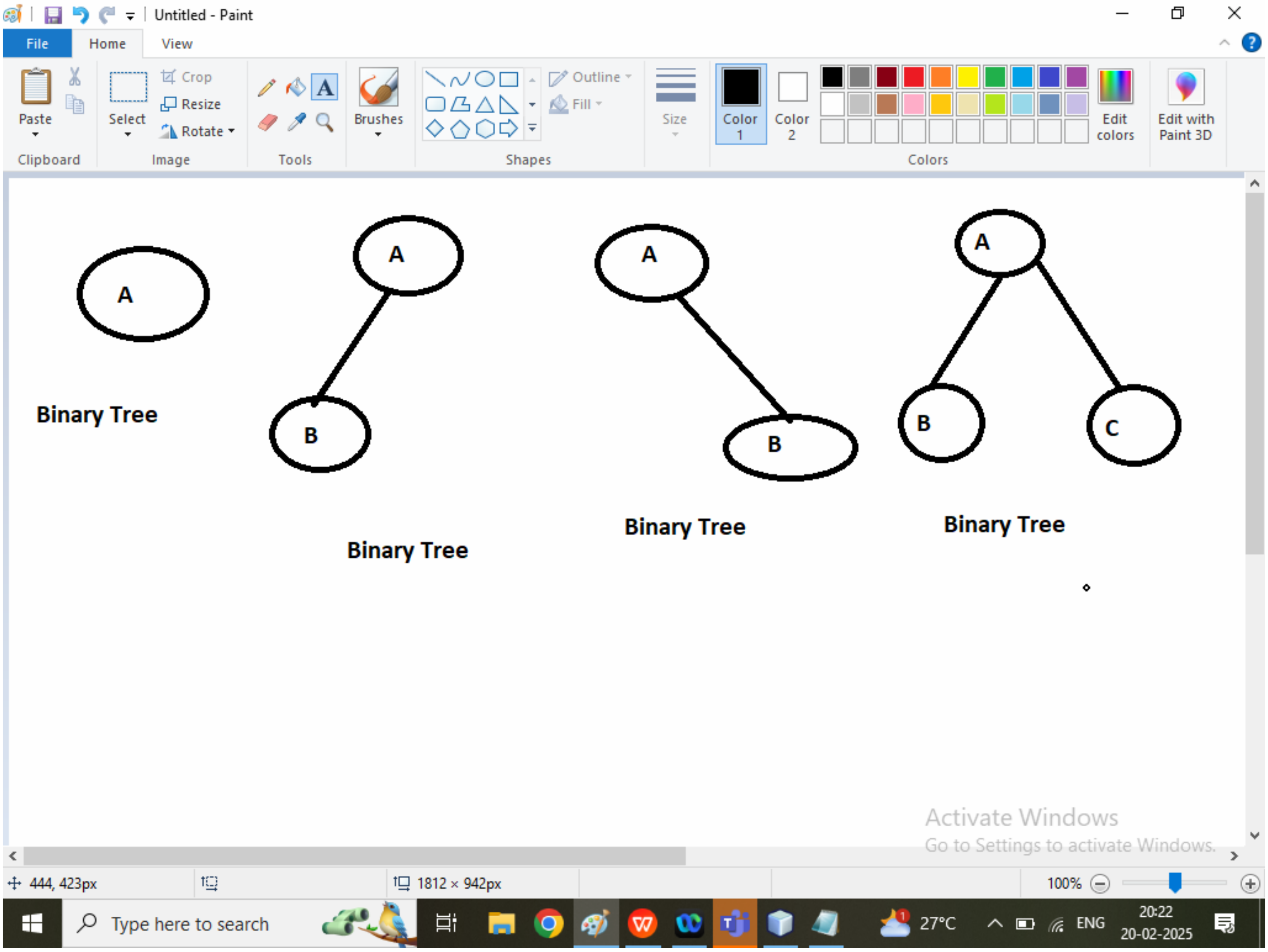
## Q2. Explain types  of tree  in data Structure?

1. General Tree: A tree can have any numbers of childern



Where General Tree is Used: XML Parsing

## 2. Binary Tree: Each Node has at most two children (left and right)

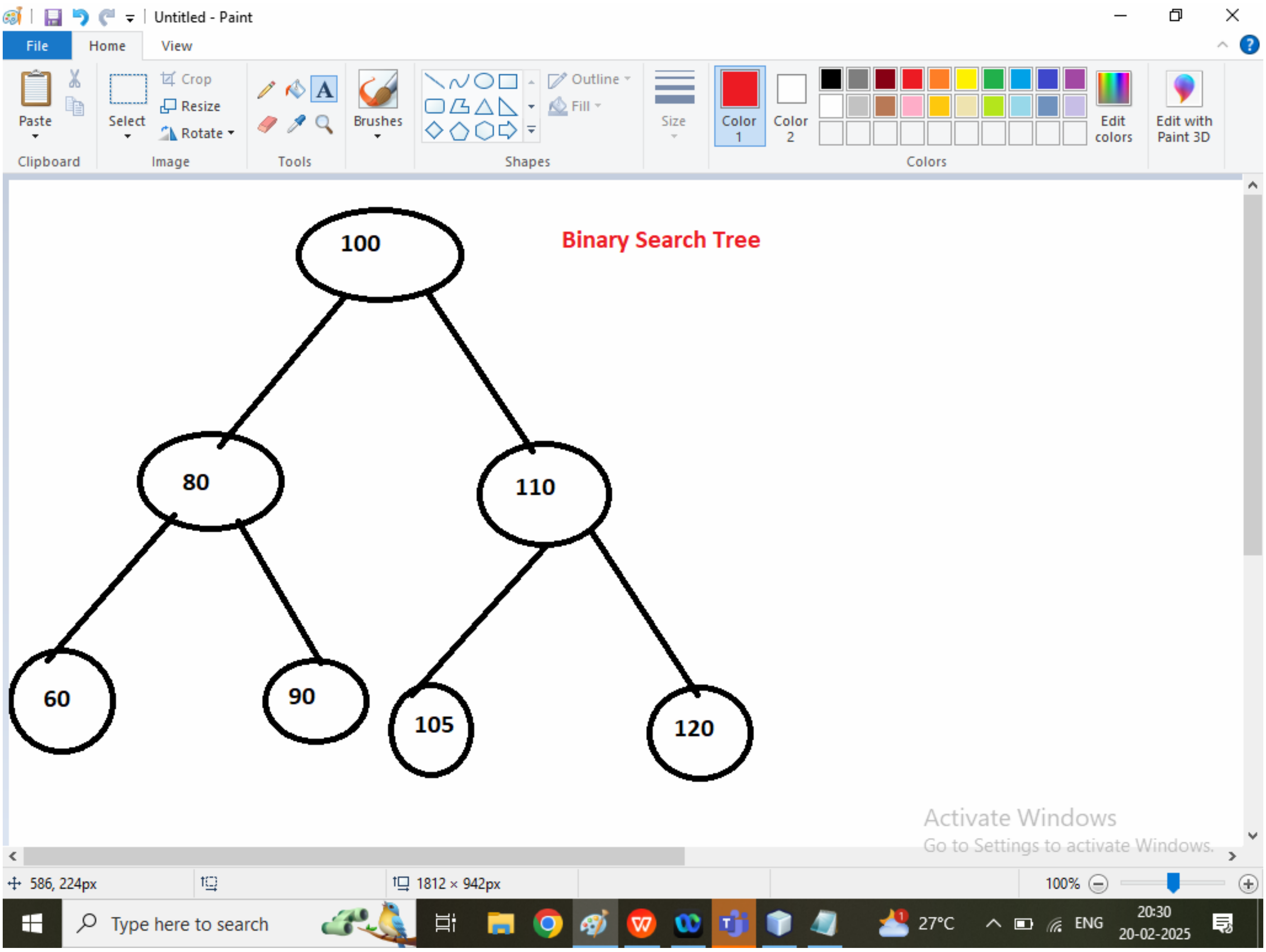


Where Binary Tree is used: expression tree, decision Tree

3. Binary Search Tree : A Sorted Binary Tree where

A. Left Sub tree contains smaller values then root value

B. Right Sub tree contains greater than or equal values then root value



4. Complete Binary Tree
5. Full Binary Tree
6. Perfect Binary Tree
7. Left Skewed Tree
8. Right Skewed  Tree
9. Balanced Binary [AVL Tree, Red Black Tree]
10. B-Tree
11. Heap (Max Heap, Min Heap)