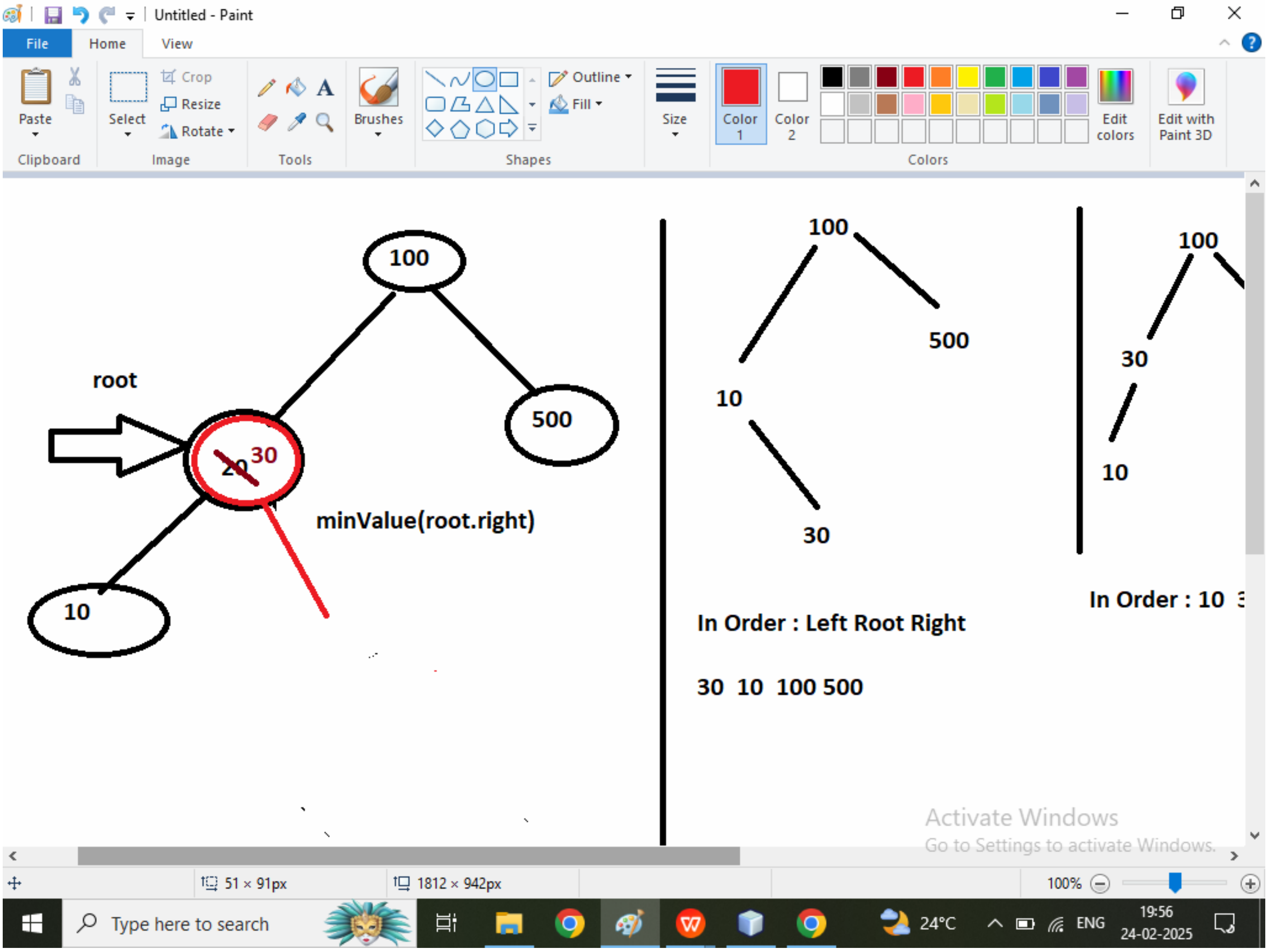
Q1. Delete Data in Binary Search Tree?

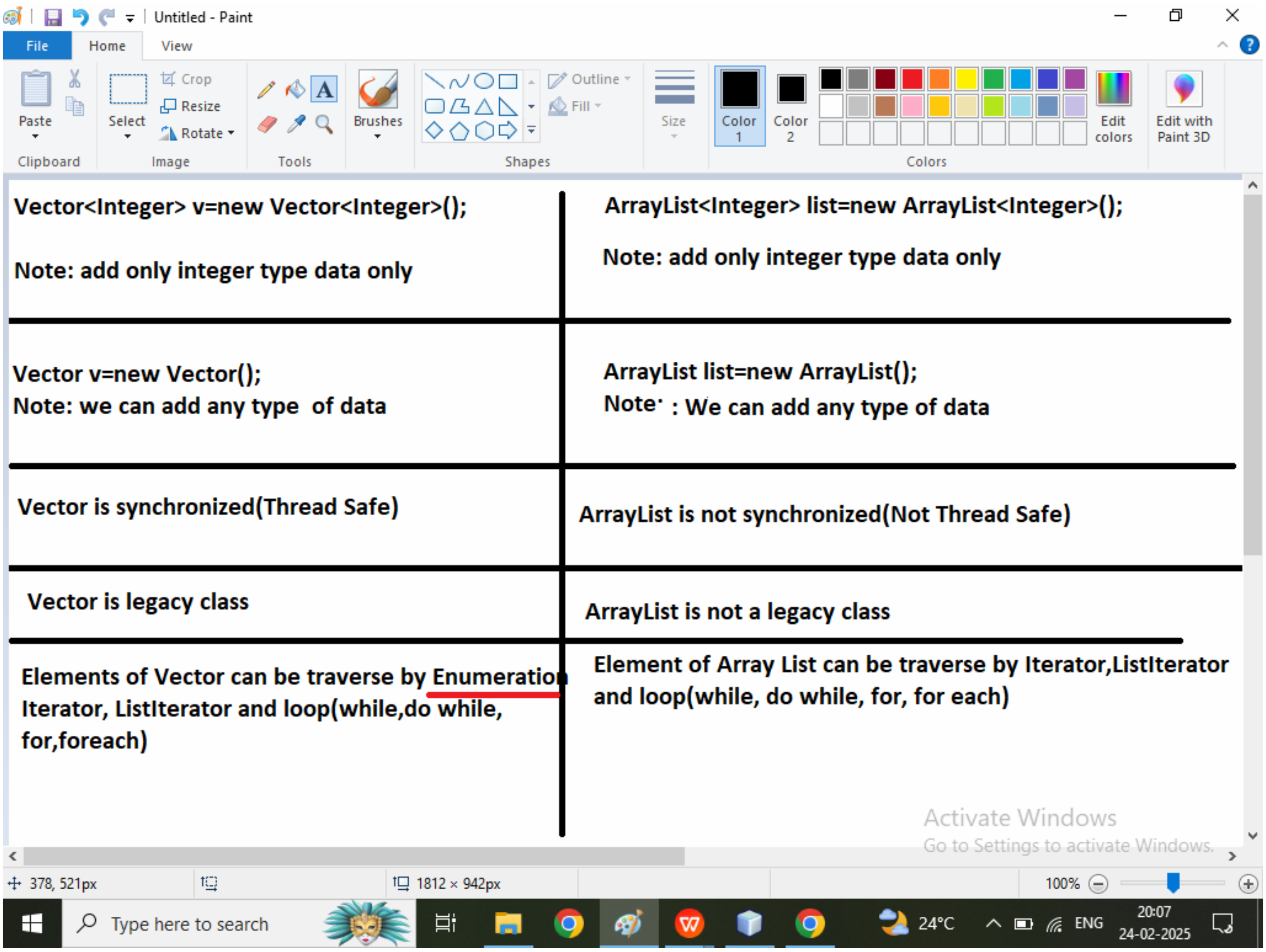
```
package dsaj6;

public class Node {

    int data;
    Node left;
    Node right;

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

    public static Node insert(Node root, int
data) {
        if (root == null) {
            return new Node(data);
        }
        if (data < root.data) {
            root.left = insert(root.left, data);
        } else {
            root.right = insert(root.right, data);
        }
        return root;
    }

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

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

        InOrder(root.left);//for Left sub Tree
        System.out.print("===>" + root.data);
        InOrder(root.right);//for right sub tree
    }

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

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

        postOrder(root.right);//for Right sub
Tree
        System.out.print("===>" + root.data);
    }
    public static Node delete(Node root, int
data) {
        if (root == null) {
            return root;
        }
        if (data < root.data) {
            root.left = delete(root.left, data);
        } else if (data > root.data) {
            root.right = delete(root.right, data);
        } else {
            //case 1 and 2 one or no child
            if (root.left == null) {
                return root.right;
            } else if (root.right == null) {
                return root.left;
            }
            //case 3 a node have left and right
child
            root.data = minValue(root.right);//
            root.right = delete(root.right,
root.data);
        }

        return root;
    }

    public static int minValue(Node root) {
        int minValue = root.data;
        while (root.left != null) {
            minValue = root.left.data;//30
        }
        return minValue;//30
    }

    public static void main(String[] args) {

        Node t1 = new Node(100);
        Node t2 = new Node(20);
        t1.left = t2;
        Node t3 = new Node(500);
        t1.right = t3;
        Node t4 = new Node(10);
        t1.left.left = t4;
        Node t5 = new Node(30);
        t1.left.right = t5;



        System.out.println("Print Before
Delete : ");
        InOrder(t1);

        delete(t1, 20);
        System.out.println("\nPrint data using
In Order Tree Traversal : ");
        InOrder(t1);

    }

}
```



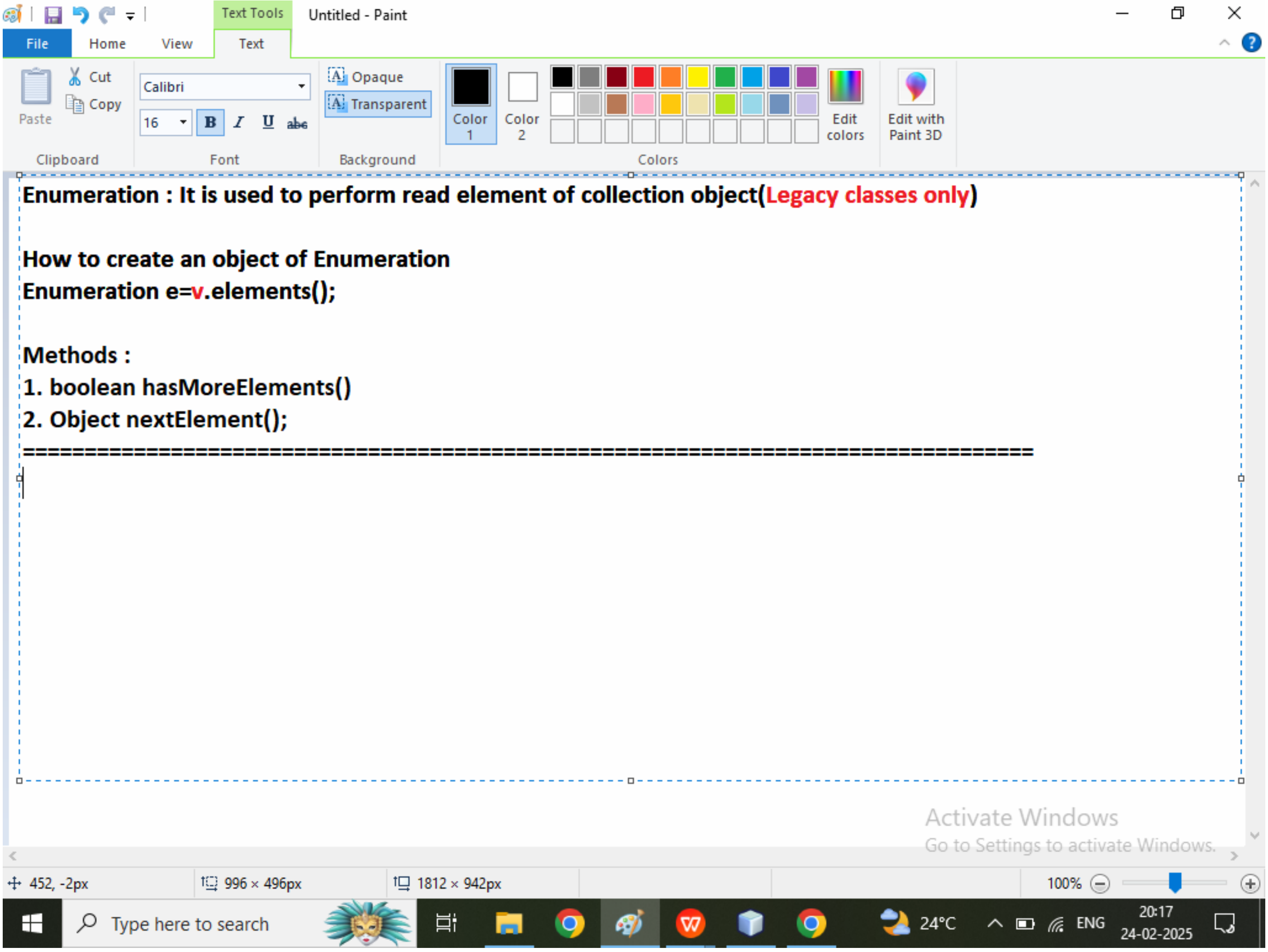
In Order : Left Root Right

30 10 100 500

In Order : 10 3



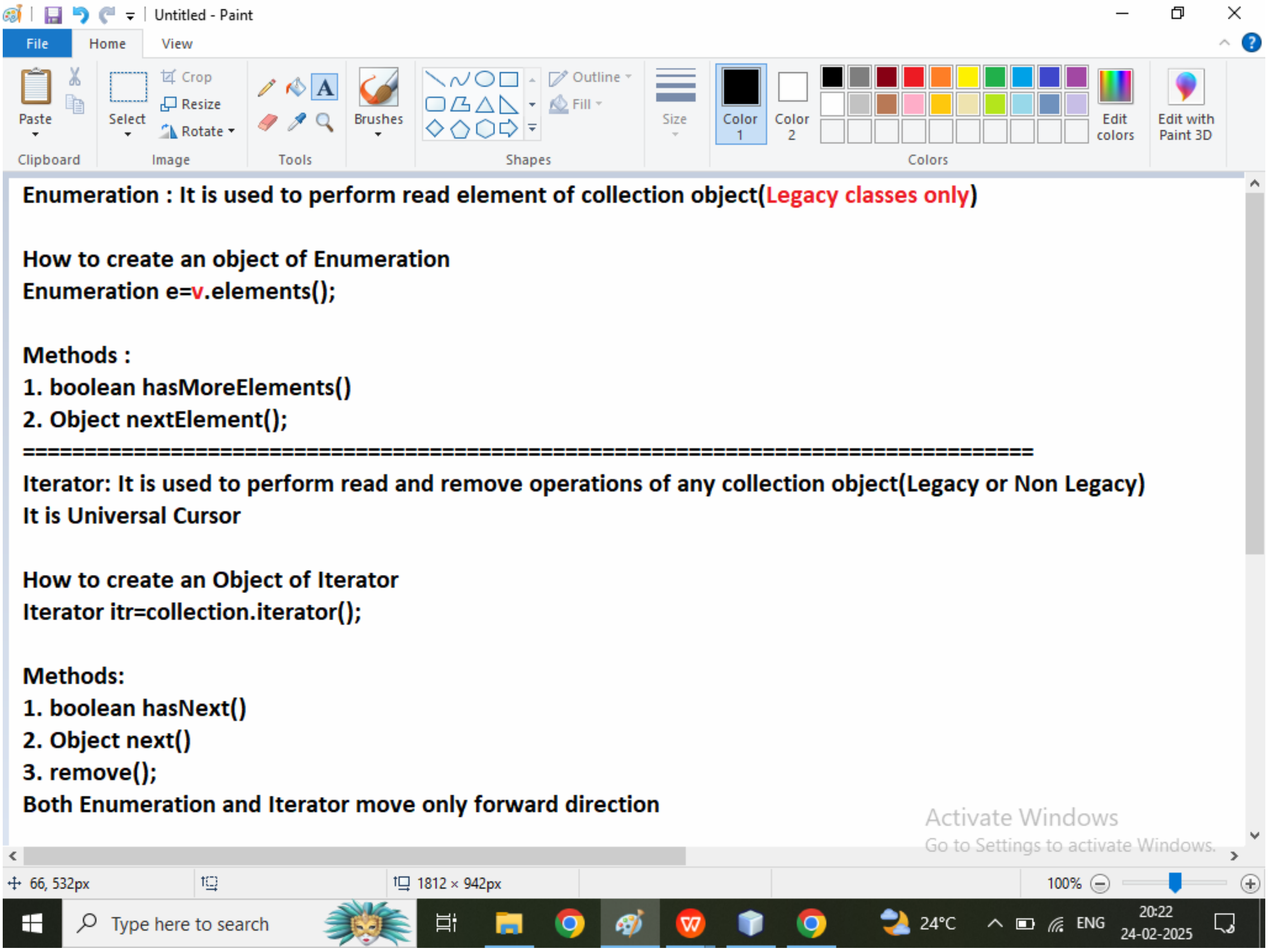
Vector<Integer> v=new Vector<Integer>(); **Note: add only integer type data only**	**ArrayList<Integer> list=new ArrayList<Integer>();** **Note: add only integer type data only**
Vector v=new Vector(); **Note: we can add any type of data**	**ArrayList list=new ArrayList();** **Note· : We can add any type of data**
Vector is synchronized(Thread Safe)	**ArrayList is not synchronized(Not Thread Safe)**
Vector is legacy class	**ArrayList is not a legacy class**
Elements of Vector can be traverse by Enumeration Iterator, ListIterator and loop(while,do while, for,foreach)	**Element of Array List can be traverse by Iterator,ListIterator and loop(while, do while, for, for each)**



Enumeration : It is used to perform read element of collection object(Legacy classes only)

How to create an object of Enumeration
Enumeration e=v.elements();

Methods :
1. boolean hasMoreElements()
2. Object nextElement();
==



Enumeration : It is used to perform read element of collection object(Legacy classes only)

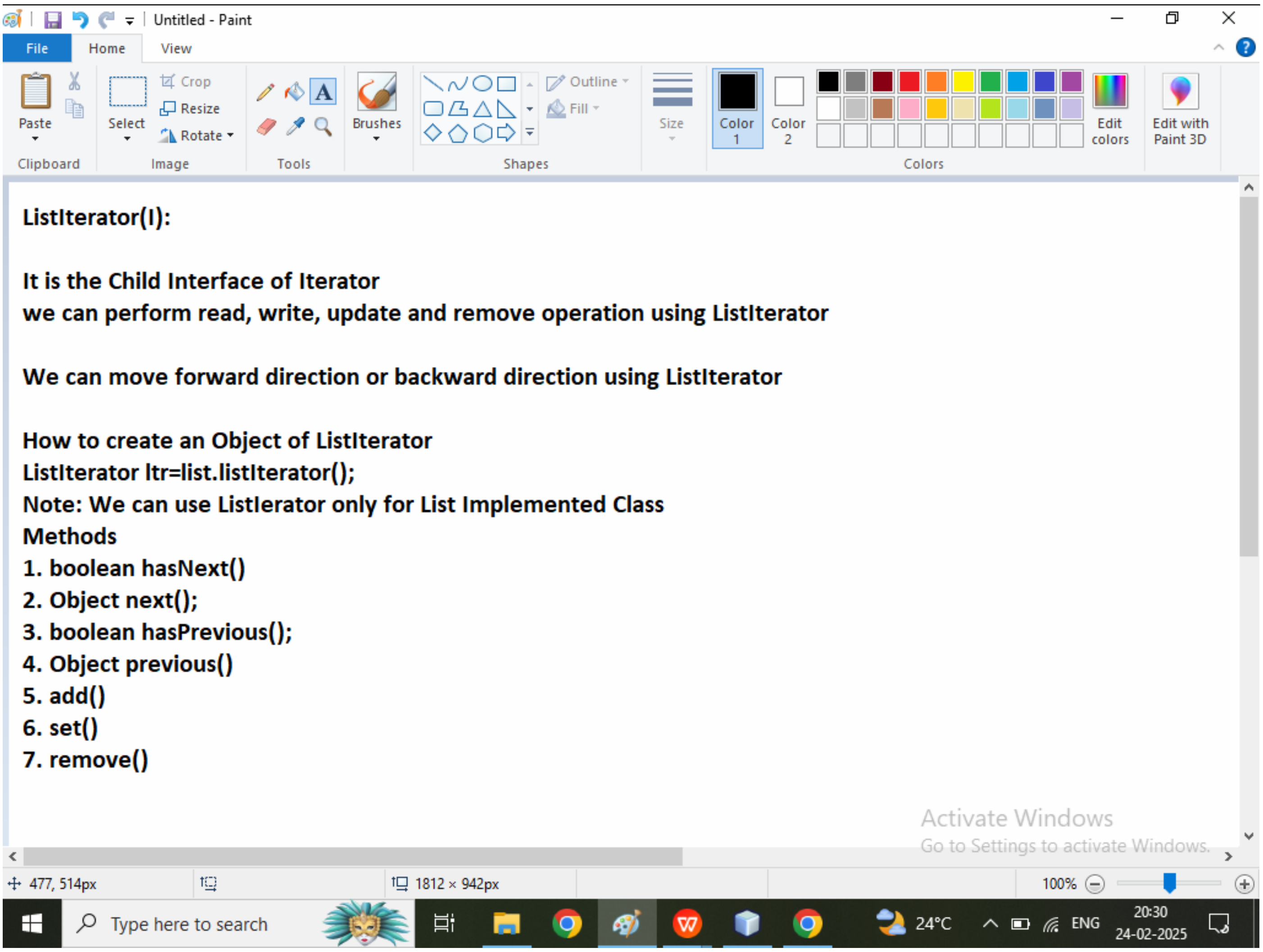
How to create an object of Enumeration
Enumeration e=v.elements();

Methods :
1. boolean hasMoreElements()
2. Object nextElement();
===
Iterator: It is used to perform read and remove operations of any collection object(Legacy or Non Legacy)
It is Universal Cursor

How to create an Object of Iterator
Iterator itr=collection.iterator();

Methods:
1. boolean hasNext()
2. Object next()
3. remove();
Both Enumeration and Iterator move only forward direction

```
/*
 * 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 dsafeb2025;

import java.util.ArrayList;
import java.util.Iterator;

/**
 *
 * @author Admin
 */
public class A1 {
    public static void main(String[] args) {

        ArrayList list=new ArrayList();
        list.add(10);
        list.add("hello");
        list.add('A');
        list.add(true);
        list.add(123.45f);
        list.add(45.55555);
         list.add(10);

        System.out.println("Array List : "+list);
        System.out.println("Print Data of Array List Using for  loop ");
        for(int i=0;i<list.size();i++){
            System.out.print("==>"+list.get(i));
        }

        System.out.println("\nPrint Data of Array List Using for each ");
        for(Object x:list){
            System.out.print("==>"+x);
        }

        System.out.println("\nPrint Data of  ArrayList Using Iterator ");
        Iterator itr=list.iterator();
        while(itr.hasNext()){
       Object x=itr.next();
       if(x.equals(10)){
         itr.remove();
       }
            System.out.print("====>"+x);
        }
        System.out.println("\n----------------------------\n");
        System.out.println(""+list);
    }

}
```



ListIterator(I):

It is the Child Interface of Iterator
we can perform read, write, update and remove operation using ListIterator

We can move forward direction or backward direction using ListIterator

How to create an Object of ListIterator
ListIterator ltr=list.listIterator();
Note: We can use ListIerator only for List Implemented Class
Methods
1. boolean hasNext()
2. Object next();
3. boolean hasPrevious();
4. Object previous()
5. add()
6. set()
7. remove()

```
/*
 * 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 dsafeb2025;
```

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.ListIterator;

/**
 *
 * @author Admin
 */
public class A1 {
    public static void main(String[] args) {

        ArrayList list=new ArrayList();
        list.add(10);
        list.add("hello");
        list.add('A');
        list.add(true);
        list.add(123.45f);
        list.add(45.55555);
         list.add(10);

        System.out.println("Array List : "+list);
        System.out.println("Print Data of Array List Using for  loop ");
        for(int i=0;i<list.size();i++){
            System.out.print("==>"+list.get(i));
        }

        System.out.println("\nPrint Data of Array List Using for each ");
        for(Object x:list){
            System.out.print("==>"+x);
        }

        System.out.println("\nPrint Data of  ArrayList Using Iterator ");
        Iterator itr=list.iterator();
        while(itr.hasNext()){
      Object x=itr.next();
      if(x.equals(10)){
          itr.remove();
      }
            System.out.print("====>"+x);
        }
        System.out.println("\n-----------------------------\n");
        System.out.println(""+list);
        System.out.println("\nPrint Data of ArrayList In forward Direction using ListIterator");
        ListIterator ltr=list.listIterator();
        while(ltr.hasNext()){
            System.out.print("==>"+ltr.next());
        }
        System.out.println("\nPrint Data of ArrayList In Back Direction using ListIterator");
      while(ltr.hasPrevious()){
            System.out.print("==>"+ltr.previous());
        }
    }


}
```