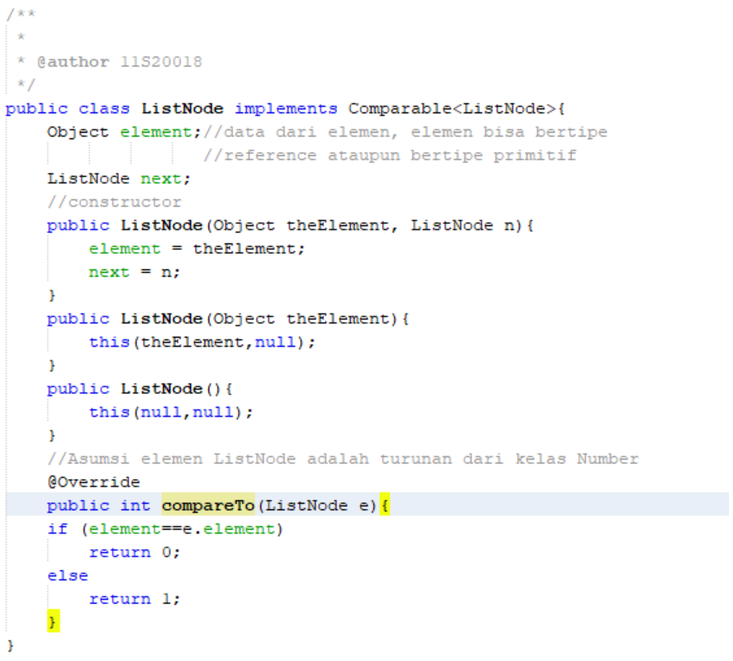
11S20018

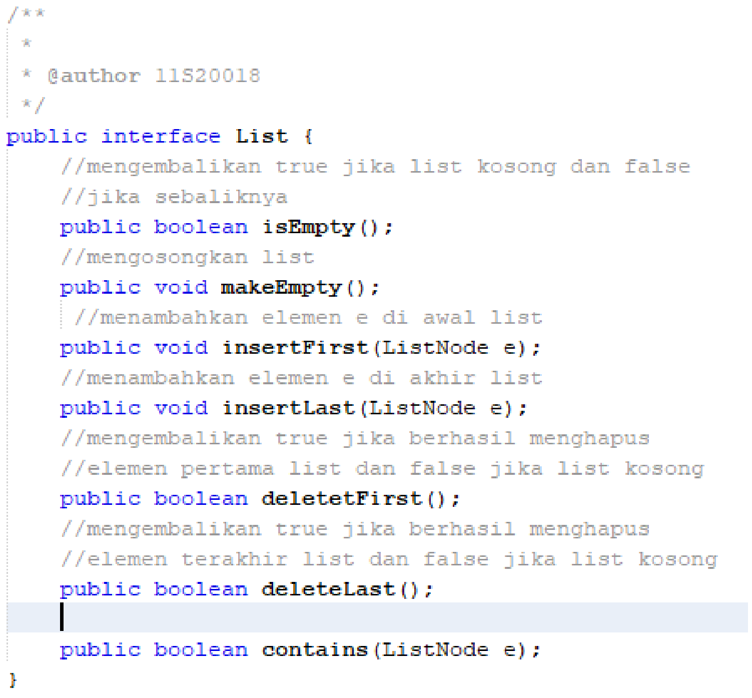
Ruth Aulya Silalahi

1. LinkedList, ListNode, List
   a) ListNode



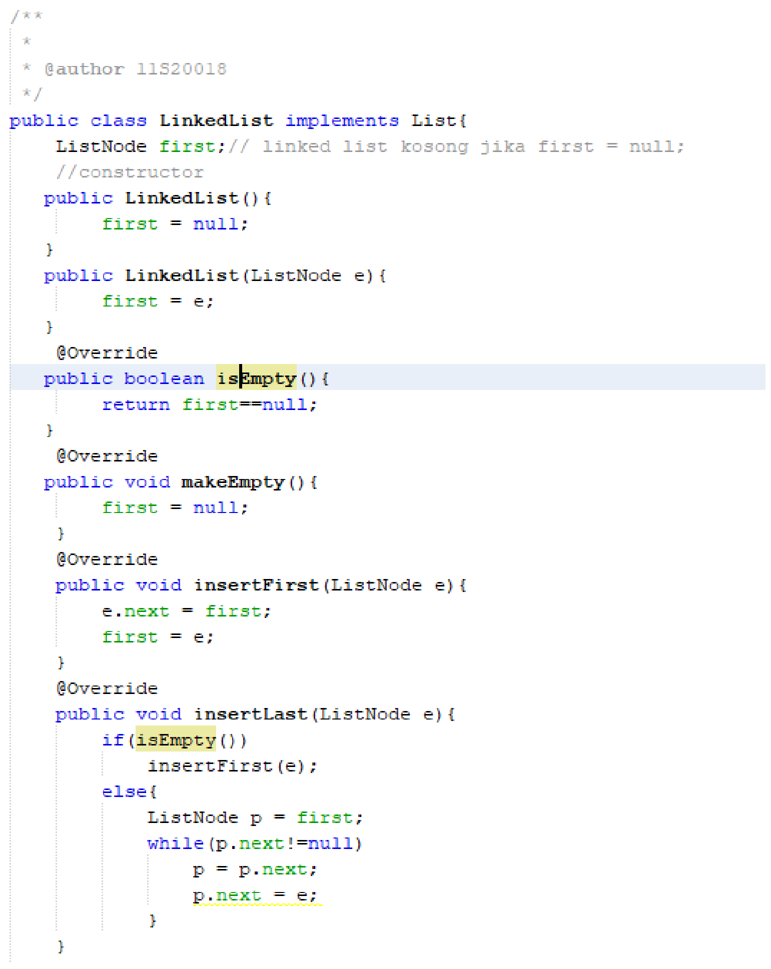
```
/**
 *
 * @author 11S20018
 */
public class ListNode implements Comparable<ListNode>{
    Object element;//data dari elemen, elemen bisa bertipe
                   //reference ataupun bertipe primitif
    ListNode next;
    //constructor
    public ListNode(Object theElement, ListNode n){
        element = theElement;
        next = n;
    }
    public ListNode(Object theElement){
        this(theElement,null);
    }
    public ListNode(){
        this(null,null);
    }
    //Asumsi elemen ListNode adalah turunan dari kelas Number
    @Override
    public int compareTo(ListNode e){
    if (element==e.element)
        return 0;
    else
        return 1;
    }
}
```

   b) List

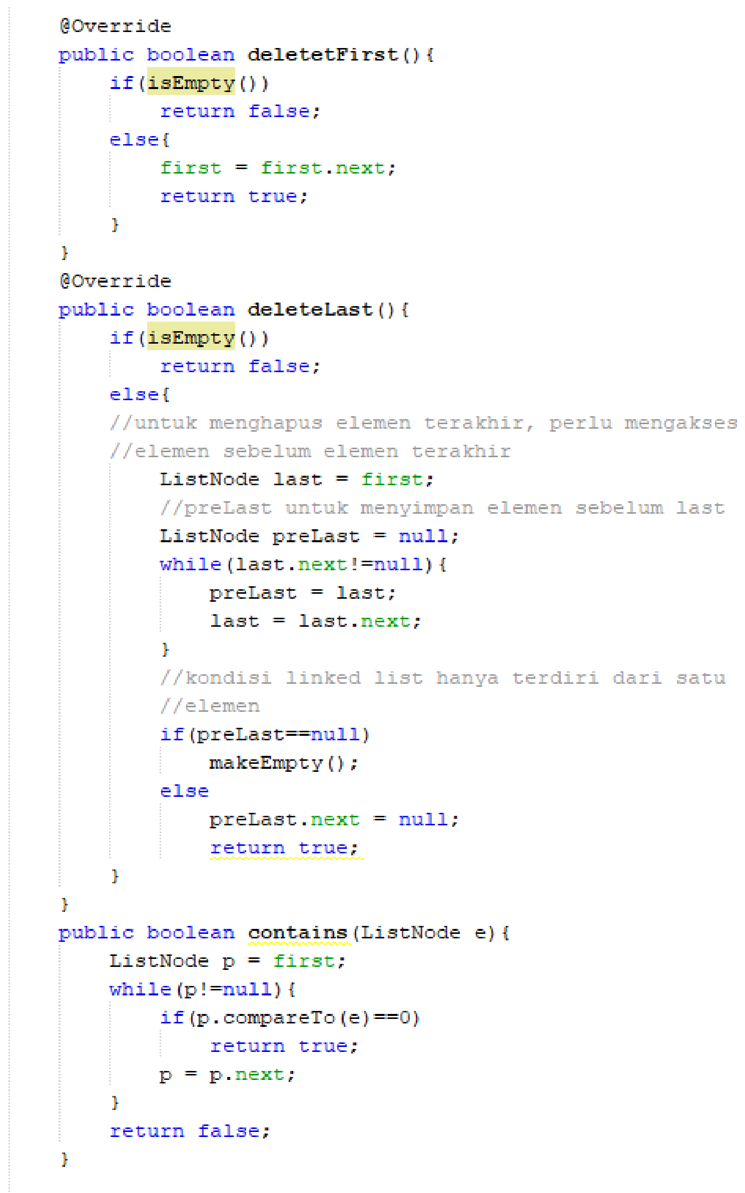


```
/**
 *
 * @author 11S20018
 */
public interface List {
    //mengembalikan true jika list kosong dan false
    //jika sebaliknya
    public boolean isEmpty();
    //mengosongkan list
    public void makeEmpty();
     //menambahkan elemen e di awal list
    public void insertFirst(ListNode e);
    //menambahkan elemen e di akhir list
    public void insertLast(ListNode e);
    //mengembalikan true jika berhasil menghapus
    //elemen pertama list dan false jika list kosong
    public boolean deletetFirst();
    //mengembalikan true jika berhasil menghapus
    //elemen terakhir list dan false jika list kosong
    public boolean deleteLast();

    public boolean contains(ListNode e);
}
```

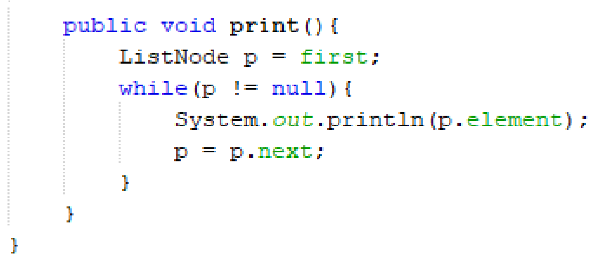
c) LinkedList



```java
/**
 *
 * @author 11S20018
 */
public class LinkedList implements List{
    ListNode first;// linked list kosong jika first = null;
    //constructor
   public LinkedList(){
        first = null;
   }
   public LinkedList(ListNode e){
        first = e;
   }
    @Override
   public boolean isEmpty(){
        return first==null;
   }
    @Override
   public void makeEmpty(){
        first = null;
    }
    @Override
    public void insertFirst(ListNode e){
        e.next = first;
        first = e;
    }
    @Override
    public void insertLast(ListNode e){
        if(isEmpty())
            insertFirst(e);
        else{
            ListNode p = first;
            while(p.next!=null)
                p = p.next;
                p.next = e;
            }
    }
```

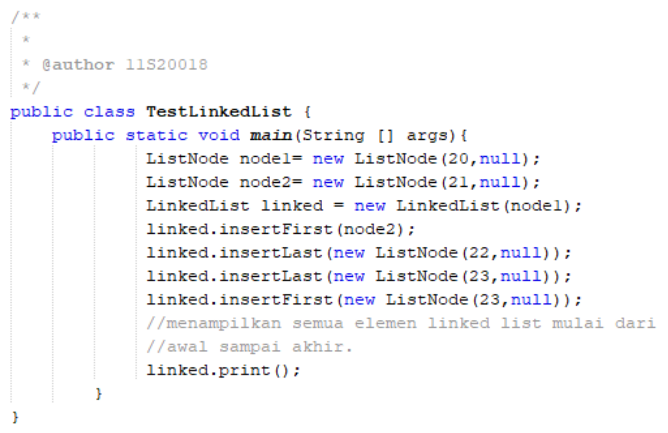


```java
    @Override
    public boolean deletetFirst(){
        if(isEmpty())
            return false;
        else{
            first = first.next;
            return true;
        }
    }
    @Override
    public boolean deleteLast(){
        if(isEmpty())
            return false;
        else{
        //untuk menghapus elemen terakhir, perlu mengakses
        //elemen sebelum elemen terakhir
            ListNode last = first;
            //preLast untuk menyimpan elemen sebelum last
            ListNode preLast = null;
            while(last.next!=null){
                preLast = last;
                last = last.next;
            }
            //kondisi linked list hanya terdiri dari satu
            //elemen
            if(preLast==null)
                makeEmpty();
            else
                preLast.next = null;
                return true;
        }
    }
    public boolean contains(ListNode e){
        ListNode p = first;
        while(p!=null){
            if(p.compareTo(e)==0)
                return true;
            p = p.next;
        }
        return false;
    }
```



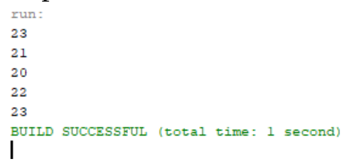
```java
    public void print(){
        ListNode p = first;
        while(p != null){
            System.out.println(p.element);
            p = p.next;
        }
    }
}
```

2. TestLinked1
   Code:



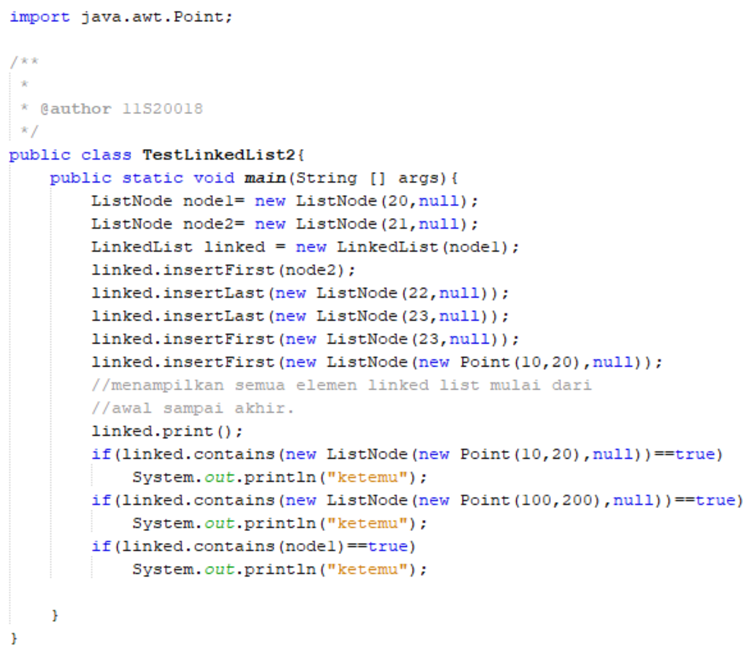
```
/**
 *
 * @author 11S20018
 */
public class TestLinkedList {
    public static void main(String [] args){
            ListNode node1= new ListNode(20,null);
            ListNode node2= new ListNode(21,null);
            LinkedList linked = new LinkedList(node1);
            linked.insertFirst(node2);
            linked.insertLast(new ListNode(22,null));
            linked.insertLast(new ListNode(23,null));
            linked.insertFirst(new ListNode(23,null));
            //menampilkan semua elemen linked list mulai dari
            //awal sampai akhir.
            linked.print();
        }
}
```

   Output:



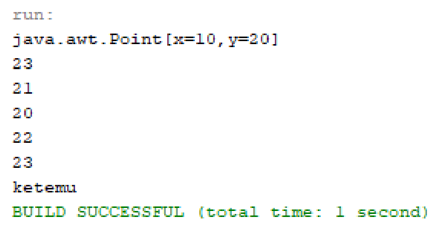
```
run:
23
21
20
22
23
BUILD SUCCESSFUL (total time: 1 second)
```

3. TestLinked2
   Code:



```
import java.awt.Point;

/**
 *
 * @author 11S20018
 */
public class TestLinkedList2{
    public static void main(String [] args){
        ListNode node1= new ListNode(20,null);
        ListNode node2= new ListNode(21,null);
        LinkedList linked = new LinkedList(node1);
        linked.insertFirst(node2);
        linked.insertLast(new ListNode(22,null));
        linked.insertLast(new ListNode(23,null));
        linked.insertFirst(new ListNode(23,null));
        linked.insertFirst(new ListNode(new Point(10,20),null));
        //menampilkan semua elemen linked list mulai dari
        //awal sampai akhir.
        linked.print();
        if(linked.contains(new ListNode(new Point(10,20),null))==true)
            System.out.println("ketemu");
        if(linked.contains(new ListNode(new Point(100,200),null))==true)
            System.out.println("ketemu");
        if(linked.contains(node1)==true)
            System.out.println("ketemu");

    }
}
```

   Output:



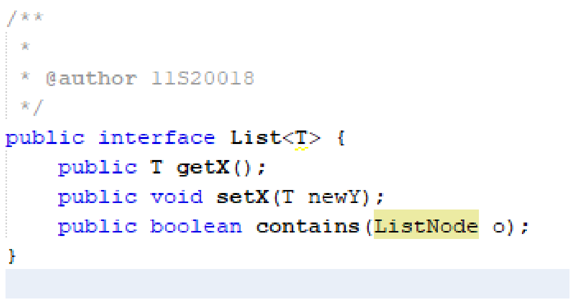
```
run:
java.awt.Point[x=10,y=20]
23
21
20
22
23
ketemu
BUILD SUCCESSFUL (total time: 1 second)
```

4. TestLinked2
   a) Ada berapa jenis elemen yang disimpan di dalam Linked List linked?
      Jawab: Terdapat dua tipe yaitu int dan point
   b) Periksa apakah setiap pemanggilan method contains() selalu memberikan hasil yang benar?

      Jawab: Ya karena mencetak ”ketemu”
   c) Tuliskan hasil analisis anda terkait jawaban pada butir 4b.

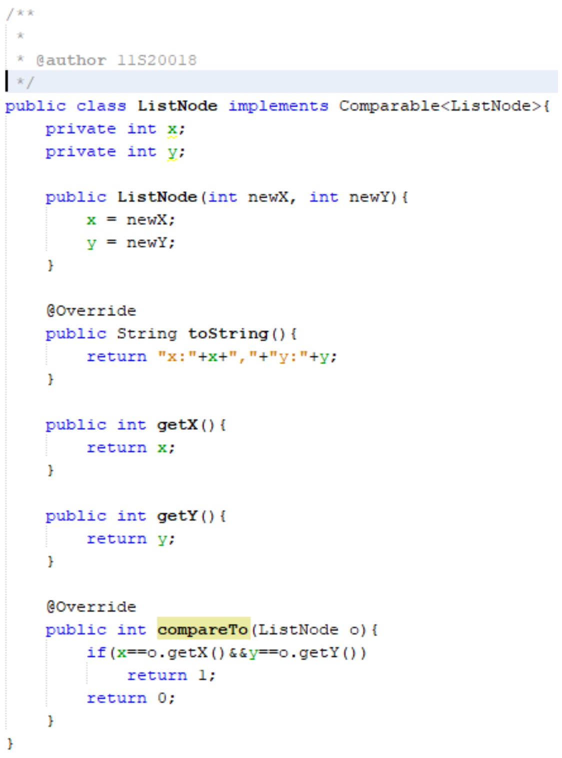
      Jawab: Pada perbandingan yang dibandingan bukan value tetapi alamat

5. Tugas anda adalah; silahkan anda merancang ulang kelas List, ListNode, dan LinkedList dengan menggunakan *generics* sehingga kasus pada butir 4 dapat dihindarkan
   Jawab:
   a) List



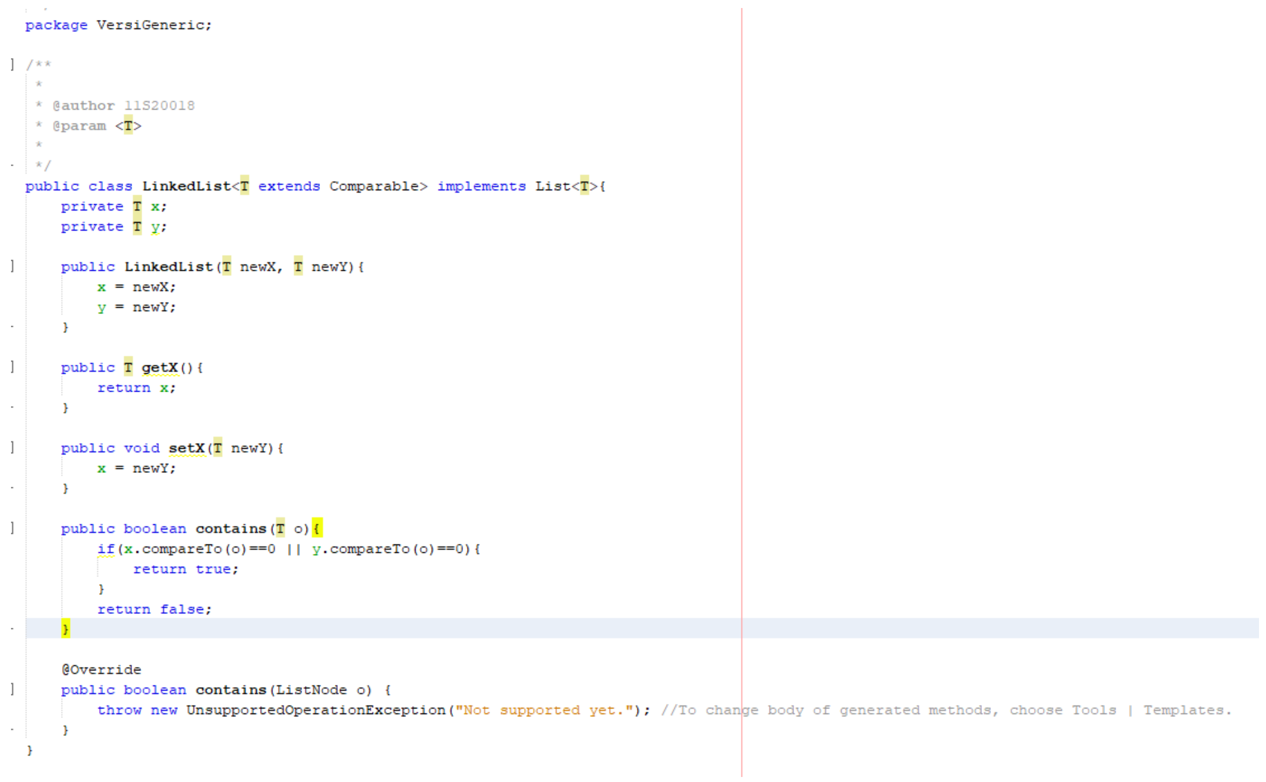
```
/**
 *
 * @author 11S20018
 */
public interface List<T> {
    public T getX();
    public void setX(T newY);
    public boolean contains(ListNode o);
}
```

   b) ListNode



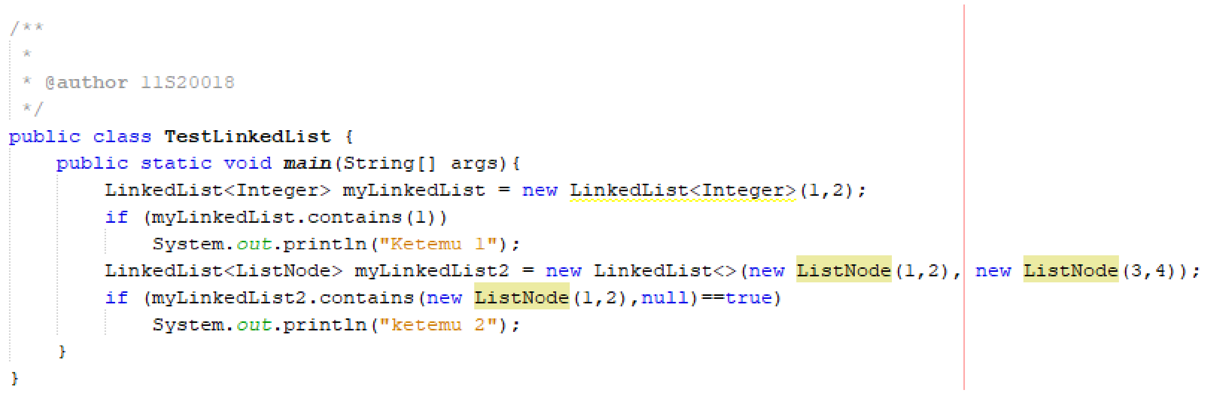
```
/**
 *
 * @author 11S20018
 */
public class ListNode implements Comparable<ListNode>{
    private int x;
    private int y;

    public ListNode(int newX, int newY){
        x = newX;
        y = newY;
    }

    @Override
    public String toString(){
        return "x:"+x+","+"y:"+y;
    }

    public int getX(){
        return x;
    }

    public int getY(){
        return y;
    }

    @Override
    public int compareTo(ListNode o){
        if(x==o.getX()&&y==o.getY())
            return 1;
        return 0;
    }
}
```

c) LinkedList



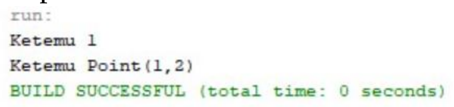
```
package VersiGeneric;

/**
 *
 * @author 11S20018
 * @param <T>
 *
 */
public class LinkedList<T extends Comparable> implements List<T>{
    private T x;
    private T y;

    public LinkedList(T newX, T newY){
        x = newX;
        y = newY;
    }

    public T getX(){
        return x;
    }

    public void setX(T newY){
        x = newY;
    }

    public boolean contains(T o){
        if(x.compareTo(o)==0 || y.compareTo(o)==0){
            return true;
        }
        return false;
    }

    @Override
    public boolean contains(ListNode o) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }
}
```

d) TestLinkedList



```
/**
 *
 * @author 11S20018
 */
public class TestLinkedList {
    public static void main(String[] args){
        LinkedList<Integer> myLinkedList = new LinkedList<Integer>(1,2);
        if (myLinkedList.contains(1))
            System.out.println("Ketemu 1");
        LinkedList<ListNode> myLinkedList2 = new LinkedList<>(new ListNode(1,2), new ListNode(3,4));
        if (myLinkedList2.contains(new ListNode(1,2),null)==true)
            System.out.println("ketemu 2");
    }
}
```

Output:



```
run:
Ketemu 1
Ketemu Point(1,2)
BUILD SUCCESSFUL (total time: 0 seconds)
```