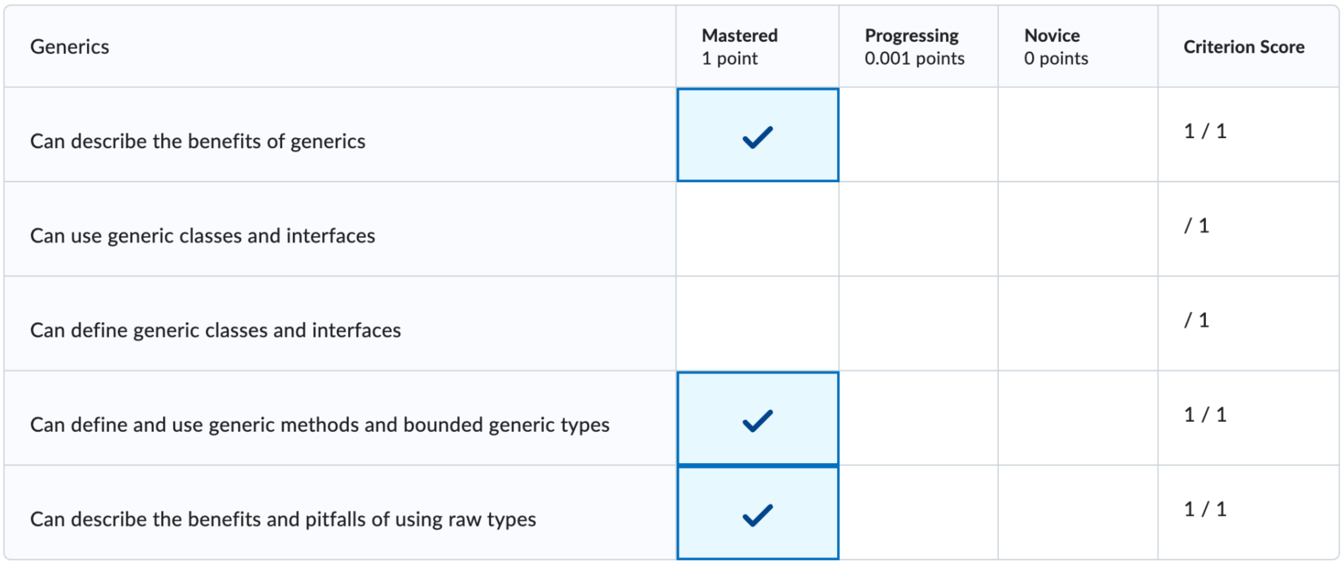
Generics Competency #5

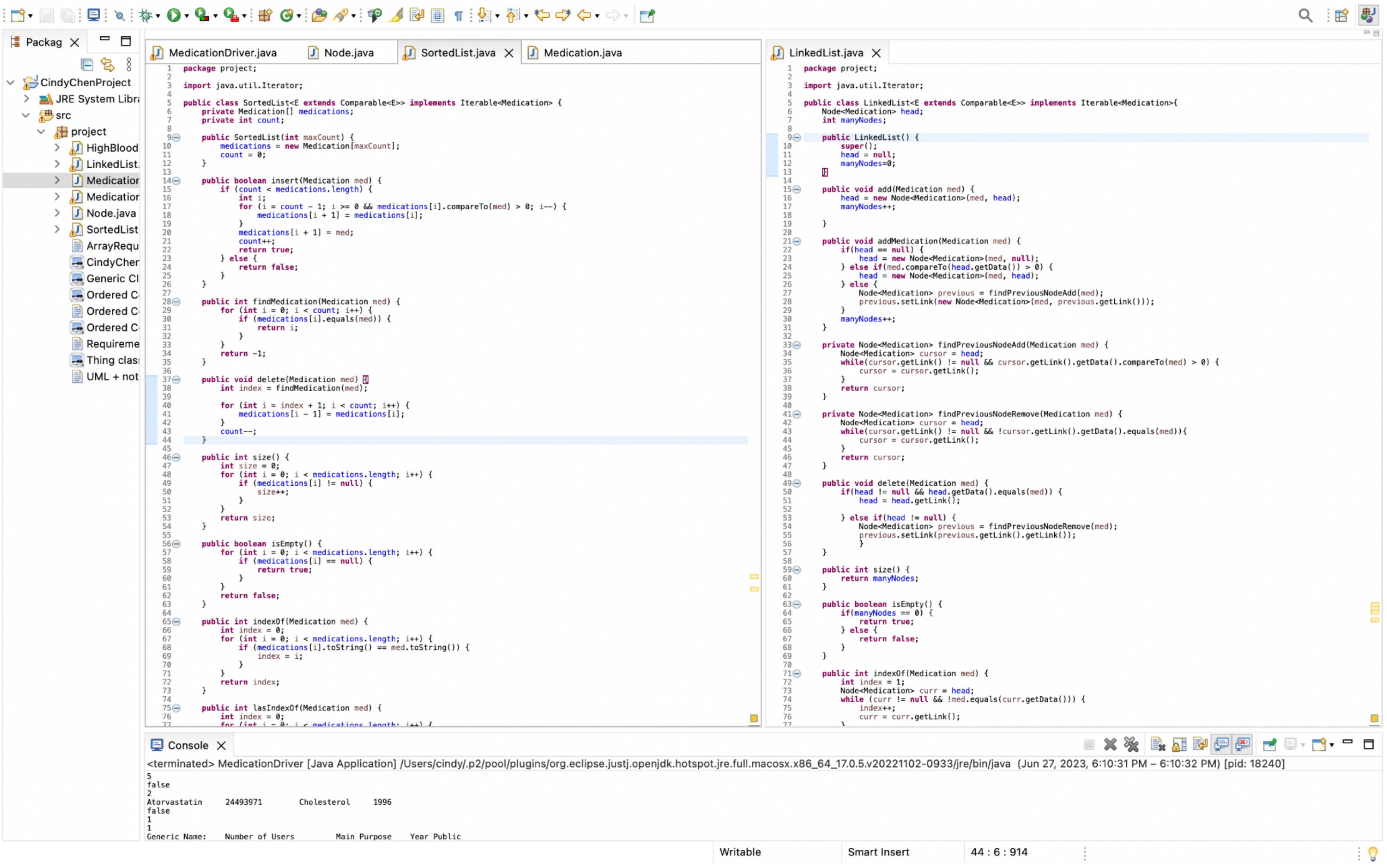
Can use generic classes and interfaces
Can define generic classes and interfaces

Generic classes: classes that are programmed to handle any type of datatype objects, not attached to a specific particular data type, flexible for all data types. The difference between generic and non-generic classes is that generic classes can be written with any datatype, and write methods/classes/interfaces once and use it everywhere. But for non-generics, the programmer would need to write/adjust the code specifically to that particular data type.

Generic interfaces: interfaces that deal with generic data types, like generic classes, they are not holded down by a particular data type. Generic interface have method headers but without method functions, allowing the interface to be implemented many times, and optionally adjusted to the programmer's needs. The generic interface is defined using the generic type parameter, when its implemented, the type argument needs to be stated.



Generics	**Mastered** 1 point	**Progressing** 0.001 points	**Novice** 0 points	**Criterion Score**
Can describe the benefits of generics	✓			1 / 1
Can use generic classes and interfaces				/ 1
Can define generic classes and interfaces				/ 1
Can define and use generic methods and bounded generic types	✓			1 / 1
Can describe the benefits and pitfalls of using raw types	✓			1 / 1



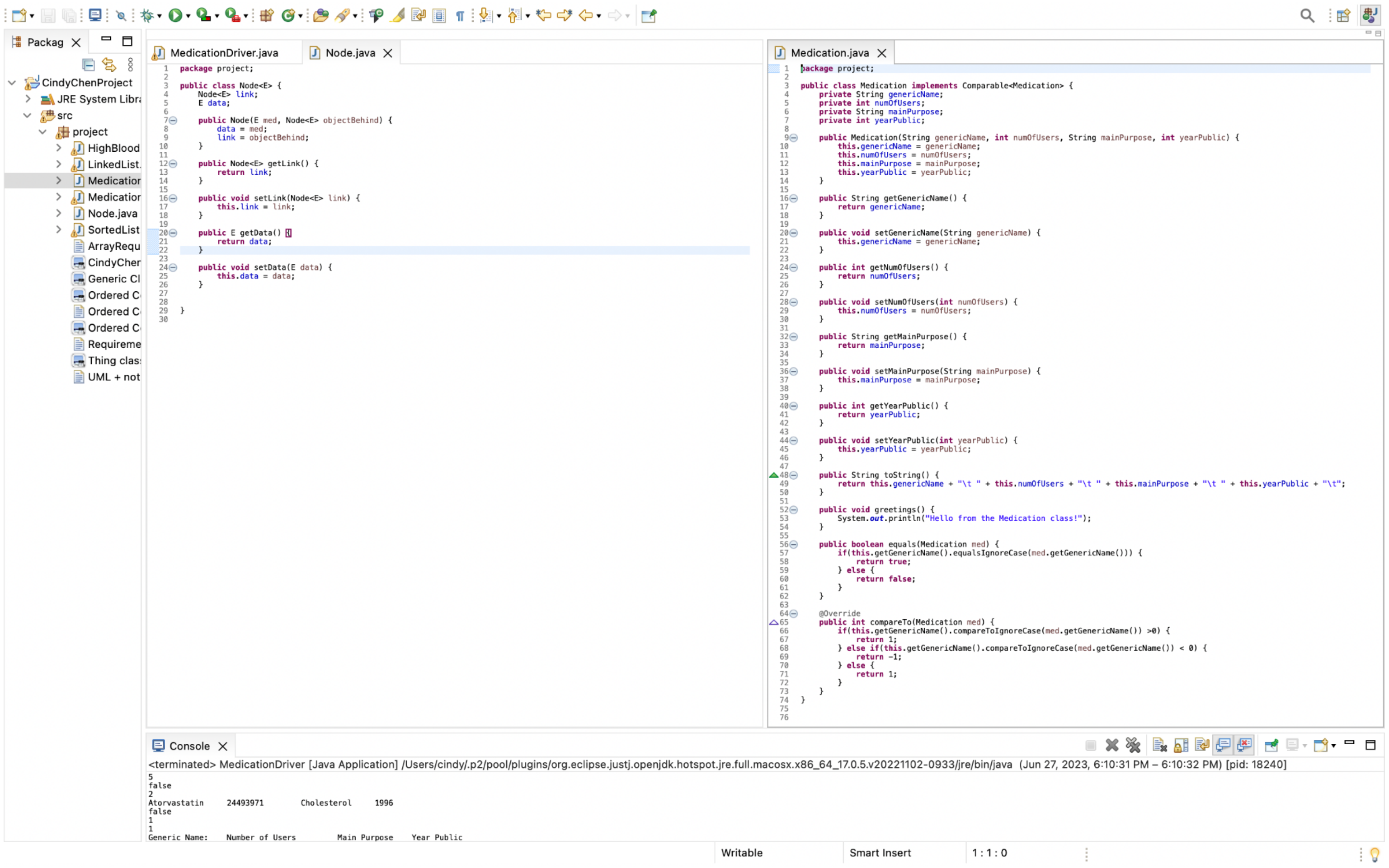
SortedList.java

```java
package project;

import java.util.Iterator;

public class SortedList<E extends Comparable<E>> implements Iterable<Medication> {
    private Medication[] medications;
    private int count;

    public SortedList(int maxCount) {
        medications = new Medication[maxCount];
        count = 0;
    }

    public boolean insert(Medication med) {
        if (count < medications.length) {
            int i;
            for (i = count - 1; i >= 0 && medications[i].compareTo(med) > 0; i--) {
                medications[i + 1] = medications[i];
            }
            medications[i + 1] = med;
            count++;
            return true;
        } else {
            return false;
        }
    }

    public int findMedication(Medication med) {
        for (int i = 0; i < count; i++) {
            if (medications[i].equals(med)) {
                return i;
            }
        }
        return -1;
    }

    public void delete(Medication med) {
        int index = findMedication(med);

        for (int i = index + 1; i < count; i++) {
            medications[i - 1] = medications[i];
        }
        count--;
    }

    public int size() {
        int size = 0;
        for (int i = 0; i < medications.length; i++) {
            if (medications[i] != null) {
                size++;
            }
        }
        return size;
    }

    public boolean isEmpty() {
        for (int i = 0; i < medications.length; i++) {
            if (medications[i] == null) {
                return true;
            }
        }
        return false;
    }

    public int indexOf(Medication med) {
        int index = 0;
        for (int i = 0; i < medications.length; i++) {
            if (medications[i].toString() == med.toString()) {
                index = i;
            }
        }
        return index;
    }

    public int lasIndexOf(Medication med) {
        int index = 0;
        for (int i = 0; i < medications.length; i++) {
```

LinkedList.java

```java
package project;

import java.util.Iterator;

public class LinkedList<E extends Comparable<E>> implements Iterable<Medication>{
    Node<Medication> head;
    int manyNodes;

    public LinkedList() {
        super();
        head = null;
        manyNodes=0;
    }

    public void add(Medication med) {
        head = new Node<Medication>(med, head);
        manyNodes++;

    }

    public void addMedication(Medication med) {
        if(head == null) {
            head = new Node<Medication>(med, null);
        } else if(med.compareTo(head.getData()) > 0) {
            head = new Node<Medication>(med, head);
        } else {
           Node<Medication> previous = findPreviousNodeAdd(med);
           previous.setLink(new Node<Medication>(med, previous.getLink()));
        }
        manyNodes++;
    }

    private Node<Medication> findPreviousNodeAdd(Medication med) {
        Node<Medication> cursor = head;
        while(cursor.getLink() != null && cursor.getLink().getData().compareTo(med) > 0) {
            cursor = cursor.getLink();
        }
        return cursor;
    }

    private Node<Medication> findPreviousNodeRemove(Medication med) {
        Node<Medication> cursor = head;
        while(cursor.getLink() != null && !cursor.getLink().getData().equals(med)){
            cursor = cursor.getLink();
        }
        return cursor;
    }

    public void delete(Medication med) {
        if(head != null && head.getData().equals(med)) {
            head = head.getLink();

        } else if(head != null) {
           Node<Medication> previous = findPreviousNodeRemove(med);
           previous.setLink(previous.getLink().getLink());
           }
    }

    public int size() {
        return manyNodes;
    }

    public boolean isEmpty() {
        if(manyNodes == 0) {
            return true;
        } else {
            return false;
        }
    }

    public int indexOf(Medication med) {
        int index = 1;
        Node<Medication> curr = head;
        while (curr != null && !med.equals(curr.getData())) {
            index++;
            curr = curr.getLink();
        }
```

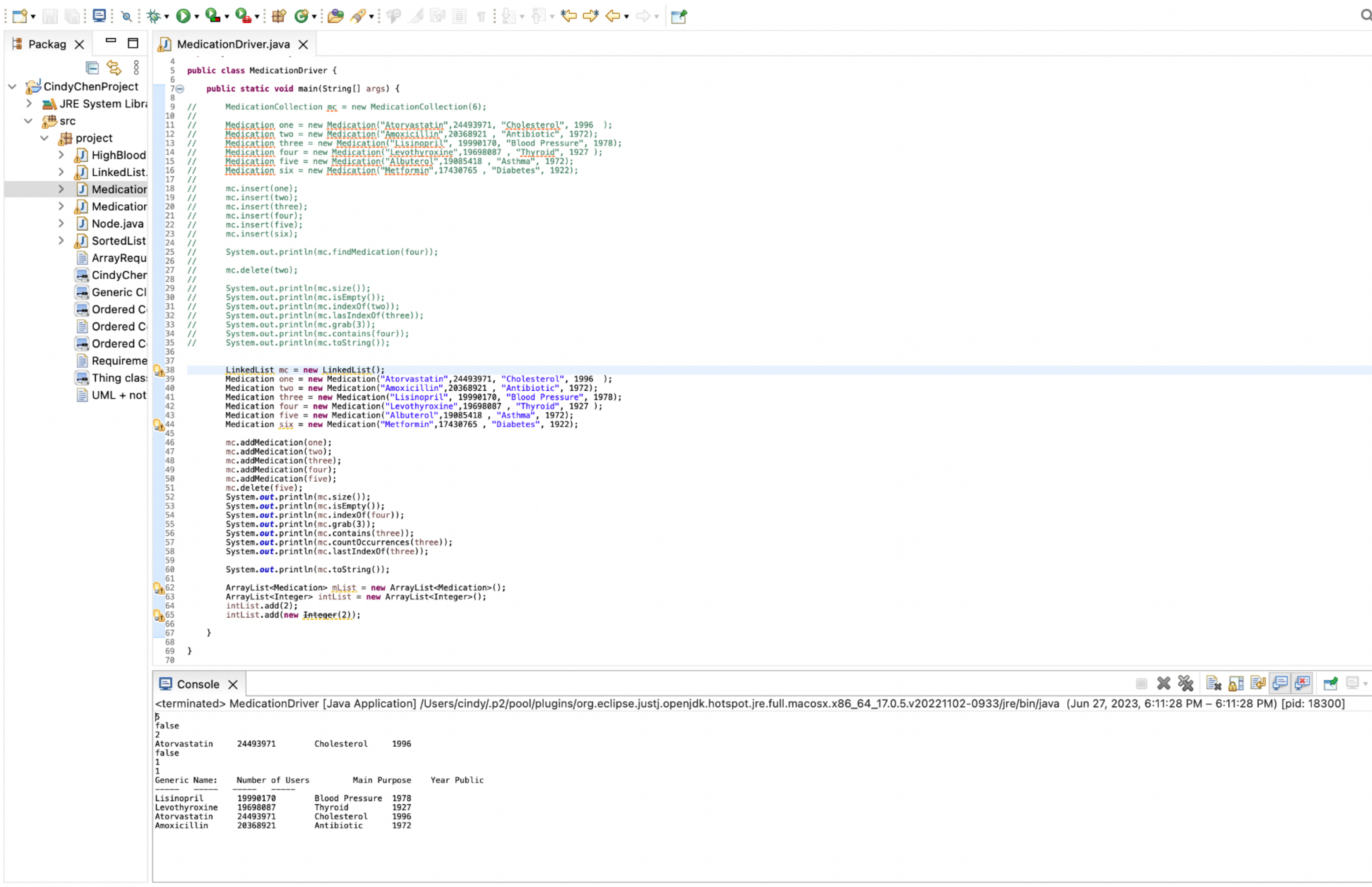
```
<terminated> MedicationDriver [Java Application] /Users/cindy/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.5.v20221102-0933/jre/bin/java  (Jun 27, 2023, 6:10:31 PM – 6:10:32 PM) [pid: 18240]
5
false
2
Atorvastatin     24493971        Cholesterol     1996
false
1
1
Generic Name:    Number of Users         Main Purpose    Year Public
```



```java
package project;

public class Node<E> {
    Node<E> link;
    E data;

    public Node(E med, Node<E> objectBehind) {
        data = med;
        link = objectBehind;
    }

    public Node<E> getLink() {
        return link;
    }

    public void setLink(Node<E> link) {
        this.link = link;
    }

    public E getData() {
        return data;
    }

    public void setData(E data) {
        this.data = data;
    }


}
```

```java
package project;

public class Medication implements Comparable<Medication> {
    private String genericName;
    private int numOfUsers;
    private String mainPurpose;
    private int yearPublic;

    public Medication(String genericName, int numOfUsers, String mainPurpose, int yearPublic) {
        this.genericName = genericName;
        this.numOfUsers = numOfUsers;
        this.mainPurpose = mainPurpose;
        this.yearPublic = yearPublic;
    }

    public String getGenericName() {
        return genericName;
    }

    public void setGenericName(String genericName) {
        this.genericName = genericName;
    }

    public int getNumOfUsers() {
        return numOfUsers;
    }

    public void setNumOfUsers(int numOfUsers) {
        this.numOfUsers = numOfUsers;
    }

    public String getMainPurpose() {
        return mainPurpose;
    }

    public void setMainPurpose(String mainPurpose) {
        this.mainPurpose = mainPurpose;
    }

    public int getYearPublic() {
        return yearPublic;
    }

    public void setYearPublic(int yearPublic) {
        this.yearPublic = yearPublic;
    }

    public String toString() {
        return this.genericName + "\t " + this.numOfUsers + "\t " + this.mainPurpose + "\t " + this.yearPublic + "\t";
    }

    public void greetings() {
        System.out.println("Hello from the Medication class!");
    }

    public boolean equals(Medication med) {
        if(this.getGenericName().equalsIgnoreCase(med.getGenericName())) {
            return true;
        } else {
            return false;
        }
    }

    @Override
    public int compareTo(Medication med) {
        if(this.getGenericName().compareToIgnoreCase(med.getGenericName()) >0) {
            return 1;
        } else if(this.getGenericName().compareToIgnoreCase(med.getGenericName()) < 0) {
            return -1;
        } else {
            return 1;
        }
    }
}


```

```
<terminated> MedicationDriver [Java Application] /Users/cindy/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.5.v20221102-0933/jre/bin/java  (Jun 27, 2023, 6:10:31 PM – 6:10:32 PM) [pid: 18240]
5
false
2
Atorvastatin     24493971        Cholesterol     1996
false
1
1
Generic Name:    Number of Users         Main Purpose    Year Public
```



```java
public class MedicationDriver {

    public static void main(String[] args) {

//        MedicationCollection mc = new MedicationCollection(6);
//
//        Medication one = new Medication("Atorvastatin",24493971, "Cholesterol", 1996  );
//        Medication two = new Medication("Amoxicillin",20368921 , "Antibiotic", 1972);
//        Medication three = new Medication("Lisinopril", 19990170, "Blood Pressure", 1978);
//        Medication four = new Medication("Levothyroxine",19698087 , "Thyroid", 1927 );
//        Medication five = new Medication("Albuterol",19085418 , "Asthma", 1972);
//        Medication six = new Medication("Metformin",17430765 , "Diabetes", 1922);
//
//        mc.insert(one);
//        mc.insert(two);
//        mc.insert(three);
//        mc.insert(four);
//        mc.insert(five);
//        mc.insert(six);
//
//        System.out.println(mc.findMedication(four));
//
//        mc.delete(two);
//
//        System.out.println(mc.size());
//        System.out.println(mc.isEmpty());
//        System.out.println(mc.indexOf(two));
//        System.out.println(mc.lasIndexOf(three));
//        System.out.println(mc.grab(3));
//        System.out.println(mc.contains(four));
//        System.out.println(mc.toString());


        LinkedList mc = new LinkedList();
        Medication one = new Medication("Atorvastatin",24493971, "Cholesterol", 1996  );
        Medication two = new Medication("Amoxicillin",20368921 , "Antibiotic", 1972);
        Medication three = new Medication("Lisinopril", 19990170, "Blood Pressure", 1978);
        Medication four = new Medication("Levothyroxine",19698087 , "Thyroid", 1927 );
        Medication five = new Medication("Albuterol",19085418 , "Asthma", 1972);
        Medication six = new Medication("Metformin",17430765 , "Diabetes", 1922);

        mc.addMedication(one);
        mc.addMedication(two);
        mc.addMedication(three);
        mc.addMedication(four);
        mc.addMedication(five);
        mc.delete(five);
        System.out.println(mc.size());
        System.out.println(mc.isEmpty());
        System.out.println(mc.indexOf(four));
        System.out.println(mc.grab(3));
        System.out.println(mc.contains(three));
        System.out.println(mc.countOccurrences(three));
        System.out.println(mc.lastIndexOf(three));

        System.out.println(mc.toString());

        ArrayList<Medication> mList = new ArrayList<Medication>();
        ArrayList<Integer> intList = new ArrayList<Integer>();
        intList.add(2);
        intList.add(new Integer(2));

    }

}
```

```
<terminated> MedicationDriver [Java Application] /Users/cindy/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.5.v20221102-0933/jre/bin/java  (Jun 27, 2023, 6:11:28 PM – 6:11:28 PM) [pid: 18300]
5
false
2
Atorvastatin     24493971        Cholesterol     1996
false
1
1
Generic Name:    Number of Users         Main Purpose    Year Public
-----   -----   -----   -----
Lisinopril       19990170        Blood Pressure  1978
Levothyroxine    19698087        Thyroid         1927
Atorvastatin     24493971        Cholesterol     1996
Amoxicillin      20368921        Antibiotic      1972
```