Implement your own Stack, Queue Class and implement add, delete, update and get method.

```
package com.company;
import java.util.Stack;

class Main
{
    public static void main (String[] args)
    {
        Stack<String> stack = new Stack<String>();
        stack.push("abc");
        stack.push("pqr");
        stack.push("gun");
        stack.push("tim");
        System.out.println(stack);
        System.out.println(stack.pop());
        System.out.println(stack);
        System.out.println(stack.peek());
        System.out.println(stack.search("pqr"));
        System.out.println(stack.search("cds"));//-1
        System.out.println(stack.empty());//false
    }
}
```

```
package com.company;
import java.util.*;

class Main
{
    public static void main (String[] args) {
        Queue<String> q = new PriorityQueue<>();

        q.add("abc");
        q.add("pqr");
        q.add("cds");

        System.out.println(q);

        q.remove("pqr");

        System.out.println("After Remove " + q);
        System.out.println( q.peek());
        System.out.println("Poll Method " + q.poll());



        System.out.println(q);
    }
}
```

Implement your own LinkedList or ArrayList Class and implement add, delete, update and get method.

```
package com.company;
import java.util.*;
class Main
{
    public static void main (String[] args) {
        LinkedList<String> l=new LinkedList<>();
        l.add("ashok");
        l.add("abc");
        l.add(null);
        l.add("ashok");
        System.out.println(l);
        l.set(0,"software");
        System.out.println(l);

        l.set(0,"ven");
        System.out.println(l);
        l.removeLast();
        System.out.println(l);
        l.addFirst("vvv");
        System.out.println(l);

        for (String s : l) {
            System.out.println(s);
        }
    }
}
```

Given custom Employee Object with age, name, id, department attributes . Write code

to retrieve employee object sorted by id descending

to retrieve fetch employee object sorted by name ascending

to retrieve fetch employee object sorted by age descending

```
package com.company;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Iterator;
import java.util.List;


class Employee {

    String Name;
```

```java
int Age;
public Employee(String Name, Integer Age)
{
   this.Name = Name;
   this.Age = Age;
}

public String getName() { return Name; }

public Integer getAge() { return Age; }


@Override public String toString()
{
   return "Customer{"
         + "Name=" + Name + ", Age=" + Age + '}';
}
static class CustomerSortingComparator
      implements Comparator<Student> {



   @Override
   public int compare(Student customer1,
                  Student customer2)
   {



      int NameCompare = customer1.getName().compareTo(
            customer2.getName());

      int AgeCompare = customer1.getAge().compareTo(
            customer2.getAge());

      return (NameCompare == 0) ? AgeCompare
            : NameCompare;
   }
}


public static void main(String[] args)
{

   List<Student> al = new ArrayList<>();

   Student obj1 = new Student("Ajay", 27);
   Student obj2 = new Student("Sneha", 23);
   Student obj3 = new Student("Simran", 37);
   Student obj4 = new Student("Ajay", 22);
   Student obj5 = new Student("Ajay", 29);
   Student obj6 = new Student("Sneha", 22);


   al.add(obj1);
   al.add(obj2);
   al.add(obj3);
   al.add(obj4);
   al.add(obj5);
   al.add(obj6);
```

```
        Iterator<Student> custIterator = al.iterator();
        System.out.println("Before Sorting:\n");

        while (custIterator.hasNext()) {

            System.out.println(custIterator.next());
        }

        al.sort(new CustomerSortingComparator());


        System.out.println("\n\nAfter Sorting:\n");


        for (Student customer : al) {
            System.out.println(customer);
        }
    }
}
```

Given following Arraylist and find common elements in both array and share time complexity of the solution
ArrayList<Integer> list1 = [2,4,1,56,3]
ArrayList<Integer> list2 = [56,8,2,4,3]

```
package com.company;
import java.util.*;

public class list4 {
        public static void main(String[] args)
        {


            int[] ids = {2,4,1,56,3};
            ArrayList<Integer> list1 = new ArrayList<>();

            for (int id: ids) {
                list1.add(id);
            }

            System.out.println("List1: "
                    + list1);


            int[] ids2 = {56,8,2,4,3};
            ArrayList<Integer>
                    list2 = new ArrayList<>();

            for (int id: ids2) {
                list2.add(id);
            }
            System.out.println("List2: "
                    + list2);


            list1.retainAll(list2);
            System.out.println("Common elements: "
                    + list1);
        }
    }
```

Implement Least Recent Used (LRU) Cache using Java collections

```
package com.company;
import java.util.*;
public class lru {

    Set<Integer> cache;
    int capacity;

    public lru(int capacity)
    {
        this.cache = new LinkedHashSet<>(capacity);
        this.capacity = capacity;
    }

    public boolean get(int key)
    {
        if (!cache.contains(key))
            return false;
        cache.remove(key);
        cache.add(key);
        return true;
    }



    public void refer(int key)
    {
        if (!get(key))
            put(key);
    }

    public void display()
    {
        LinkedList<Integer> list = new LinkedList<>(cache);

        Iterator<Integer> itr = list.descendingIterator();

        while (itr.hasNext())
            System.out.print(itr.next() + " ");
    }
```

```java
    public void put(int key)
    {

        if (cache.size() == capacity) {
            int firstKey = cache.iterator().next();
            cache.remove(firstKey);
        }

        cache.add(key);
    }

    public static void main(String[] args)
    {
        lru ca = new lru(4);
        ca.refer(1);
        ca.refer(2);
        ca.refer(3);
        ca.refer(1);
        ca.refer(4);
        ca.refer(5);
        ca.display();
    }
}
```