DECLARATIONS

```
public class MyArrayList<E>
{
    private int size; // Number of elements in the list
    private E[] data;
    private int MAXELEMENTS = 100;
    /** Create an empty list */
    public MyArrayList() {
        data = (E[])new Object[MAXELEMENTS];// cannot create array of generics
        size = 0; // Number of elements in the list
    }
```

add()

```
  public void add(int index, E e) {
        // Ensure the index is in the right range
        if (index < 0 || index > size)
            throw new IndexOutOfBoundsException
                    ("Index: " + index + ", Size: " + size);
        // Move the elements to the right after the specified index
        for (int i = size - 1; i >= index; i--)
            data[i + 1] = data[i];
        // Insert new element to data[index]
        data[index] = e;
        // Increase size by 1
        size++;
    }
```

contains()

```
    public boolean contains(Object e) {
        for (int i = 0; i < size; i++)
            if (e.equals(data[i])) return true;
        return false;
    }
```

get()

```
    public E get(int index) {
        if (index < 0 || index >= size)
            throw new IndexOutOfBoundsException
                    ("Index: " + index + ", Size: " + size);
        return data[index];
    }
```

remove()

```
        public E remove(int index) {
```

```
        if (index < 0 || index >= size)
            throw new IndexOutOfBoundsException
                    ("Index: " + index + ", Size: " + size);
        E e = data[index];
        // Shift data to the left
        for (int j = index; j < size - 1; j++)
            data[j] = data[j + 1];
        data[size - 1] = null; // This element is now null
        // Decrement size
        size--;
        return e;
    }
```

clear()

```
public void clear()
    {
        size = 0;
    }
```

merge()

```
    public MyArrayList<E> merge(MyArrayList<E> param) {
        int callingCounter = 0;
        int paramCounter = 0;
        int returnCounter = 0;

        MyArrayList<E> returnList = new MyArrayList<>();

        if (this.getSize() == 0)
            return param;

        if (param.getSize() == 0)
            return this;

        if (this.getSize() + param.getSize() >= MAXELEMENTS)
            throw new IndexOutOfBoundsException("Too many elements to merge. Return List
size > " + MAXELEMENTS);

        while (callingCounter < this.getSize() && paramCounter < param.getSize()) {

            if (
((Comparable)this.data[callingCounter]).compareTo(param.data[paramCounter]) < 0 )
            {
                returnList.data[returnCounter] = this.data[callingCounter];
                callingCounter++;
                returnCounter++;
            }
            else
            {
                returnList.data[returnCounter] = param.data[paramCounter];
                paramCounter++;
                returnCounter++;
            }

        }

        if (callingCounter < this.getSize()) {
            for (callingCounter = callingCounter; callingCounter < this.getSize();
callingCounter++) {
                returnList.data[returnCounter] = this.data[callingCounter];
                returnCounter++;
            }
        }

        if (paramCounter < param.getSize()) {
            for (paramCounter = paramCounter; paramCounter < param.getSize();
paramCounter++) {
                returnList.data[returnCounter] = param.data[paramCounter];
                returnCounter++;
            }
        }

        returnList.size = returnCounter;
        return returnList;
    }
```

toString()

```
   public String toString() {
    String result="[";
    for (int i = 0; i < size; i++) {
      result+= data[i];
      if (i < size - 1) result+=", ";
    }
    return result.toString() + "]";
  }

    public String toString() {
    String result="[";
    for (int i = 0; i < size-1; i++)
      result = result + data[i] + ",";
    result = result + data[size-1] +"]";
    return result;
  }
```

getSize()

```
    public int getSize() {
        return size;
    }
```

sort()

```
public boolean sortList() {
   E temp;
   for (int i = 0; i < size-1; i++)
    {
      for (int j = 0; j  <size-1; j++)
       {
        if(((Comparable)data[j]).compareTo(data[j+1])>0)
         {
          temp= data[j+1];
          data[j+1]=data[j];
          data[j]=temp;
         }
      }
    }
    return true;
 }
```

filter()

```
public void filter (E low, E high)

{
    int j=0;
    E[] temp = (E[])new Object[MAXELEMENTS];

    if (getSize()== 0)
        return;
    if (((Comparable)low).compareTo(high)>0)
        return;

    for (int i = 0; i< size; i++)
    {
      if ((((Comparable)data[i]).compareTo(low) >=0) &&
        (((Comparable)data[i]).compareTo(high) <=0))
      {
          temp[j] = data[i];
          j++;
      }
    }
    data = temp;
    size = j;
}
```

Test class for Filter method

```java
public class TestMyArrayList {

   public static void main(String[] args) {
   // Create a list of circles and rectangles
   MyArrayList<Integer> list = new MyArrayList<>();

   System.out.println("TEST WITH EMPTY LIST:"+list);
   list.filter(new Integer(3), new Integer(5));
   System.out.println(list);
   // Add elements to the list
   list.add(0,new Integer(6));
   list.add(1,new Integer(5));
   list.add(2,new Integer(3));
   list.add(3,new Integer(4));
   list.add(4,new Integer(2));
   list.add(5,new Integer(5));
   list.add(6,new Integer(1));


   System.out.println("TEST with low> high");
   list.filter(new Integer(5), new Integer(3)); // test with low > high
   System.out.println(list);
   System.out.println("TEST with low < high");
   list.filter(new Integer(3), new Integer(5));
   System.out.println(list);
 }
```

Test class for merge method

```java
public class TestMyArrayList {
  public static void main(String[] args) {
    // Create a list of circles and rectangles
    MyArrayList<Integer> list1 = new MyArrayList<>();
    MyArrayList<Integer> list2 = new MyArrayList<>();
    MyArrayList<Integer> list3 = new MyArrayList<>();

    System.out.println("\nTEST 1: Both lists empty");
    list3=list1.merge(list2);
    System.out.println("list1 = " + list1);
    System.out.println("list2 = " + list2);
    System.out.println("list3 = " + list3);

    System.out.println("\nTEST 2: Param list empty");
    list1.add(0,new Integer(3));
    list1.add(1,new Integer(8));
    list1.add(2,new Integer(17));
    list3=list1.merge(list2);
    System.out.println("list1 = " + list1);
    System.out.println("list2 = " + list2);
    System.out.println("list3 = " + list3);

    System.out.println("\nTEST 3: Calling list empty");
    list1.clear();
    list2.add(0,new Integer(3));
    list2.add(1,new Integer(8));
    list2.add(2,new Integer(17));
    list3=list1.merge(list2);
    System.out.println("list1 = " + list1);
    System.out.println("list2 = " + list2);
    System.out.println("list3 = " + list3);

    System.out.println("\nTEST 4: Calling list shorter than param list");
    list1.clear();
    list2.clear();

    list1.add(0,new Integer(8));
    list1.add(1,new Integer(17));
    list2.add(0,new Integer(6));
    list2.add(1,new Integer(12));
    list2.add(2,new Integer(19));
    list2.add(3,new Integer(20));
    list3=list1.merge(list2);
    System.out.println("list1 = " + list1);
    System.out.println("list2 = " + list2);
    System.out.println("list3 = " + list3);
    System.out.println("\nTEST 5: Calling list longer than param list");
    list1.clear();
    list2.clear();
    list1.add(0,new Integer(8));
    list1.add(1,new Integer(17));
    list1.add(2,new Integer(18));
    list1.add(3,new Integer(20));
    list2.add(0,new Integer(6));
    list2.add(1,new Integer(12));
    list3=list1.merge(list2);
    System.out.println("list1 = " + list1);
    System.out.println("list2 = " + list2);
    System.out.println("list3 = " + list3);

    System.out.println("\nTEST 6: Equal sizes");
    list1.clear();
    list2.clear();
    list1.add(0,new Integer(8));
    list1.add(1,new Integer(17));
    list1.add(2,new Integer(18));
    list1.add(3,new Integer(20));
    list2.add(0,new Integer(6));
    list2.add(1,new Integer(12));
    list2.add(2,new Integer(19));
    list2.add(3,new Integer(21));

    list3=list1.merge(list2);
    System.out.println("list1 = " + list1);
    System.out.println("list2 = " + list2);
    System.out.println("list3 = " + list3);


  }

}


}
```