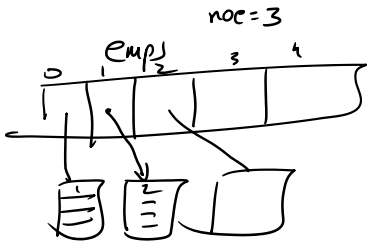
```
class Employee
{
    private int empno;
    private String name;
    private int salary;
    private String desig;

    Employee()
    {

    }
    Employee(int empno, String name, int salary, String desig)
    {   this.empno = empno;
        this.name = name;
        this.salary = salary;
        this.design = design;
    }
    =
    public String toString()
    {
        return "Empno:" + empno + "Name :" + name + "Salary:" + salary + "Designation :" + design;
    }
}
```

```
class EmployeeList
{
    private Employee[] emps;
    private int noe;

    EmployeeList()
    {
        emps = new Employee[5];
        noe = 0;
    }
    EmployeeList(int size)
    {
        emps = new Employee[size];
        noe = 0;
    }
    public void addEmployee(int eno, String nm, int sal, String design)
    {
        Employee e = new Employee();
        e.setEmpno(eno);
        e.setName(nm);
        e.setSalary(sal);
        e.setDesign(design);
        emps[noe] = e;
        noe++;
    }
    public void displayEmployee()
    {
        for(int i=0; i<noe; i++)
        {
            S.o.p("Empno:" + emps[i].getEmpno());
            S.o.p("Name:" + emps[i].getName());
            —
            —
```

ArrayList<Employee> list = new ArrayList<>();

list.add(e)

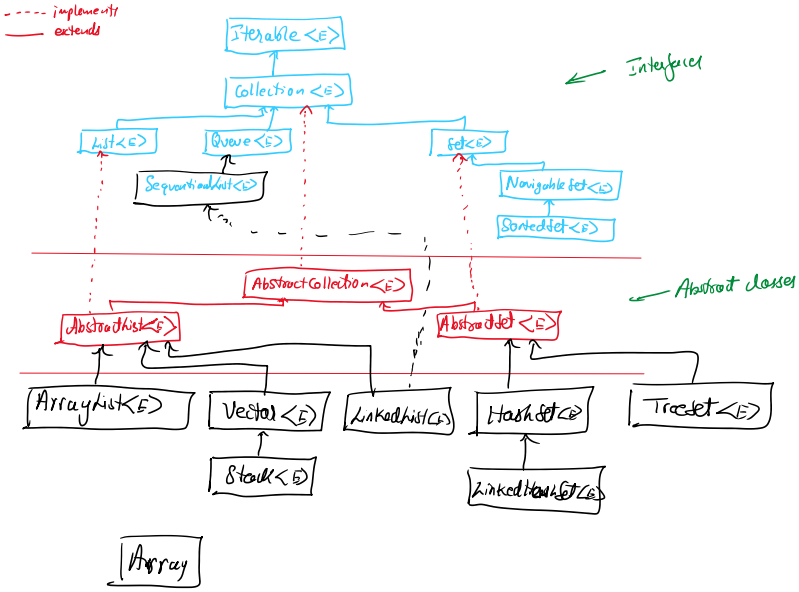



```
class Main
{
    psvm Main()
    {
        Scanner scan = new Scanner(...)
        int n = scan.nextInt();
        EmployeeList list
        for(int i=1; i<=n; i++)
        {
            int eno = scan.nextInt();
            scan.nextline();
            String nm = scan.nextLine();
            int sal = scan.nextInt();
            scan.nextline()
            String design = scan.nextLine();
```

2500
Dev
2
Ravi
3500
Tester
3
Kiran
4500
Manager

Collections → java.util

Data Structures
- Array → ArrayList
- LinkedList → LinkedList
- Tree → TreeSet
- Hashing → HashSet
- Stack → Stack
- Queue → Queue

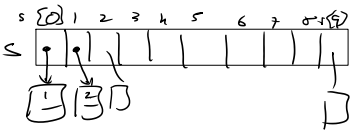


Array

List
1) Duplicate values
2) Order list
3) Multiple null values
4) DS
   - → ArrayList
   - → LinkedList
   - → Stack
   - → Vector

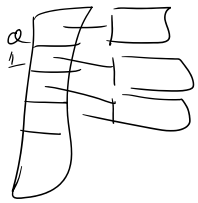
Set
1) Unique values
2) un-ordered set
3) one null value
4) DS
   - → HashSet
   - → LinkedHashSet
   - → TreeSet


```
Student[] s = new Student[10];
s[0] = new Student(1001, "Anil");
s[1] = new Student(1002, "Kiran");
```




```
s[10] = ____
```
→ ArrayIndexOutOfBoundException

```
ArrayList<Student> list = new ArrayList<>();
list.add(new Student(1, "Anil"));
list.add(new Student(2, "Kiran"));
```

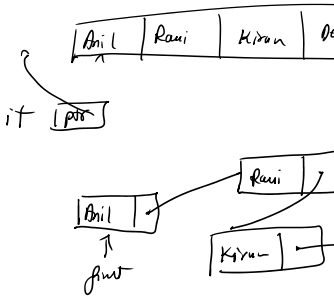


Collection

1) boolean add(E el)
2) boolean remove(E el)
3) boolean contains(E el)

```
ArrayList<String> list = new ArrayList<>();
list.add("Anil");
list.add("Ravi");
list.add("Kiran");
list.add("Dev");
```

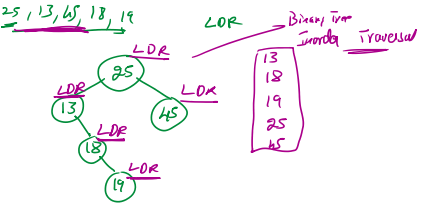
```
list.remove("Kiran");
```



4) boolean addAll(Collection<E> c)
5) boolean removeAll(Collection<E> c)
6) boolean retainAll(Collection<E> c)
7) boolean isEmpty()

```
LinkedList<String> list2 = new LinkedList<>();
list2.add("Ravi");
list2.add("Dev");
```

```
list.removeAll(list2)
```



Iterator<String> it = list.iterator()

1) E next()
2) boolean hasNext()
3) void remove()

```
while(it.hasNext())
{
    s.o.p(it.next());
}
```

Iterator<String> it = list.iterator();

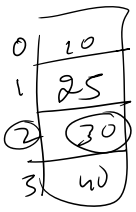
```
while(it.hasNext())
{
    String st = it.next();
    if(st.equals("kiran"))
        it.remove();
}
```

```
for(String data : list)
{
    s.o.p(data);
}
```

8) Iterator<E> iterator()

List

1) void add(int index, E el)
2) void remove(int index);
3) E get(int index)
4) E set(int index, E el)
5) int indexOf(E el)
   ↳ -1 → not found
6) int lastIndexOf(E el)
7) ListIterator<E> listIterator()
   ↳ hasNext()
   next
   remove
   previous
   hasPrevious



list.add(1, "Suresh")
int i = list.indexOf(30);

```
for(int i=0; i<list.size(); i++)
{
    s.o.p(list.get(i));
}
```

list.set(1, 25);

```
inte
{
}
```

Collections.sort((f,s) —

Collections.sort(list, (f,s)

face Comparator<E>

public int compare(E f, E s);

```
> f
    if(f > s)
        return 1;
    else if(f < s)
        return -1;
    else
        return 0;
});
-> f-s));
```

```
class IntComparator implements Comparator<Integer>
{
    public int compare(E f, E s)
    {
        return s-f;
    }
}

Collections.sort(list, new IntComparator());
```

```
ArrayList<Integer> list;
list = new Integer<>();
list.add(10);
list.add(20);
list.add(30);
list.add(40);
list.remove(new Integer(20));
```

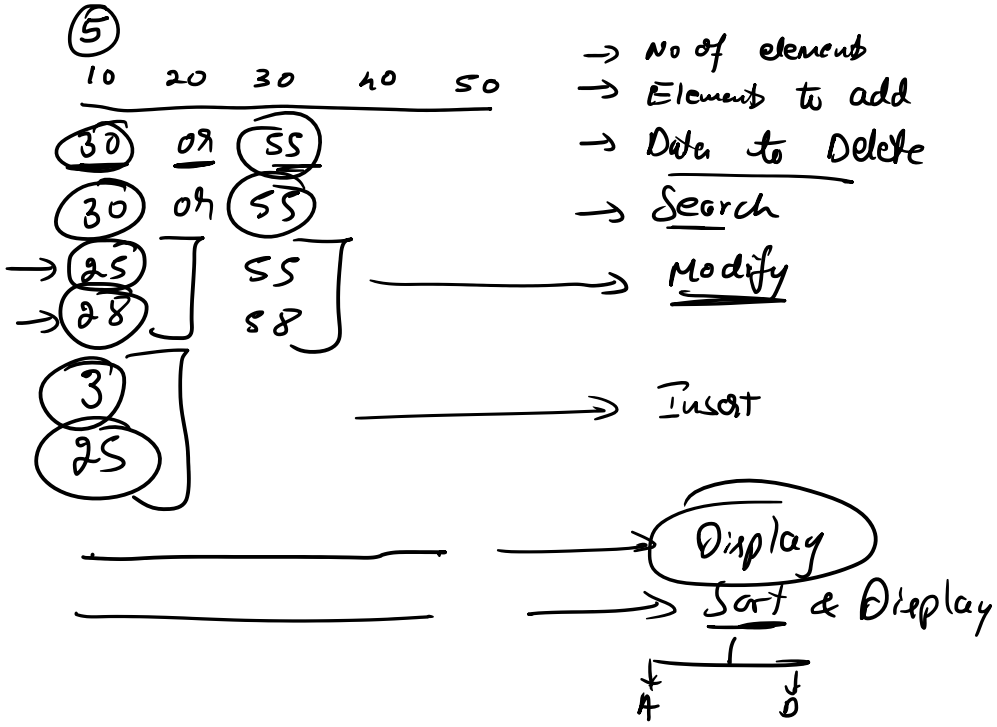
byte → Byte
short → Short
int → Integer
long → Long
float → Float
double → Double
char → Character
↓
Wapper classes
(java.lang)

14 December 2024 09:54

- → Add elements } C
- → Insert in-betwee. }
- → Delete by data → D
- → Search
- → Sort
- → Update → U
- → Display → R

Integer -
String - ✓
User-Defined data -

ArrayList
LinkedList
HashSet
LinkedHashSet
TreeSet
HashMap
TreeMap




```
class Main
{
    public static void main(String[] args)
    {
        ArrayList<Integer> list = new ArrayList<>();
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=1; i<=n; i++)
        {
            list.add(scan.nextInt());
        }
        int deleteData = scan.nextInt();
```

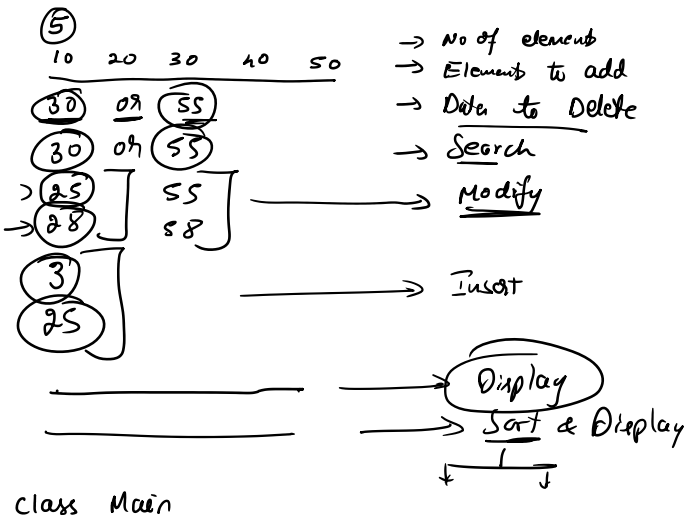
```
}
int deleteData = scan.nextInt();
if(list.contains(new Integer(deleteData)))
{
    list.remove(new Integer(deleteData));
    S.O.P(deleteData + " deleted successfully");
}
else
{
    S.O.P(deleteData + " not found");
}
int searchData = scan.nextInt();
int index = list.indexOf(new Integer(searchData));
if(index == -1)
    S.O.P(searchData + " not found");
else
    S.O.P(searaData + " found at position " + (index+1));

int modifyDataSearch = scan.nextInt();
int modifyDataTo = scan.nextInt();
index = list.indexOf(new Integer(modifyDataSearch));
if(index == -1)
    S.O.P(modifyDataSearch + " not found");
else
{
    list.set(index, modifyDataTo);
    S.O.P(modifyDataSearch + " modified successfully");
}
index = scan.nextInt();
int insertData = scan.nextInt();
list.add(index, insertData);
for(Integer d : list)
{
    S.O.println(d + " ");
}
```

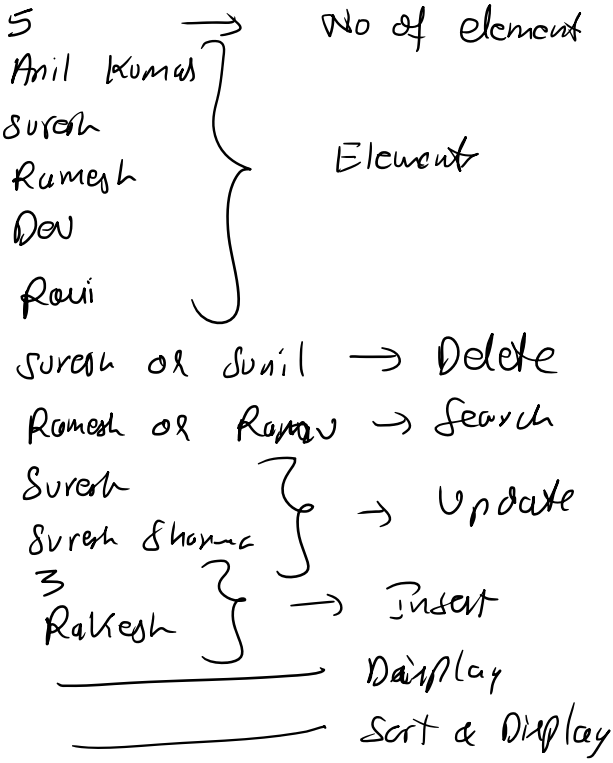
Comparable

```
        s.o.print(d + " ");
    }
    Collections.sort(list);
    s.o.p("in Sorted in Ascending order:");
    for(Integer d : list)
        s.o.print(d + " ");
    }
    Collections.sort(list, (d1, d2) -> Integer.compare(d2, d1));
    s.o.p("in Data sorted in descending order");
    for(Integer d: list)
        s.o.print(d + " ");
  }
}
```

Comparable

Comparator






```
class Main
{
    public static void main(String[] args)
    {
        ArrayList<String> list = new ArrayList<>();
        Scanner scan = new Scanner(System.in);
        int n= scan.nextInt();
        scan.nextLine();
        for(int i=1; i<=n; i++)
        {
            list.add(scan.nextLine());
        }
        String deleteData = scan.nextLine();
        if(list.contains(deleteData))
        {
            list.remove(deleteData)
            S.O.P(deleteData +" deleted successfully");
        }
        else
        {
            S.O.P(deleteData +" not found");
        }
    String searchData = scan.nextLine();
        int index = list.indexOf(searchData);
        if(index == -1)
            S.O.P(searchData+" not found");
        else
            S.O.P(searchData +" found at position "+(index+1));

    String modifyDataSearch = scan.nextLine();
    String modifyDataTo = scan.nextLine();
        index = list.indexOf(modifyDataSearch);
        if(index == -1)
            S.O.P(modifyDataSearch +" not found");
        else
        {
            list.set(index, modifyDataTo);
            S.O.P(modifyDataSearch +" modified successfully");
```

```
}
index = scan.nextInt();
scan.nextLine();
String insertData = scan.nextLine();
list.add(index, insertData);
for(String d : list)
    s.o.print(d + " ");
}
Collections.sort(list);
s.o.p("in Sorted in Ascending order:");
for(String d : list)
    s.o.print(d + " ");
}
Collections.sort(list, (d1, d2) -> d2.compareTo(d1));
s.o.p("in Data sorted in descending order");
for(String d: list)
    s.o.print(d+" ");
}
}
```

Comparable

arun
Arun

Comparator

> Integer.compare(d2,d1)

3 → No of Emp
1 ✓
Ravi ✓
3500 ✓
2
Kiran
3500
3
Ramesh
4500
→ Emp data

3 or 5 → Emp no to delete → D ✓
3 or 5 → Empno to search → R ✓
3 or 5
Ramesh Kumar
5000
→ update → U

→ Display
→ Display sorted by empno in Asc
→ " " " Name in Asc
" " " Salary in desc

Collections.sort(list)

```
class Employee implements Comparable<Employee>
{
    private int empno;
    private String name;
    private double salary;
    Constructor/Getter/setter/toString

    public int compareTo(Employee e2)
    {
        return Integer.compare(this.empno, e2.empno);
        return this.name.compareTo(e2.name);
        return Double.compare(this.salary, e2.salary);
    }

    public boolean equals(Object obj)
    {
        Employee e2 = (Employee) obj;
        if(this.empno == e2.empno)
            return true;
        else
            return false;
    }
}
```

interface Comparable<E>
{
  public int compareTo(E e2);
}

e1.compareTo(e2)
this → e1, e2

e
i
i+1
e[i].compareTo(e[i+1])
this → e2

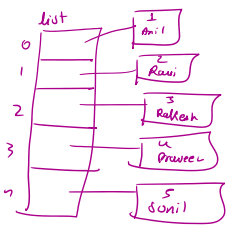
Asc → 1500 Ramesh, 2500 Anil, 2500 Arun
name ↓

Sort

```
for(int i=1; i<e.length(); i++)
{
    for(int j=0; j<e.length()-1; j++)
    {
        if(e[j].compareTo(e[j+1]) == 1)
        {
        }
    }
}
```

```
public int compareTo(Employee e2)
{
    if(this.salary > e2.salary)
        return +1;
    else if(this.salary < e2.salary)
        return -1;
    else
    {
        return this.name.compareTo(e2.name);
    }
}
```

```
}

class Main
{
    public static void main(String[] args)
    {
        LinkedList<Employee> list = new LinkedList<>();
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=1; i<=n; i++)
        {
            int eno = scan.nextInt();
            scan.nextLine();
            String nm = scan.nextLine();
            double sal = scan.nextDouble();
            Employee e = new Employee(eno, nm, sal);
            list.add(e);
        }
        int eno = scan.nextInt();
        Employee tmp = new Employee();
        tmp.setEmpno(eno);
        if(list.contains(tmp))
        {
            list.remove(tmp);
            S.O.P("Employee deleted successfully");
        }
        else
            S.O.P("Employee not found");
        int searchEno = scan.nextInt();
        tmp = new Employee();
        tmp.setEmpno(searchEno);
        int index = list.indexOf(tmp);
        if(index
```




```
ArrayList<Employee> list = new ArrayList<>();
list.contains(emp)
list.indexOf(emp)
list.remove(emp)
```

```
Employee tmp = new Employee();
tmp.setEmpno(5);
if(list.contains(tmp))
    list.remove(tmp);
```

```
tmp = new Employee();
tmp.setEmpno(searchEno);
int index = list.indexOf(tmp);
if(index == -1)
    S.O.P("Employee not found");
else
    S.O.P("Employee found at position: "+(index+1));

int eeno = scan.nextInt();
scan.nextLine();
String upName = scan.nextLine();
double upSal = scan.nextDouble();

tmp = new Employee(searchEno, upName, upSal);
index = list.indexOf(tmp);
if(index == -1)
    S.O.P("Employee not found");
else
{
    list.set(index, tmp);
    S.O.P("Successfully updated");
}

S.O.P("Employee list");
for(Employee x : list)
    S.O.P(x);
}
S.O.P("Sorted based on empno Asc");
Collections.sort(list);
for(Employee x : list)
    S.O.P(x);
S.O.P("Sorted based on name Asc");
Collections.sort(list, new NameComparator());
for(Employee x : list)
    S.O.P(x);
S.O.P("Sorted based on Salary Asc");
Collections.sort(list, new SalaryComparator());
for(Employee x : list)
    S.O.P(x);
}
}
```

```
class NameComparator implements Comparator<Employee>
{
    public int compare(Employee e1, Employee e2)
    {
        return e1.getName().compareTo(e2.getName());
    }
}
class SalaryComparator implements Comparator<Employee>
{
    public int compare(Employee e1, Employee e2)
    {
    return Double.compare(e1.getSalary(), e2.getSalary());
}
```