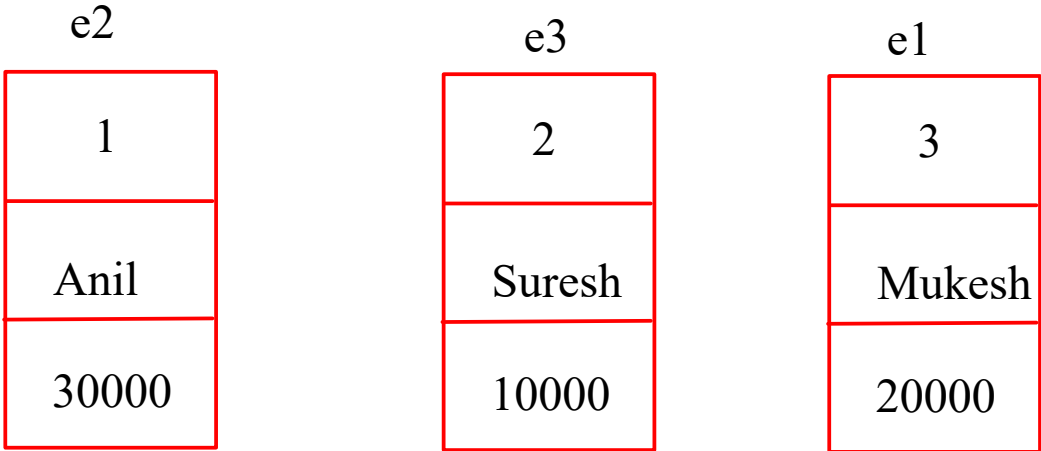


Generics

- class
- method
- interface

```
class Box<T>{  
  
}  
  
void display(Box<? super Integer> b){  
  
}  
  
<T>void printArray(T []arr)  
  
Box<Integer>b1 = new Box<Integer>();  
Box<String>b2 = new Box<>();  
Box<Date>b3 = new Box<>();  
Box<Double>b4 = new Box<>();
```

```
interface Comparable<T>{  
    int compareTo(T o);  
}  
  
interface Comparator<T>{  
    int compare(T o1, To2);  
}
```



```
arr[] = {e1,e2,e3};  
  
Arrays.sort(arr);  
  
void sort(Object arr[]){  
    arr[0];  
    Comparable c= arr[1];  
    arrr[2];  
  
    if(c.compareTo(arr[2])>0)  
        swap()  
}
```

```
class Employee implements Comparable<Employee>{  
  
    int compareTo(Employee o){  
        return this.id - o.id;  
    }  
}
```

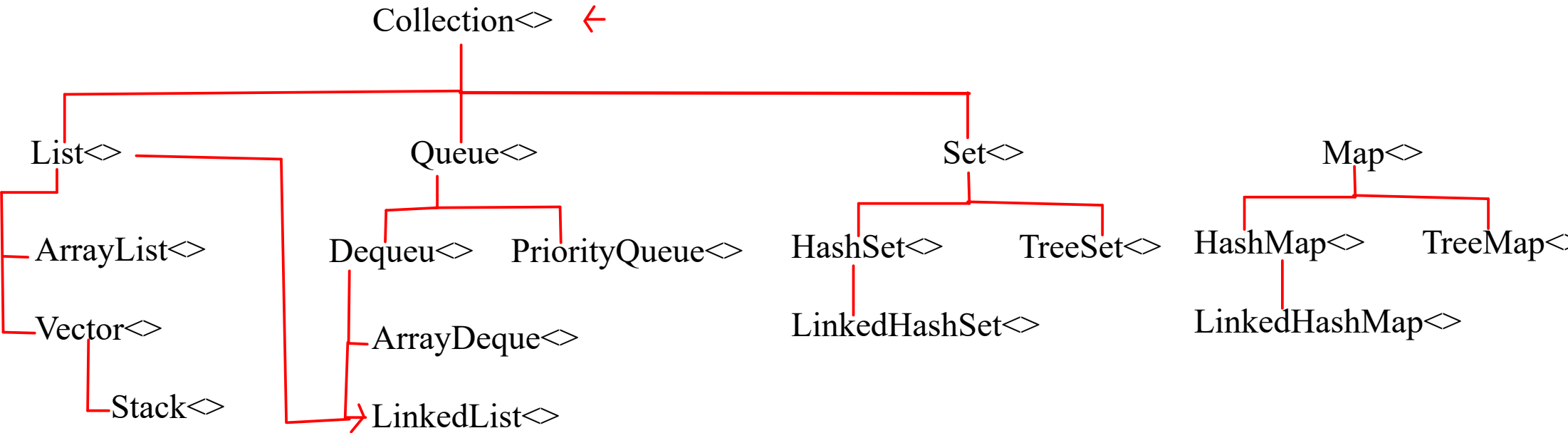
```
Arrays.sort(arr, empSalComp);
```

```
class EmpSalaryComparator implements Comparator<Employee>{  
    int compare(T o1, T o2){  
        return 0;  
    }  
}
```

```
void sort(T arr[], Comparator comp){  
    arr[0];  
    arr[1];  
    arrr[2];  
  
    if(comp.compare(arr[0],arr[1])>0)  
        swap()  
}
```

```
EmpSalaryComparator empSalComp= new EmpSalaryComparator();
```

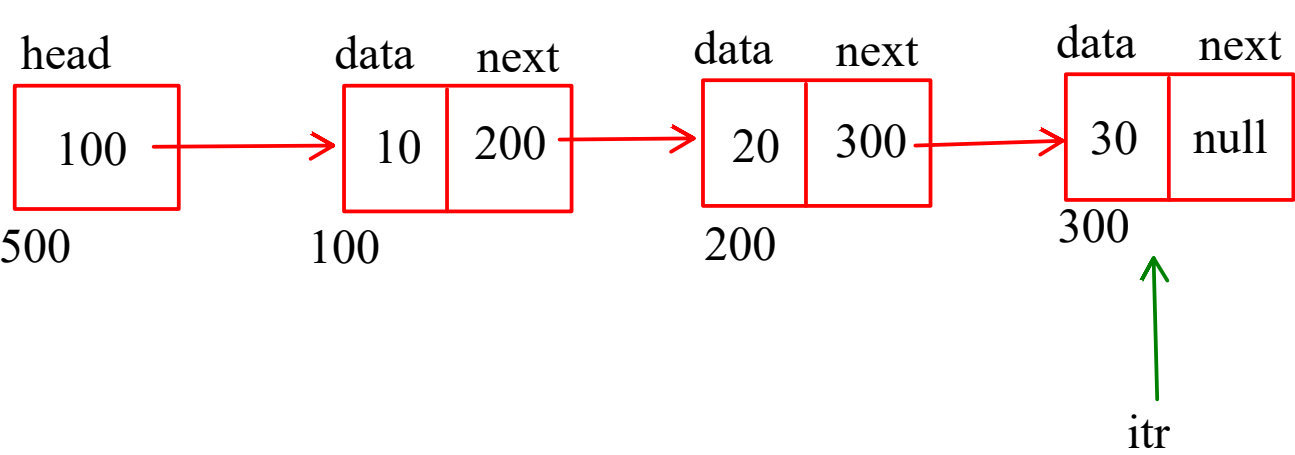
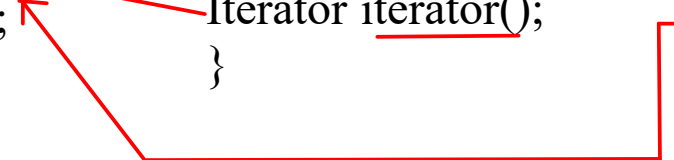
Collection



interface Iterator{
boolean hasNext();
E next();
}

interface Iterable{
Iterator iterator();
}

interface Collection extends Iterable{
Iterator iterator();
}



```
Iterator<Integer> itr = c1.iterator();  
while (itr.hasNext()) {  
    Integer i1 = itr.next();  
    System.out.println(i1);  
}
```

Traversals

- 1. For-Each
 - Available for Collection interface and all its sub classes
- 2. Iterator
 - Available for Collection interface and all its sub classes
- 3. For-Loop index based
 - Available only for the List interface and its sub classes
- 4. Enumeration
 - Available only for the Vector

Iterable and Iterator

- Iterator is used to traverse the collection
- All the subclass have to implement the Iterator and they have to return the object of their implemented Iterator.
- Sub classes have to return the object of Iterator through the method iterator() inherited from the Iterable interface
- Iterable yields an Iterator
- It means Iterable interface helps to provide a method design that forces all the sub classes to implement the Iterator interface and return its object.

Anil, Suresh, Mukesh, Ramesh,

0	1	2	3
↑			
		i/P	

Anil	- 30000
Mukesh	- 20000
Anil	- 15000
Suresh	- 40000
Mukesh	- 35000

Arrays
Collections

o/p-1		o/p	
Anil	15000	Anil	30000
Anil	30000	Anil	15000
Mukesh	20000	Mukesh	35000
Mukesh	35000	Mukesh	20000
Suresh	40000	Suresh	40000