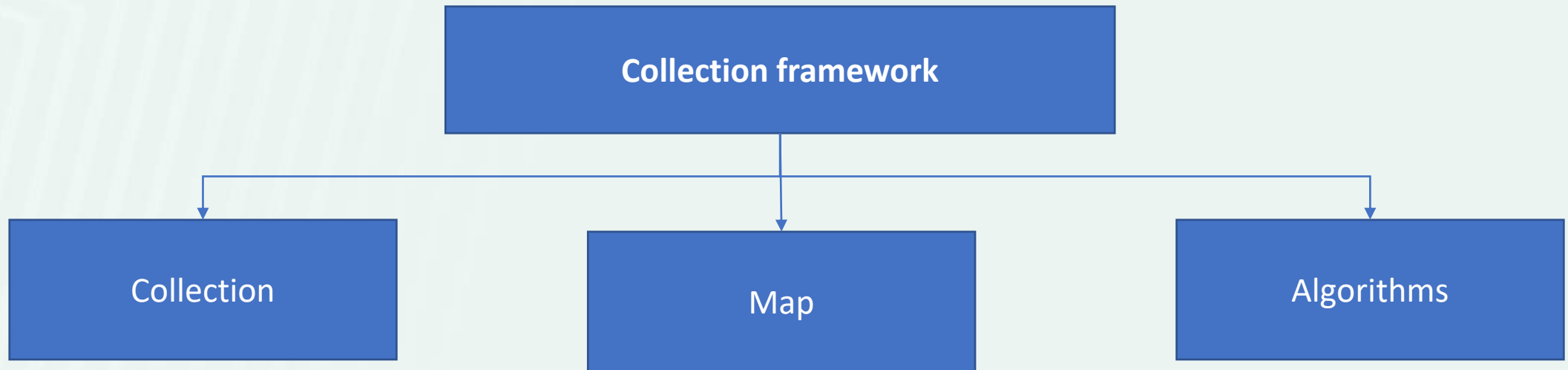


Java da to'plamlar. Collection va List Interfacelari.

Reja:

- **Collection framework**
- **Collection interface**
- **List interface**


Collection framework



Collection(To'plam)

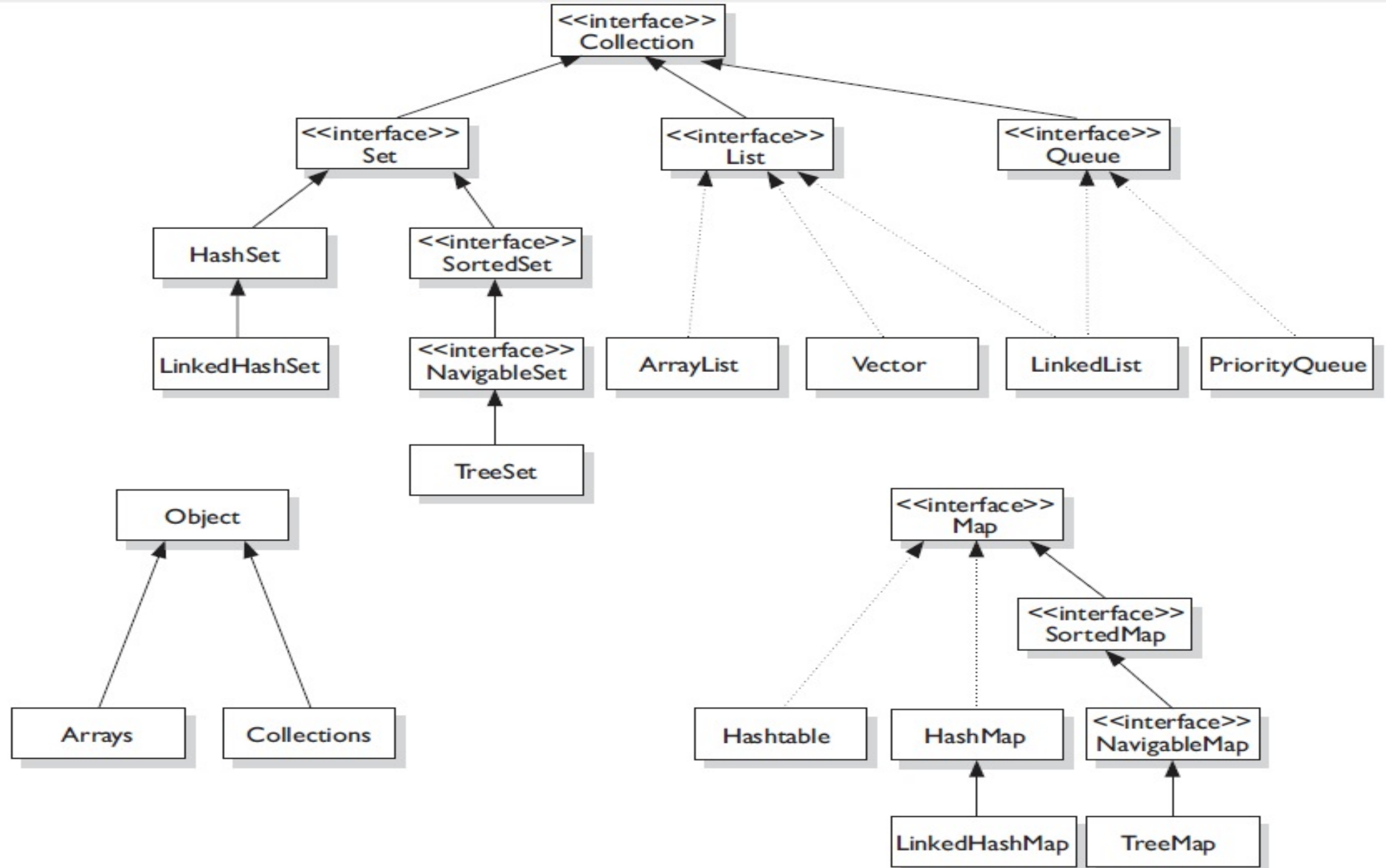
Collection(To'plam) – ob'ektlarni bitta guruh yoki to'plamga yig'ib ushbu guruh yoki to'plam bilan ishlash imkonini yaratadi. Ya'ni collection bizga ob'ektlar guruhi bilan quyidagi amallarni bajarish imkonini beradi: qo'shish, o'chirish, ko'rish va boshqa maxsus amallar.





Ob'ektlar to'plami mavjud bo'lib ular ustida amallar bajarish lozim bo'lsa, buning uchun bizga maxsus klass kerak bo'ladi. Ushbu klass collection dir. Bunda 2 ta alohida hoaltni qayd etish lozim:

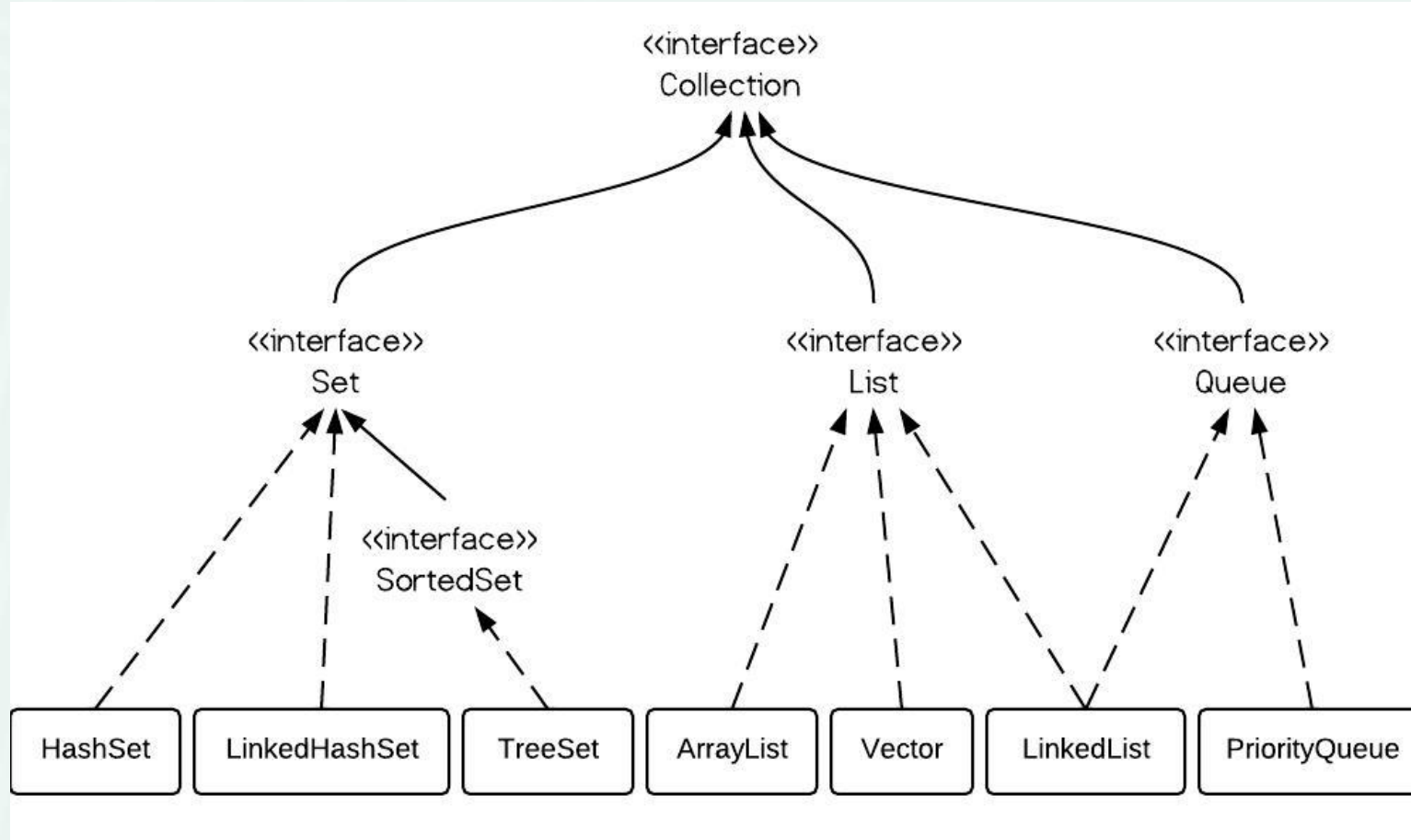
- 1) To'plam odatda (bo'lishi shart) qo'shish, o'chirish, ro'yxat bo'yicha o'tish, elementni olish kabi asosiy funksiyalariga ega boladi. Shuningdek, qo'shimcha o'ziga xos bo'lgan imkoniyatlarga ega bo'lish zarurati ham mavjud va aynan shu narsa collection klasslarining turli xilligini ta'minlaydi.
- 2) To'plam asosan bir toifaga ega ob'ektlar jamlanadi. Istisno holatlari ham bo'lishi mumkin



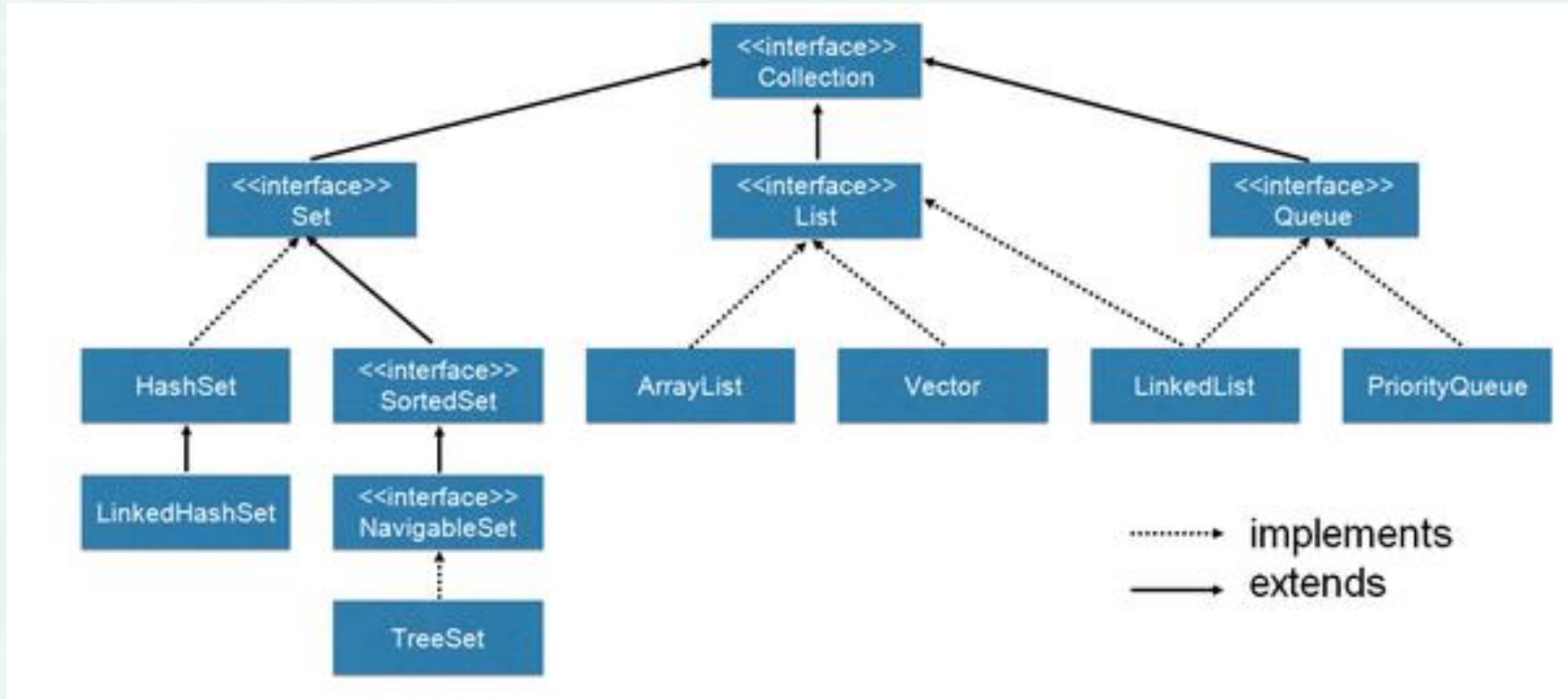
- **Collection** – kolleksiyalar ierarxiyasi asosidir. Uning 3 quyi Interface lari mavjud: Set, List, Queue. Javada Collection ni to'g'ridan to'g'ri realizasiya qilib bo'lmaydi. Lekin uning quyi interfacelarini realizasiya qilsa bo'ladi.
- **Map** – bunda elementlar kalit – qiymat juftligi shaklida saqlanadi. Kalit takrorlanmaydi va bitta kalitga bittadan ortiq qiymat berilishi mumkin emas.
- **Algorithms** – Collectionlar ustida bajariladigan amallar algortimlari. Masalan saralash, qidirish, to'ldirish o'cherish va hk.



Collection Interface



Collection Interface




- **List** – tartiblangan kolleksiya (ketma-ketlikdagi). Listda takrorlanuvchi elementlar bo'lishi mumkin. Foydalanuvchi Listning istalgan elementiga uning butun sonli indeksi(joylashgan o'rni) orqali murojat qilishi mumkin.
- **Set** – o'zida takrorlanuvchi elementlarni saqlashga ruxsat etmaydi. Barcha element unikal bo'ladi.
- **Queue**(очеред) – bir necha elementni ishlov berishdan oldin saqlash uchun ishlatiladi. Collection ning asosiy amallaridan tashqari qo'shimcha qo'yish, olish va tekshirish amallari mavjud. Elementlarini odatda FIFO (first-in, first-out) shaklda tartiblaydi.

Collection –interfeysi asosan maksimal umumiylik talab qilinadigan hollarda ishlatiladi.

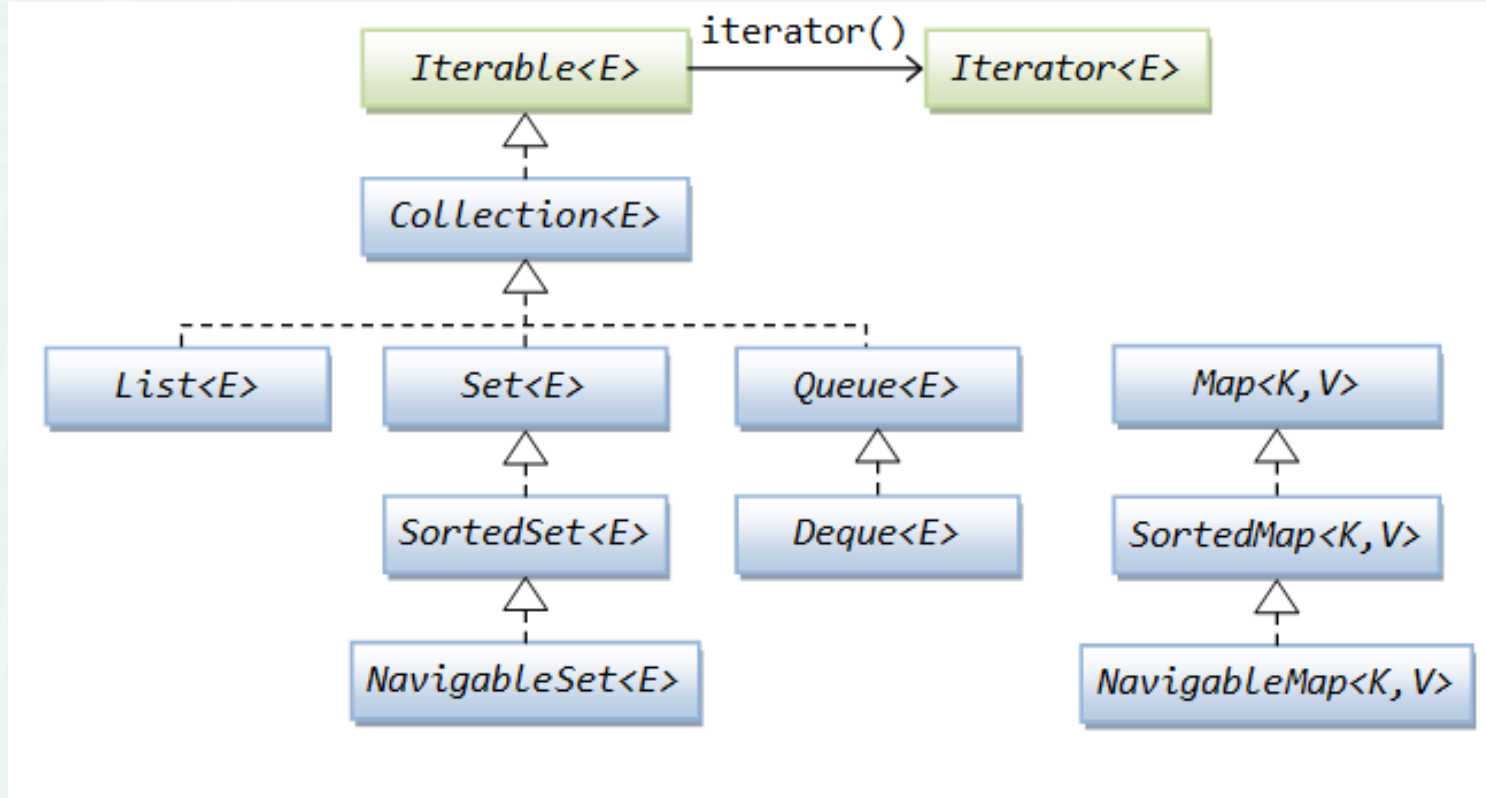
Uning quyidagi asosiy methodlari bor:

1. **boolean add(item)** – kolleksiyaga itemni qo'shadi va true qaytaradi aks holda false.
2. **Iterator<E> iterator()** – elementlar bo'yicha o'tishga mo'ljallangan Iteratorni qaytaradi.
3. **boolean addAll (Collection<? extends E> col)** – kolleksiyaga col koleksiyasining barcha elementlarini qo'shadi va true qaytaradi aks holda false.
4. **void clear ()** – kolleksiyadan barcha elementlarni o'chiradi.
5. **boolean contains (Object item)** – agarda item kolleksiyaning ichida bor bo'lsa true aks holda false qaytaradi.
6. **boolean isEmpty ()** – agarada kolleksiya bo'sh bo'lsa true aks holda false qaytaradi.
7. **boolean remove (Object item)** – item ni kolleksiyadan o'chiradi va true qaytaradi aks holda false.
8. **boolean removeAll (Collection<?> col)** – joriy kolleksiyadan col koleksiyasi elementlarini o'chiradi va true qaytaradi aks holda false.
9. **boolean retainAll (Collection<?> col)** - joriy kolleksiyadan col koleksiyasi elementlaridan boshqasini o'chiradi va true qaytaradi aks holda false.
10. **int size ()** – kolleksiyadagi elementlar sonini qaytaradi.
11. **Object[] toArray ()** – kolleksiyaning barcha elementlaridan tashkil topgan massiv qaytaradi.



```
Collection col = new ArrayList();  
col.add("1");  
col.add("2");  
col.add("3");
```

Iterator



Iterator

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

List Interface:

- **Arraylist**
- **LinkedList**
- **ArrayList vs LinkedList**

List Interface



Oddiy ro'yxatlarni yaratish uchun Collection interface ini kengaytiruvchi List interface idan foydalaniladi. Uning ko'p ishlatiladigan quyidagi methodlari bor:

- **void add(int index, E obj)** - Listga **index** indeks bo'yicha obj ob'ektini qo'shadi.
- **boolean addAll(index, collection)** - Listga **index** indeks bo'yicha collection elementlarini qo'shadi.
- **E get(int index)** – ro'yxatdan **index** indeksdagi ob'ektni qaytaradi.
- **int indexOf(Object obj)** – ro'xatdagi birinchi joylashgan **obj** ob'ektining indeksini qaytaradi aks holda -1.
- **int lastIndexOf(Object obj)** – ro'xatdagi oxirgi joylashgan **obj** ob'ektining indeksini qaytaradi aks holda -1.
- **ListIterator<E> listIterator ()** – elementlar bo'yicha o'tishga mo'ljallangan Iteratorni qaytaradi.
- **E remove(int index)** – **index** indeksdagi elementni o'chirib ochirilgan elementni qaytaradi.
- **E set(int index, E obj)** – **index** indeksdagi elementga obj ob'ekti qiymatini o'zlashtiradi.
- **void sort(Comparator<? super E> comp)** – ro'yxatni **comp** comparator yordamida sortirovka qiladi.
- **List<E> subList(int start, int end)** – **start** va **end** indeksleri oralig'idagi elementlar to'plamini qaytaradi.

ArrayList

- **ArrayList Features**
- **How ArrayList Works?**
- **Creating an ArrayList**
- **Access elements from ArrayList**
- **Iterating over an ArrayList**
- **Sorting an ArrayList**

- List interface ning default realizatsiyasi ArrayList klassidir.
- ArrayList o'zida oddiy ro'yxatni saqlashga mo'ljallangan va elementlari soni belgilab qo'yilmagan array(massiv)ning analogidir.

ArrayList Features

- 1.Ordered (Tartiblangan)** – ArrayList elementlari qo'shilish tartibida tartiblanadi.
- 2.Index based** – index bo'yicha ixtiyoriy elementga murojat qilish mumkin. Index 0 dan boshlanadi.
- 3.Dynamic resizing** – Hajmi dinamik o'zgaruvchan.
- 4.Non synchronized** – ArrayList sinxron emas.
- 5.Duplicates allowed** –bitta objectni takror saqlashga ruxsat etiladi.

How ArrayList Works?

```
public class ArrayList<E> extends AbstractList<E> implements List<E>,  
RandomAccess, Cloneable, java.io.Serializable {
```

```
    transient Object[] elementData;    //backing array
```

```
    private int size;                //array or list size
```

```
    //more code
```

```
}
```

```
List list = new ArrayList();
```

int size	0
Object[] els	
int modCount	0

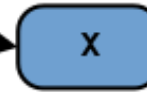


Singleton empty array



```
list.add(x);
```

int size	1
Object[] els	
int modCount	1



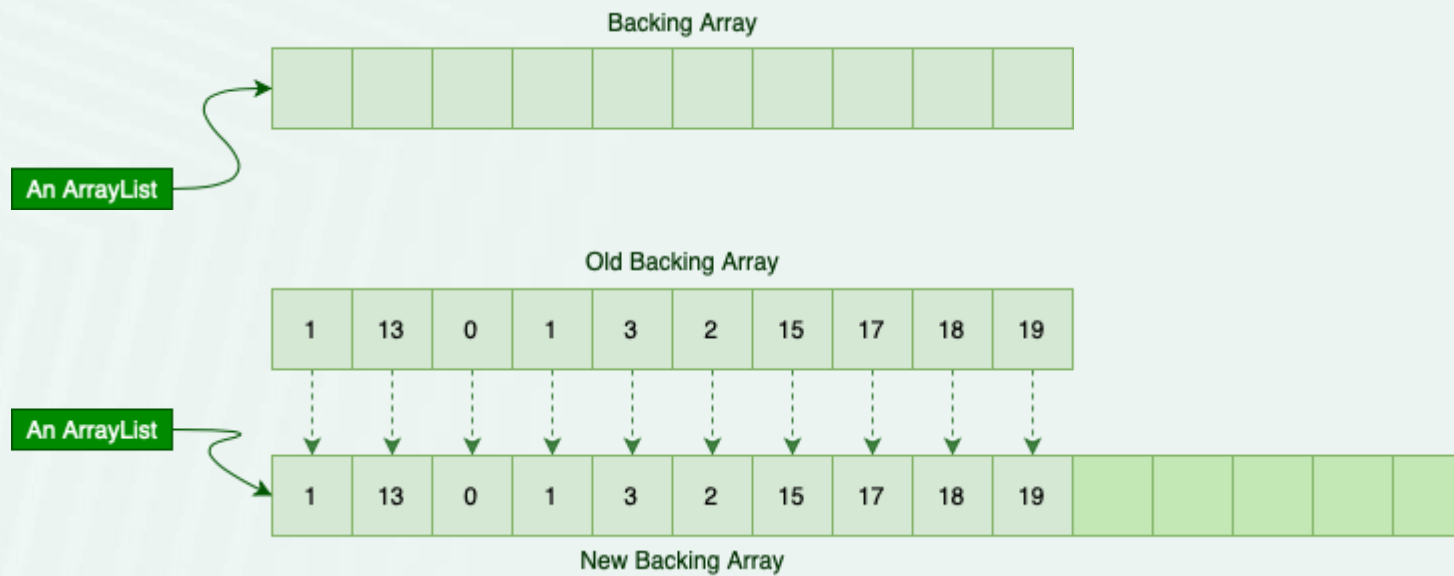
Default capacity = 10



```
list.clear();
```

int size	0
Object[] els	
int modCount	2






Creating an ArrayList

Quyidagi konstruktorlardan biri orqali yaratish mumkin:

Constructor	Description
<code>ArrayList()</code>	Default konstruktor. Dastlabki uzunligi 10 teng bo'lgan bo'sh ArrayList yaratadi.
<code>ArrayList(int capacity)</code>	Uzunligi capacity teng bo'lgan bo'sh ArrayList yaratadi.
<code>ArrayList(Collection<? extends E> c)</code>	c collection elementlaridan iborat ArrayList yaratadi.



//Non-generic arraylist - NOT RECOMMENDED !!

ArrayList list = **new** ArrayList();

//Generic Arraylist with default capacity

List<Integer> numbers = **new** ArrayList<>();

//Generic Arraylist with the given capacity

List<Integer> numbers = **new** ArrayList<>(6);

//Generic Arraylist initialized with another collection

List<Integer> numbers = **new** ArrayList<>(Arrays.asList(1,2,3,4,5));

Access elements from ArrayList

```
ArrayList<String> alphabetsList = new ArrayList<>(Arrays.asList("A", "B", "C"));
```

```
String aChar = alphabetsList.get(0); // A get(index)
```

```
Iterator<String> iterator = alphabetsList.iterator();
```

```
while(iterator.hasNext())  
{  
    System.out.println(iterator.next()); // iterator.next()  
}
```

Iterating over an ArrayList

```
ArrayList<Integer> digits = new ArrayList<>(Arrays.asList(1,2,3,4,5,6));
```

- **Iterator**

```
Iterator<Integer> iterator = digits.iterator();
```

```
while(iterator.hasNext()) {  
    System.out.println(iterator.next());  
}
```

- **For loop**

```
for(int i = 0; i < digits.size(); i++) {  
    System.out.print(digits.get(i));  
}
```

- **forEach loop**

```
for(Integer d : digits) {  
    System.out.print(d);  
}
```

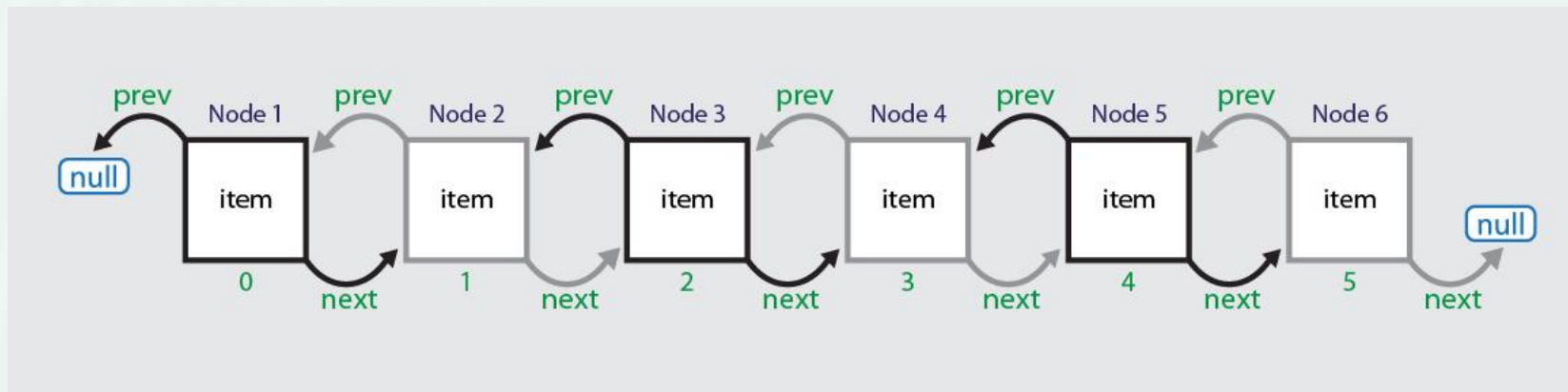
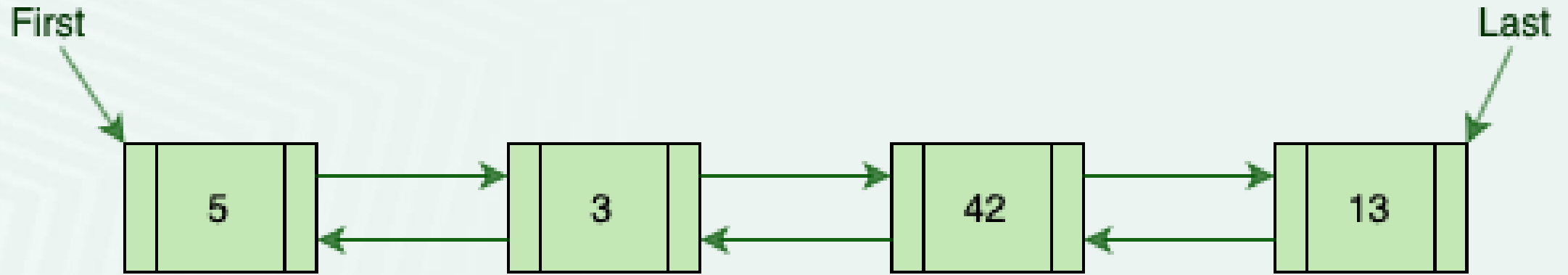
- **Collection forEach**

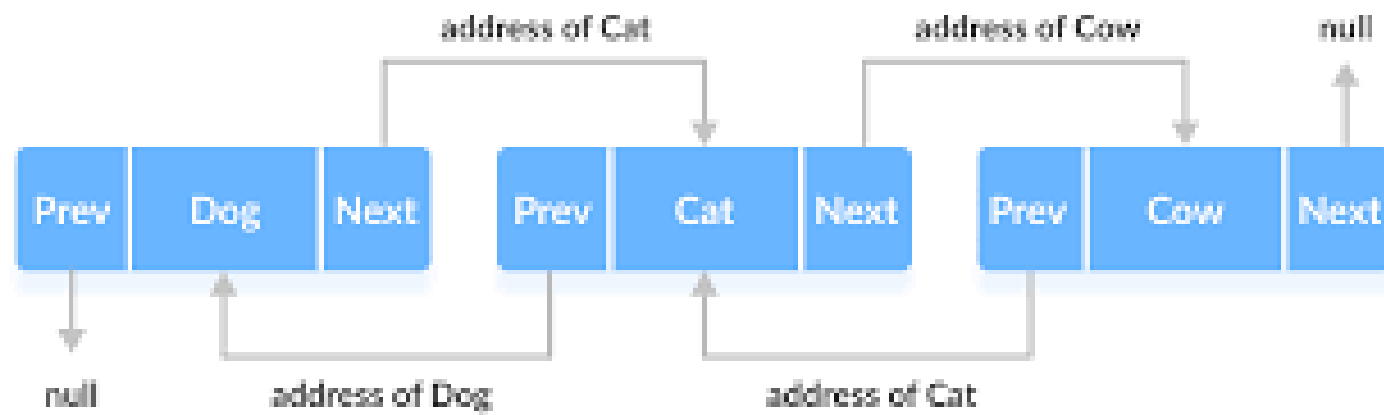
```
digits.forEach(integer -> System.out.println(integer));
```

Sorting an ArrayList

1. **Comparator class**
2. **Lambda Expression**
3. **Comparator.comparing method**

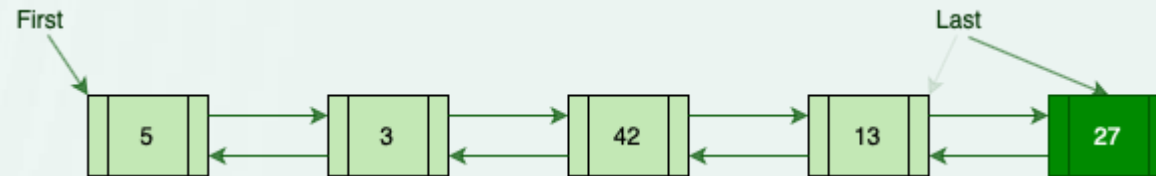
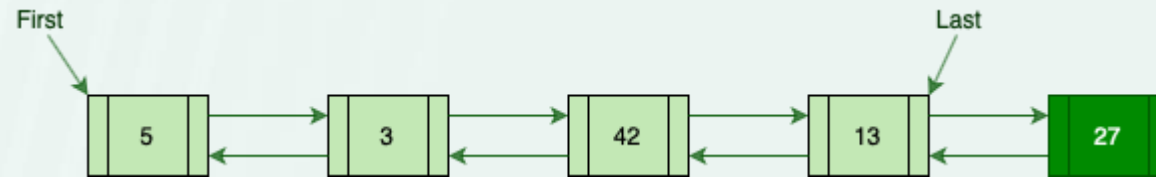
LinkedList





LinkedList Implementation in Java

Add new element



METHOD	DESCRIPTION
<u>addFirst(E e)</u>	This method Inserts the specified element at the beginning of this list.
<u>addLast(E e)</u>	This method Appends the specified element to the end of this list.
<u>descendingIterator()</u>	This method returns an iterator over the elements in this deque in reverse sequential order.
<u>getFirst()</u>	This method returns the first element in this list.
<u>getLast()</u>	This method returns the last element in this list.
<u>removeFirst()</u>	This method removes and returns the first element from this list.
<u>removeLast()</u>	This method removes and returns the last element from this list.

ArrayList vs LinkedList

