Collection framework içerisindeki bütün yapıları önemli methodlarıyla örnekleyip açıklayınız.

1. List

1.1. ArrayList:

ArrayListler dinamik olarak verileri saklamamızı sağlar. En büyük avantajlarından biri saklayabileceğimiz verilerin bir sınırının olmamasıdır: ArrayListlerin kapasaitesi artırılıp azaltılabilir. List interface'ini implement eder.

boolean add(Integer e): Elemanı listenin sonuna ekler.

Benzer şekilde "addAll" metotları, bir collection'daki elemanları listenin sonuna ya da listedeki belli bir yere ekler. Yine void int(index, Integer element), elemanı listenin istenen indisine ekler ve sonra gelen elemanları sağa kaydırır.

Örnek Kod:
```
ArrayList<Integer> sayi = new ArrayList<>();
sayi.add(1); //1 sayısını listeye ekler.
sayi.add(2); //2 sayısını listeye ekler.
```

void clear(): Listenin tüm elemanlarını siler.

void ensureCapacity(int min_capacity): ArrayList instance'ının kapasitesini artırır. Gerekli durumda, ArrayList'in en az min capacity kadar eleman tutmasını garantiye alır.

get(int index): İndis numarası ile listenin belirli bir elemanını bulmamızı sağlar.

Örnek Kod:
```
System.out.println(sayi.get(0)); //0.indisteki 1 sayısını döner.
```

boolean isEmpty(): Listenin boş olup olmadığını kontrol eder. Boşsa true değilse false döner.

Örnek Kod:
```
System.out.println(sayi.isEmpty()); //Liste boş olmadığı için false döner.
```

Object clone(): ArrayList'in kopyasını oluşturur.

Örnek Kod:
```
ArrayList<Integer> sayi2 = (ArrayList<Integer> sayi.clone();
System.out.println(sayi2.get(0)); //Klonlanan sayi2 listesinin 0.indisindeki 1 sayısını döner.
```

boolean contains(Object o): Liste belirli bir elemanı içeriyorsa true yoksa false döner.

Örnek Kod: System.out.println(sayi.contains(1)); //1 sayısı listede bulunduğundan true döner.

Benzer şekilde, containsAll bir collection'un elemanları listenin içindeyse true değilse false döner.

int indexOf(Object o): Belirli bir elemanın listede bulunduğu ilk indisi döner. Eğer eleman yoksa -1 döner.

Örnek Kod: System.out.println(sayi.indexOf(1)); //1 sayısının indisi olan 0'ı döner.

boolean remove(Object o): Eğer mevcutsa, listenin belirli bir indisindeki elemanı siler.

Örnek Kod: sayi.remove(1);

System.out.println(sayi.get(1)); //1. indisteki 2 sayısını sildiğimizden hata verir.

Benzer şekilde, remove(int index) metodu; listenin verilen indisindeki elemanı, eleman mevcutsa siler ve diğer elemanları sola kaydırır. removeAll(Collection<> c) ise verilen collection'daki tüm elemanları listeden siler.

void sort(Comparator<> c): Liste elemanlarını istenen karşılaştırıcıya göre sıralar.

Örnek Kod: sayi.add(3);

```
sayi.add(2);
sayi.sort(null);
for (int i = 0; i < sayi.size(); i++
{
```

System.out.println(sayi.get(i)); //Listeyi null ile sıraladığımızdan sayıların doğal sırasını döner. Yani 1, 2, 3 şeklinde

```
}
```

int size(): Listenin eleman sayısını verir.

Örnek Kod: System.out.println(sayi.size()); //Listede üç sayı bulunduğundan size olarak 3 döner.

1.2. LinkedList: Çift bağlantılı liste kullanarak elemanları depolar. Her eleman data ve adres bilgisine sahip bir objedir. AbstractList sınıfını miras alır ve Deque ile List interfacelerini implement eder. ArrayList'ten farklı olarak önemli metotları şu şekildedir:

boolean offer(Integer e): Elemanı listenin sonuna ekler.
Örnek Kod: Yukarıdaki ArrayList'i LinkedList'e çevirelim.

```
sayi.offer(4);
for (int i = 0; i < sayi.size(); i++
{
        System.out.println(sayi.get(i)); //4 sayısını sonna ekler. 1, 3, 2, 4 şeklinde döner.
}
```

boolean offerFirst(Integer e): Elemanı listenin başına sokar.
Örnek Kod: sayi.offerFirst(0);
System.out.println(sayi.get(0)); // 0 sayısını listenin başına, yani 0. indise ekledi ve 0 sayısını döndü.

int element(): Listenin ilk elemanını alır ama elemanı silmez. Liste boşsa hata verir.
Örnek Kod: System.out.println(sayi.element()); //İlk elemanı döner yani 0. Ayrıca bu elemanı silmez.

int getFirst(): Listenin ilk elemanını döner.
Örnek Kod: System.out.println(sayi.getFirst()); //İlk elemanı döner: 0

int getLast(): Listenin son elemanını döner.
Örnek Kod: System.out.println(sayi.getLast()); //Son elemanı döner: 4

int peek(): Listenin ilk elemanını alır ama listeyi silmez. Liste boşsa null döner.
Örnek Kod: System.out.println(sayi.peek()) //İlk elemanı döner: 0. Liste boş olsaydı null dönerdi.
Benzer şekilde int peekFirst() ilk elemanı alır ama silmez, int peekLast() son elemanı alır ama elemanı silmez. Liste boşsa ikisi de null döner.

int poll: Listenin ilk elemanını alır ve elemanı siler.
Benzer şekilde int pollFirst() ilk elemanı alır ama silmez, int pollLast() son elemanı alır ve elemanı siler. Liste boşsa ikisi de null döner.

Örnek Kod: sayi.poll();

System.out.println(sayi.get(0)); //İlk elemanı, yani 0'ı sildi. Dolayısıyla ilk eleman tekrar 1 oldu.

int pop(): Listenin ilk elemanını döner ve elemanı siler.

Örnek kod: System.out.println(sayi.pop()); //1 elemanını döner ve elemanı siler. 0. indisi yazdırırsak artık 3 olduğunu görürüz.

void push(Integer e): Elemanı listenin başına sokar.

Örnek kod: sayi.push(1);

System.out.println(sayi.get(0)); //1 sayısını listenin başsına soktu ve 0. indis tekrar 1 oldu. addFirst metodu ile eşdeğerdir

int set(int index, Integer element): Verilen indisteki elemanı, verilen yeni elemanla değiştirir.

Örnek kod: sayi.set(0, 0);

System.out.println(sayi.get(0)); //0. indisteki 1 sayısını 0 i değiştirdi ve 0. indisin artık 0 olduğunu gösterdi.

1.3. Vector: ArrayList gibi Vector'ın da kapasitesi artırılıp azaltılabilir. List interface'ini implement eder. Arraylerden farklı olarak vektörlere bir değişken tipi verilmesi gerekmez. Bunun yerine Vector, diğer nesnelerden referans alan dinamik liste içerir. Önceki maddelerden farklı önemli metotları şunlardır:

int capacity(): Vector'ın o anki kapasitesini döner.

Örnek kod: Vector<Integer> sayi = new Vector<>();

vector.add(1);

vector.add(2);

vector.add(3);

System.out.println(sayi.capacity()); //Default olarak 10 değerini döndü. Bu değere ulaşıldığında kapasite otomatik artar ya da Vector için de ensureCapacity metodu kullanılabilir.

elementAt(int index): Verilen indisteki bileşeni döner.

Örnek Kod: System.out.println(sayi.elementAt(0)); //Listenin 0. indisindeki 1 elemanını döndü.

firstElement: Vector'ın ilk elemanını döner.

Örnek Kod: System.out.println(sayi.firstElement()); //Listenin ilk elemanını yani 1 sayısını döndü.

insertElementAt(Object object, int index): Verilen objeyi verilen indise Vector'ın bileşeni olarak koyar. Sonra gelen bileşenleri bir aşağı kaydırır(ya da indislerini bir artırır).

Örnek Kod: sayi.insertElementAt(0, 0);

System.out.println(sayi.get(0)); //0 elemanını Vector'ın 0. indisine ekler ve 0. indise baktığımızda artık 0 olduğunu görürüz.

lastElement(): Son bileşeni döner.

Örnek kod: System.out.println(sayi.lastElement()); //Son eleman olan 3'ü döner.

boolean removeElement(Object obj): Verilen objenin Vector'daki ilk örneğini siler ve sonra gelen bileşenleri yukarı kaydırır(ya da indislerini bir azaltır).

Örnek kod: sayi.add(0);

```
sayi.removeElement(0);
for (int i = 0; i < sayi.size(); i++)
{
        System.out.println(sayi.get(i)); //0 sayısının ilk örneği olan 0. indisteki 0 sayısını sildi ancak sona eklediğimiz 0 kaldı.
}
```

void removeAllElements(): Tüm bileşenleri siler.

void removeElementAt(int index): Verilen indisteki bileşeni siler. Sonra gelen bileşenleri yukarı kaydırır(ya da indislerini bir azaltır).

Örnek kod: sayi.removeElementAt(3);

```
for (int i = 0; i < sayi.size(); i++)
{
        System.out.println(sayi.get(i)); //3. indisteki 0 sayısını sildi.
}
```

set(int index, object element): Verilen elemanı, verilen indisteki elemanla değiştirir.

Örnek Kod: sayi.set(0, 0);

System.out.println(sayi.get(0)); //0. indisteki 1 sayısını 0 ile değiştirdi.

1.4. Stack: Vector sınıfının bir alt sınıfıdır. Vector sınıfını extend eder. Last-in-First-Out mantığı ile çalışır. Yani son eklenen eleman Stack'in en tepesindedir. Vektor metotlarına ve yukarıdakilere benzer peek(), pop(), push() metotlarına ek olarak sahip olduğu önemli metotlar:

boolean empty(): Stack'in boş olup olmadığına bakar.

Örnek kod: Yukarıdaki Vector'ı Stack ile değiştirelim. Elimizde eleman olarak sırasıyla 0, 2, 3 var.

System.out.println(sayi.empty()); //Stack boş olmadığı için false döndü

int search(Object o): Objenin Stack'in tepesine uzaklığını dönerek o objenin Stack'teki yerini verir. Örneğin Stack sayi = new Stack() şeklinde tanımlanan Stack'e sırayla 1, 2 ve 3 sayılarını eklersek 3 en tepede olacak şekilde sırlanırlar ve search() metodu en tepedeki 3 için 1, 2 için 2 ve 1 için 3 döner.

2. Queue

2.1. Priority Queue: Nesnelerin bir önceliğe göre davranması gerektiğinde kullanılır. Queue interface'ini implement eder. Listler ile benzer metotlara ek olarak önemli metotları şunlardır:

comparator(): Queue'deki elemanları sıralamak için kullanılan comparator'u döner. Eğer elemanlar, o elemanların doğal davranışında sıralanmışsa null döner.

Örnek kod: PriorityQueue<Integer> = new PriorityQueue<>();

sayi.add(1);

sayi.add(2);

sayi.add(3);

Comparator comp = sayi.compaator() //Comparator'u import ettik.

System.out.println(comp); //Queue sıralı olduğundan null döndü.

2.2. ArrayDeque: Hem baştan hem sondan elemen eklemeye olanak veren Deque interface'ini implement eder. Deque interface'i hem First-in-First-Out hem de Last-in-First-Out olarak çalışabilir. ArrayDequelerin yukarıdaki metotlara ek önemli metotları şöyledir:

descendingIterator(): Elemanları sondan başa olarak döndürür.

3. Set

3.1. HashSet: Set interface'ini implement eder. Tüm Setler gibi HashSetlerde de bir elemandan en fazla bir tane bulunabilir. HashSet'e elemanlar eklendiğindei bu elemanlar eklenme sıralarına göre değil hash codelarına göre sıralanır. Önemli metotlarının hepsi yukarıda açıklanmıştır.

3.2. LinkedHashSet: HashSet'in düzgün sıralanmış versiyonudur. Önemli metotlarının hepsi yukarıdadır. Set interface'ini implement eder.

3.3. TreeSet: SortedSet ve Set interfacelerini implement eder. Elemanlar otomatik olarak kendi doğal sıralarına göre sıralanmıştır. Başlıca metotlarının hepsi yukarıdadır.

4. Map

4.1. HahsMap: Map interface'ini implement eder. Datalar key ve value ikilisi olarak tutar ve bu elemanlara erişim de buna uygun şekilde yapılır. Keyler benzersizken bir value'den birden fazla olabilir. Yukarıdaki benzerlerine ek olarak başlıca metotları şu şekildedir:

boolean containsKey(Object key): Map verilen key'i içeriyorsa true yoksa false döner.

Örnek kod: HashMap<Integer, String> map = new HashMap<>();

map.put(1, "A");

map.put(2, "B");

map.put(3, "C");

System.out.println(map.containsKey(3)); //Map, 3 key'ini içerdiğinden true döndü.

boolean containsValue(Object value): Map verilen value'yü içeriyorsa true yoksa false döner.

Örnek kod: System.out.println(map.containsValue("D")); //Map "D" value'sünü içermediğinden false döndü.

get(Object key): Verilen key'e karşılık gelen value'yü döner. Mevcut değilse false döner.

Örnek kod: System.out.println(map.get(1)); //1 key'ine karşılık gelen "A" value'sünü döndü.

keySet(): Map'teki keyleri Set olarak döner.

Örnek kod: System.out.println(map.keySet()); //[1,2,3] olarak keylerin Set'ini döndü.

put(key, value): Belirlenen (key, value) ikilisini map'e ekler. Eğer key zaten Map'te mevcutsa o key'e karşılık gelen value'yü eklenen value ile değiştirir.

Örnek kod: map.put(4, "D");

System.out.println(map.get(4)); //(4, "D") ikilisini ekledi ve ilgili value 4'ü döndü.

putAll(Map): Verilen Map'teki tüm ikilileri yeni Map'e taşır. Eğer keyler çakışıyorsa valueleri değiştirme işlemi yapar.

HashMap<Integer, String> map2 = new HashMap<>();

map.putAll(map2);

System.out.println(map2.keySet()); //map'teki ikilileri map2'ye ekledi ve map2'nin keylerini yazdırarak bunu gördük.

remove(Object key): Verilen key için (key, value) ikilisini siler.

Örnek kod: map.remove(4);
System.out.println(map.keySet());
System.out.println(map.values()); //4 ve karşılık gelen value'yü sildi ve [1,2,3] ile [A,B,C] listelerini döndü.

boolean remove(Object key, Object value): Verilen key, verilen value ile eşleşiyorsa (key, value) ikilisini siler.

replace(key, value): Verilen key'in bir value'sü varsa bu value'yü yenisiyle değiştirir.

boolean replace(key, oldValue, newValue): Verilen key verilen value ile eşleşiyorsa bu value'yü yenisiyle değiştirir.

Örnek kod: map.remove(3, "C");

map.remove(1, "D");

System.out.println(map.keySet());

System.out.println(map.values()); //3 ve "C" eşleştiği için o çifti sildi ama 1 ve "D" eşleşmediği için işlem yapmadı.

values(): Map'in içerdiği valueleri Collection olarak döner.

4.2. LinkedHashMap: Map interface'ini implement eder. HashMap ile oldukça benzerdir. Yalnızca (key, value) ikililerinin Map'e eklenme sırasını korur. Metotları HashMap ile aynıdır.

4.3. Hashtable: Map interface'ini implement eder. HashMap ve LinkedHashMap gibi (key, value) ikililerinden oluşur. Hashtablelarda herhangi bir obje key ya da value olarak kullanılabilir. Yukarıdakilerden farklı olarak önemli metotları şunlardır:

elements(): Hashtable'daki valuelerin bir numaralandırmasını verir. Bu özellik HashMaplerde yoktur.

4.4. TreeMap: Map, sahip olduğu keylerin doğal sıralanmasına göre ya da belirli bir karşılaştırıcıya göre sıralıdır. SortedMap interface'ini implement eder. Diğer Maplerden farklı başlıca metotları şöyledir:

ceilingEntry(key): Verilen key'den büyük ya da ona eşit en küçük key için (key, value) ikilisini döner. Öyle bir key yoksa null döner.

Örnek kod: Yukarıdaki HashMap'i TreeMap'e çevirip sildiğimiz (3, "C") ikilisini tekrar ekleyelim.

```
map.put(3, "C");

System.out.println(map.ceilingEntry(2)); //2, Map'te bulunduğu için 2=B döndü ancak 2 olmasaydı 3=C dönerdi.
```

ceilingKey(key): Verilen key'den büyük ya da ona eşit en küçük key'i döner. Eğer öyle bir key yoksa null döner.

Örnek kod: System.out.println(map.ceilingKey(2)); //2 Map'te bulunduğu için 2'yi döndü ama bulunmasaydı 3 dönerdi.

floorEntry() ve floorKey() metotları ise benzer işlemi verilen key'den küçük ya da ona eşit en büyük key için yapar.

comparator(): Map'teki keyleri sıralamak için kullanılan karşılaştırıcıyı döner. Eğer keyler doğal olarak sıralıysa null döner.

entrySet(): (key, value) ikililerini Set olarak döner.

Örnek kod: System.out.println(map.entrySet()); //(key, value) ikililerini [1=A, 2=B, 3=C] şeklinde döndü.

firstEntry(): En küçük key için (key, value) ikilisini döner.

Örnek kod: System.out.println(map.firstEntry()); //1=A şeklinde döndü.

firstKey(): Map'teki en küçük key'i döner.

Örnek kod: System.out.println(map.firstKey()); //1 şeklinde döndü.

lastEntry(): En büyük key için (key, value) ikilisini döner.

Örnek kod: System.out.println(map.lastEntry()); //3=C şeklinde döndü.

lastKey(): En büyük key'i döner.

Örnek kod: System.out.println(map.lastKey()); //3 şeklinde döndü.

pollFirstEntry(): En düşük key için (key, value) ikilisini döner ve ikiliyi siler. Map boşsa null döner.

Örnek kod:
```
System.out.println(map.pollFirstEntry());

System.out.println(map.keySet());
```

System.out.println(map.values()); //1 key'ine karşılık gelen çifti sildi ve [2,3] ile [B,C] şeklinde döndü.

pollLastEntry(): En büyük key için (key, value) ikilisini döner ve ikiliyi siler. Map boşsa null döner.

Örnek kod: System.out.println(map.pollLastEntry());

System.out.println(map.keySet());

System.out.println(map.values()); //3 key'ine karşılık gelen çifti sildi ve [2] ile [B] şeklinde döndü.