

ARRAYLIST

Hazırlayan

Zeyneb Eda YILMAZ

İçindekiler

1. ArrayList.....	3
1.1. Java ArrayList Fonksiyonları	3
1.1.1. add() Fonksiyonu:	3
1.1.2. addAll() Fonksiyonu:	4
1.1.3. clear() Fonksiyonu:.....	4
1.1.4. clone() Fonksiyonu:	4
1.1.5. get() Fonksiyonu:	5
1.1.6. indexOf() Fonksiyonu:	5
1.1.7. remove() Fonksiyonu:	5
1.1.8. removeAll() Fonksiyonu:	6
1.1.9. set() Fonksiyonu:.....	6
1.1.10. size() Fonksiyonu:.....	7
1.1.11. subList() Fonksiyonu:.....	7
1.1.12. toArray() Fonksiyonu:.....	7
1.1.13. contains() Fonksiyonu:	8
1.1.14. isEmpty() Fonksiyonu:	8
1.1.15. lastIndexOf() Fonksiyonu:.....	9
1.1.16. retainAll() Fonksiyonu:	9
2. Kaynakça	10

1. ArrayList

Java ArrayList sınıfı, öğeleri depolamak amacıyla kullanılan dinamik sınıftır. “java.util” paketinde bulunur ve Java Collections Framework’unun bir parçasıdır. ArrayList, AbstractList soyut sınıfından miras alır ve List interface’ini uygular.

ArrayList dinamik bir sınıf olduğu için, tanımlanırken belirli bir boyutla tanımlanmış olsa dahi, nesneler eklendiğinde boyutu büyür veya nesneler çıkarıldığında boyutu küçülür.

ArrayList tanımlanırken, int, double, char gibi primitif tipteki veriler kullanılamaz. Primitif tiplerin sarmalayıcı sınıfları kullanılır. Sarmalayıcı sınıflara; int için Integer, double için Double, char için Character sınıfları örnek olarak verilebilir. Sarmalayıcı sınıflarla tanımlanmış olan ArrayList’lere, primitif tipteki nesneler eklenebilir.

```
1 import java.util.*;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         ArrayList<Integer> integerList = new ArrayList<Integer>();
8         integerList.add(500);
9
10        ArrayList<Double> doubleList = new ArrayList<Double>();
11        doubleList.add(500.8);
12
13        ArrayList<Character> characterList = new ArrayList<Character>();
14        characterList.add('Z');
15    }
16 }
```

Şekil 1: Sarmalayıcı sınıflar kullanarak ArrayList tanımlama ve primitif tipte nesne ekleme

1.1. Java ArrayList Fonksiyonları

1.1.1. add() Fonksiyonu:

- ❖ Bu yöntem, listeye eleman eklemek için kullanılır.
- ❖ İstenilen nesne, listenin sonuna eklenir.

```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(500);
integerList.add(600);
integerList.add(700);
integerList.add(800);
integerList.add(900);
```

Şekil 2: add() fonksiyonunun kullanım örneği

```
[500, 600, 700, 800, 900]
```

Şekil 3: Şekil 2'deki kodun çıktısı

1.1.2. addAll() Fonksiyonu:

- ❖ Bu yöntem listeye birden çok eleman eklemek için kullanılır.
- ❖ Bu yöntemde parametre olarak bir koleksiyon verilmelidir.

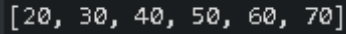
```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(20);
integerList.add(30);
integerList.add(40);

ArrayList<Integer> integerList2 = new ArrayList<Integer>();
integerList2.add(50);
integerList2.add(60);
integerList2.add(70);

integerList.addAll(integerList2);

System.out.println(integerList);
```

Şekil 4: addAll() fonksiyonunun kullanım örneği



[20, 30, 40, 50, 60, 70]

Şekil 5: Şekil 4'deki kodun çıktısı

1.1.3. clear() Fonksiyonu:

- ❖ Listedeki tüm nesnelerin silinmesi amacıyla kullanılır.

```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(500);
integerList.add(600);
integerList.add(700);
integerList.add(800);
integerList.add(900);

integerList.clear();
```

Şekil 6: clear() fonksiyonunun kullanım örneği

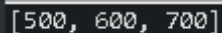
1.1.4. clone() Fonksiyonu:

- ❖ Belirtilen listenin bir kopyasını oluşturmak amacıyla kullanılır.

```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(500);
integerList.add(600);
integerList.add(700);

ArrayList<Integer> integerListClone = (ArrayList<Integer>) integerList.clone();
System.out.println(integerListClone);
```

Şekil 7: clone() fonksiyonunun kullanım örneği



[500, 600, 700]

Şekil 8: Şekil 7'deki kodun çıktısı

1.1.5. get() Fonksiyonu:

- ❖ ArrayList içerisindeki istenilen nesneyi çağırmak amacıyla kullanılır.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Zeyneb");
names.add("Eda");
names.add("Yılmaz");
names.add("Engin");
names.add("Demiroğ");
System.out.println("Üçüncü indexteki nesne: "+names.get(3));
```

Şekil 9: get() fonksiyonunun kullanım örneği

Üçüncü indexteki nesne: Engin

Şekil 10: Şekil 9'daki kodun çıktısı

1.1.6. indexOf() Fonksiyonu:

- ❖ ArrayList içerisinde, istenilen nesnenin index numarasını bulmak amacıyla kullanılır.
- ❖ İstenilen nesne, liste içerisinde yoksa -1 değeri döndürür.

```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(500);
integerList.add(600);
integerList.add(700);
integerList.add(800);

System.out.println(integerList.indexOf(800));
System.out.println(integerList.indexOf(900));
```

Şekil 11: indexOf() fonksiyonunun kullanım örneği

3
-1

Şekil 12: Şekil 11'deki kodun çıktısı

1.1.7. remove() Fonksiyonu:

- ❖ ArrayList içerisindeki istenilen nesnenin, listeden silinmesi amacıyla kullanılır.
- ❖ Parametre olarak istenilen nesnenin index numarası verilir.
- ❖ İstenilen nesnenin index numarası bilinmiyorsa indexOf() fonksiyonu kullanılabilir.

```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(300);
integerList.add(400);
integerList.add(500);
integerList.add(600);
integerList.add(700);
integerList.add(800);

integerList.remove(2);
integerList.remove(integerList.indexOf(700));
System.out.println("Liste: "+integerList);
```

Şekil 13:remove() fonksiyonunun kullanım örneği

Liste: [300, 400, 600, 800]

Şekil 14: Şekil 13'deki kodun çıktısı

1.1.8. removeAll() Fonksiyonu:

- ❖ ArrayList içerisindeki nesnelerden birden fazlasını aynı anda silmek amacıyla kullanılır.
- ❖ Parametre olarak bir koleksiyon verilmelidir.

```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(300);
integerList.add(400);
integerList.add(500);
integerList.add(600);
integerList.add(700);
integerList.add(800);

ArrayList<Integer> removeList = new ArrayList<Integer>();
removeList.add(400);
removeList.add(500);

integerList.removeAll(removeList);
System.out.println(integerList);
```

Şekil 15: removeAll() fonksiyonunun kullanım örneği

[300, 600, 700, 800]

Şekil 16: Şekil 15'deki kodun çıktısı

1.1.9. set() Fonksiyonu:

- ❖ ArrayList içerisinde, istenilen konuma istenilen nesneyi eklemek amacıyla kullanılır.
- ❖ Parametre olarak bir tane tam sayı ve bir tane nesne alır.

```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(300);
integerList.add(400);
integerList.add(500);
integerList.add(600);

integerList.set(2, 900);
System.out.println(integerList);
```

Şekil 17: set() fonksiyonunun kullanım örneği

[300, 400, 900, 600]

Şekil 18: Şekil 17'deki kodun çıktısı

1.1.10.size() Fonksiyonu:

- ❖ ArrayList içerisindeki nesnelerin sayısını bulmak amacıyla kullanılır.

```
ArrayList<Integer> integerList = new ArrayList<Integer>();
integerList.add(300);
integerList.add(400);
integerList.add(500);
integerList.add(600);

System.out.println("Liste içerisindeki nesne sayısı: "+integerList.size());
```

Şekil 19: size() fonksiyonunun kullanım örneği

```
Liste içerisindeki nesne sayısı: 4
```

Şekil 20: Şekil 19'daki kodun çıktısı

1.1.11.subList() Fonksiyonu:

- ❖ Parametrede istenilen sayı aralığındaki nesneleri arrayList içerisinde bulup, bir alt listeye atamak amacıyla kullanılır.
- ❖ Parametre olarak iki tane tam sayı değeri alır.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Zeyneb");
names.add("Eda");
names.add("Yılmaz");
names.add("Engin");
names.add("Demiroğ");

List<String> namesSubList = names.subList(1, 4);

System.out.println(namesSubList);
```

Şekil 21: subList() fonksiyonunun kullanım örneği

```
[Eda, Yılmaz, Engin]
```

Şekil 22: Şekil 21'deki kodun çıktısı

1.1.12.toArray() Fonksiyonu:

- ❖ ArrayList'i diziye çevirmek amacıyla kullanılır.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Zeyneb");
names.add("Eda");
names.add("Yılmaz");
names.add("Engin");
names.add("Demiroğ");
System.out.println("ArrayList: " + names);
Object[] nameArray = names.toArray();
for (Object name : nameArray) {
    System.out.println("Array: " + name);
}
```

Şekil 23: toArray() fonksiyonunun kullanım örneği

```
ArrayList: [Zeyneb, Eda, Yılmaz, Engin, Demiroğ]
Array: Zeyneb
Array: Eda
Array: Yılmaz
Array: Engin
Array: Demiroğ
```

Şekil 24: Şekil 23'deki kodun çıktısı

1.1.13.contains() Fonksiyonu:

- ❖ Parametrede istenilen nesnenin, ArrayList içerisinde bulunup bulunmadığını kontrol etmek amacıyla kullanılır.
- ❖ Nesne, ArrayList içerisinde varsa true değeri, yoksa false değeri döndürür.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Zeyneb");
names.add("Eda");
names.add("Yılmaz");
names.add("Engin");
names.add("Demiroğ");
System.out.println(names.contains("Eda"));
```

Şekil 25: contains() fonksiyonunun kullanım örneği

```
true
```

Şekil 26: Şekil 25'deki kodun çıktısı

1.1.14.isEmpty() Fonksiyonu:

- ❖ ArrayList'in boş olup olmadığını kontrol etmek amacıyla kullanılır.
- ❖ ArrayList boş ise true değeri, boş değil ise false değeri döndürür.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Zeyneb");
names.add("Eda");
names.add("Yılmaz");
names.add("Engin");
names.add("Demiroğ");
ArrayList<String> cities = new ArrayList<String>();

System.out.println("Names listesi: " + names.isEmpty());
System.out.println("Cities listesi: " + cities.isEmpty());
```

Şekil 27: isEmpty() fonksiyonunun kullanım örneği

```
Names listesi: false
Cities listesi: true
```

Şekil 28: Şekil 27'deki kodun çıktısı

1.1.15.lastIndexOf() Fonksiyonu:

- ❖ ArrayList içerisinde, parametrede istenilen nesnenin son index numarasını bulmak amacıyla kullanılır.
- ❖ ArrayList içerisinde istenilen nesneden birden çok sayıda varsa en son sıradaki nesnenin index numarasını döndürür.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Zeyneb");
names.add("Eda");
names.add("Yılmaz");
names.add("Engin");
names.add("Demiroğ");
names.add("Zeyneb");
names.add("Engin");

System.out.println("Zeyneb: "+names.lastIndexOf("Zeyneb"));
System.out.println("Engin: "+names.lastIndexOf("Engin"));
```

Şekil 29: lastIndexOf() fonksiyonunun kullanım örneği

```
Zeyneb: 5
Engin: 6
```

Şekil 30: Şekil 29'daki kodun çıktısı

1.1.16.retainAll() Fonksiyonu:

- ❖ ArrayList içerisinde, parametrede belirtilen ArrayList içerisindeki nesnelerden eşleşenleri tutar, eşleşmeyenleri siler.
- ❖ Parametre olarak bir ArrayList alır.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Zeyneb");
names.add("Eda");
names.add("Yılmaz");
names.add("Engin");
names.add("Demiroğ");

ArrayList<String> firstNames = new ArrayList<String>();
firstNames.add("Zeyneb");
firstNames.add("Eda");
firstNames.add("Engin");

names.retainAll(firstNames);
System.out.println(names);
```

Şekil 31: retainAll() fonksiyonunun kullanım örneği

```
[Zeyneb, Eda, Engin]
```

Şekil 32: Şekil 31'deki kodun çıktısı

2. Kaynakça

- ❖ <https://www.tasarimkodlama.com/java-programlama/java-arraylist-metotlari/>
- ❖ <https://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/dataStructures/Collections/ClassArrayList01.pdf>
- ❖ <https://medium.com/@denizf.b/java-collections-arraylist-nedir-7519bc1b7654>