

MTK467 Nesneye Yönelik Programlama

Hafta 14 - Generics2

Zümra Kavafoğlu
<https://zumrakavafoglu.github.io/>

MyMaptest2: Address sınıfı

```
public class Address {  
  
    private int houseNumber;  
    private String street;  
    private String city;  
    private String country;  
  
    Address(int houseNumber, String street, String city, String country){  
        this.houseNumber = houseNumber;  
        this.street = street;  
        this.city = city;  
        this.country = country;  
    }  
  
    public int getHouseNumber() {  
        return houseNumber;  
    }  
  
    public void setHouseNumber(int houseNumber) {  
        this.houseNumber = houseNumber;  
    }  
  
    public String getStreet() {  
        return street;  
    }  
  
    public void setStreet(String street) {  
        this.street = street;  
    }  
}
```

MyMaptest2: Address sınıfı (devamı)

```
public String getCity() {  
    return city;  
}  
  
public void setCity(String city) {  
    this.city = city;  
}  
  
public String getCountry() {  
    return country;  
}  
  
public void setCountry(String country) {  
    this.country = country;  
}  
  
@Override  
public String toString() {  
    return street + " No: " + houseNumber + " - " + city + " / " + country ;  
}
```

MyMaptest2: AddressPhoneMap

```
public class MyMapTest2 {  
    public static void main(String[] args){  
        MyMap<Address, Long> addressPhoneMap = new MyMap<Address, Long>();  
  
        addressPhoneMap.put(new Address(13, "Sumak Sk.", "Ankara", "Turkiye"), 3122112321L);  
        addressPhoneMap.put(new Address(2, "Moda Sk.", "Istanbul", "Turkiye"), 2123458989L);  
        addressPhoneMap.put(new Address(13, "Sumak Sk.", "Ankara", "Turkiye"), 3126572321L);  
  
        System.out.println(addressPhoneMap);  
    }  
}
```

MyMaptest2: AddressPhoneMap

```
public class MyMapTest2 {  
  
    public static void main(String[] args){  
  
        MyMap<Address, Long> addressPhoneMap = new MyMap<Address, Long>();  
  
        addressPhoneMap.put(new Address(13, "Sumak Sk.", "Ankara", "Türkiye"), 3122112321L);  
        addressPhoneMap.put(new Address(2, "Moda Sk.", "İstanbul", "Türkiye"), 2123458989L);  
        addressPhoneMap.put(new Address(13, "Sumak Sk.", "Ankara", "Türkiye"), 3126572321L);  
  
        System.out.println(addressPhoneMap);  
    }  
}
```



Aynı anahtar değerli iki girdi map'e eklenirse, bu anahtara karşılık gelen değer güncellenir.

MyMaptest2: AddressPhoneMap

```
public class MyMapTest2 {  
  
    public static void main(String[] args){  
  
        MyMap<Address, Long> addressPhoneMap = new MyMap<Address, Long>();  
  
        addressPhoneMap.put(new Address(13, "Sumak Sk.", "Ankara", "Turkiye"), 3122112321L);  
        addressPhoneMap.put(new Address(2, "Moda Sk.", "Istanbul", "Turkiye"), 2123458989L);  
        addressPhoneMap.put(new Address(13, "Sumak Sk.", "Ankara", "Turkiye"), 3126572321L);  
  
        System.out.println(addressPhoneMap);  
  
    }  
}
```

Ama çıktıda görüldüğü gibi map iki anahtarı farklı kabul etmiş, dolayısıyla mapde aynı anahtara sahip iki girdi var.

Aynı anahtar değerli iki girdi map'e eklenirse, bu anahtara karşılık gelen değer güncellenir.

Çıktı

Map:

[Sumak Sk. No: 13 – Ankara / Turkiye, 3122112321]
[Moda Sk. No: 2 – Istanbul / Turkiye, 2123458989]
[Sumak Sk. No: 13 – Ankara / Turkiye, 3126572321]

MyMaptest2: AddressPhoneMap

```
public void put(T key, S value){  
    if(this.isEmpty()){  
        firstPair = new Pair<T, S>(key, value);  
    }  
    else{  
        Pair<T, S> pair = this.getPair(key);  
  
        if(pair != null){  
            pair.setValue(value);  
        }else{  
            this.getLast().setNext(new Pair<T, S>(key, value));  
        }  
    }  
}
```

MyMaptest2: AddressPhoneMap

```
private Pair<T, S> getPair(T key){  
    if(key == null){  
        System.out.println("Parameter key is null");  
        return null;  
    }  
  
    Pair<T, S> pair = firstPair;  
    while (pair != null){  
        if(pair.getKey().equals(key)){  
            return pair;  
        }  
  
        pair = pair.getNext();  
    }  
  
    return null;  
}
```


Overriding equals method

Bir sınıfın equals yöntemi override edilmezse == ile aynı biçimde çalışır yani nesnenin içeriğini değil hafızadaki yerini karşılaştırır.

```
public class TestEquality1 {  
    public static void main(String[] args){  
        MyMap<Address, Long> addressPhoneMap = new MyMap<Address, Long>();  
  
        Address address1 = new Address(13, "Sumak Sk.", "Ankara", "Türkiye");  
        Address address2 = new Address(13, "Sumak Sk.", "Ankara", "Türkiye");  
  
        System.out.println("Adress1: "+address1);  
        System.out.println("Adress2: "+address2);  
  
        if(address1.equals(address2))  
            System.out.println("Adress1 is same with address2");  
        else  
            System.out.println("Adress2 is not same with address2");  
    }  
}
```

```
Adress1: Sumak Sk. No: 13 – Ankara / Türkiye  
Adress2: Sumak Sk. No: 13 – Ankara / Türkiye  
Adress2 is not same with address2
```

Overriding equals method

- equals metodunun içeriği aynı iki Adress nesnesi için true döndürmesi için, Address sınıfının equals metodunu override etmeliyiz.
- Ancak bir kural olarak, bir sınıfın equals metodunu override ettiğimiz her zaman o sınıfın hashCode() metodunu da override etmeliyiz.
- hashcode javanın her bir nesne için ürettiği bir tamsayıdır. hashCode() metodunu override etmek için Objects sınıfının hash metodu kullanılabilir.

Overriding equals method

```
@Override
public boolean equals(Object o) {
    if (this == o)
        return true;
    if (o == null || getClass() != o.getClass())
        return false;

    Address address = (Address) o;

    if (houseNumber != address.houseNumber)
        return false;
    if (!street.equals(address.street))
        return false;
    if (!city.equals(address.city))
        return false;
    if (!country.equals(address.country))
        return false;

    return true;
}

@Override
public int hashCode() {
    return Objects.hash(houseNumber, street, city, country);
}
```

Overriding equals method

- Address sınıfının equals ve hashCode metotlarının override edilmesiyle istediğimiz sonuca ulaşırız.

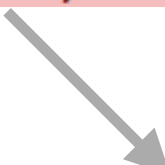
```
public class TestEquality1 {  
    public static void main(String[] args){  
        MyMap<Address, Long> addressPhoneMap = new MyMap<Address, Long>();  
  
        Address address1 = new Address(13, "Sumak Sk.", "Ankara", "Turkiye");  
        Address address2 = new Address(13, "Sumak Sk.", "Ankara", "Turkiye");  
  
        System.out.println("Adress1: "+address1);  
        System.out.println("Adress2: "+address2);  
  
        if(address1.equals(address2))  
            System.out.println("Adress1 is same with address2");  
        else  
            System.out.println("Adress2 is not same with address2");  
    }  
}
```

```
Adress1: Sumak Sk. No: 13 - Ankara / Turkiye  
Adress2: Sumak Sk. No: 13 - Ankara / Turkiye  
Adress1 is same with address2
```

Overriding equals method

- Address sınıfının equals ve hashCode metotlarının override edilmesiyle istediğimiz sonuca ulaşırız.

```
public class MyMapTest2 {  
  
    public static void main(String[] args){  
  
        MyMap<Address, Long> addressPhoneMap = new MyMap<Address, Long>();  
  
        addressPhoneMap.put(new Address(13, "Sumak Sk.", "Ankara", "Türkiye"), 3122112321L);  
        addressPhoneMap.put(new Address(2, "Moda Sk.", "İstanbul", "Türkiye"), 2123458989L);  
        addressPhoneMap.put(new Address(13, "Sumak Sk.", "Ankara", "Türkiye"), 3126572321L);  
  
        System.out.println(addressPhoneMap);  
    }  
}
```



Aynı anahtar değerli iki girdi map'e eklenirse, bu anahtara karşılık gelen değer güncellenir.

Map:

```
[ Sumak Sk. No: 13 – Ankara / Türkiye, 3126572321 ]  
[ Moda Sk. No: 2 – İstanbul / Türkiye, 2123458989 ]
```

Generic Sınıflar: Ham tipler (Raw types)

- Herhangi bir tip argümanı belirlemeden bir generic sınıf nesnesi oluşturmayı sağlar.

```
Stack objectStack = new Stack( 5 );
```

- Tip argümanı belirli bir Generic sınıf nesnesi, ham bir generic sınıf nesnesine atanabilir.

```
Stack rawTypeStack2 = new Stack<Double>(5);
```

- Tam tersi de yapılabilir ama güvenli değildir. Örneğin aşağıdaki örnekte sağ tarafta oluşturulan Stack tipi Integer olmayan elemanlar da içerebilir.

```
Stack<Integer> integerStack = new Stack(10);
```


Generic Sınıflar: Ham tipler (Raw types)

```
public class RawTypeTest {  
    public static void main(String[] args) {  
        Double[] doubleElements = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6};  
        Integer[] integerElements = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};  
  
        Stack rawTypeStack1 = new Stack(5);  
  
        Stack rawTypeStack2 = new Stack<Double>(5);  
  
        Stack<Integer> integerStack = new Stack(10); // stack stores Integer objects  
  
        StackTestWithGenericMethod.pushArray(rawTypeStack1, doubleElements);  
        StackTestWithGenericMethod.pushArray(rawTypeStack2, doubleElements);  
        StackTestWithGenericMethod.pushArray(integerStack, integerElements);  
    }  
}
```

Generic Arayüz

```
public interface IComparable {  
    public int compareTo(Object o);  
}
```

Generic olmayan IComparable arayüzü

Circle IComparable arayüzünü implement eder

```
public class Circle extends GeometricObject implements IComparable
```

Circle sınıfının compareTo metodu

```
@Override  
public int compareTo(Object o) {  
    if (getRadius() > ((Circle) o).getRadius())  
        return 1;  
    else if (getRadius() < ((Circle) o).getRadius())  
        return -1;  
    else return 0;  
}
```

```
public class ComparableTest {  
    public static void main(String[] args){  
        Circle c1 = new Circle(3.0);  
        Rectangle r1 = new Rectangle(2,3);  
        System.out.print(c1.compareTo(r1));  
    }  
}
```

```
Exception in thread "main" java.lang.ClassCastException: Rectangle cannot be cast to Circle  
    at Circle.compareTo(Circle.java:49)  
    at ComparableTest.main(ComparableTest.java:11)
```

Rectangle nesnesi Circle sınıfına cast edilemeyeceğinden çalışma zamanında hata verir.

Generic Arayüz

```
public interface GenericComparable <T> {  
    public int compareTo(T other);  
}
```

T tip parametrelili Generic arayüz

Circle2, GenericComparable<Circle2> arayüzünü implement eder

```
public class Circle2 extends GeometricObject implements GenericComparable<Circle2>
```

```
@Override  
public int compareTo(Circle2 other) {  
    if (getRadius() > other.getRadius())  
        return 1;  
    else if (getRadius() < other.getRadius())  
        return -1;  
    else  
        return 0;  
}
```

Circle2 sınıfının compareTo metodu

```
public class ComparableTest2 {  
    public static void main(String[] args){  
        Circle2 c2 = new Circle2(3.0);  
        Rectangle r1 = new Rectangle(2,3);  
        System.out.print(c2.compareTo(r1));  
    }  
}
```

compareTo (Circle2) in Circle2 cannot be applied
to (Rectangle)

type-safety: Yanlış tipte argüman
derleme zamanı hatası verir

Tip parametreleri için üst sınır

- Tip parametrelerinin temsil edeceği değişken tiplerini kısıtlamak için bu parametrelere bir üst sınır konulabilir. Burada üst sınırdan kasıt bir süper sınıf veya arayüzdür. Eğer herhangi bir üst sınır belirtilmediyse üst sınır Object sınıfıdır.
- Örneğin tip parametresi T'nin yalnızca SClass süper sınıfının çocuklarını temsil etmesi isteniyorsa o zaman tip parametresi kısmına
< T extends SClass >
yazılır. Eğer SClass bir arayüzse ve T'nin yalnızca SClass arayüzünü implement eden sınıfları temsil etmesi isteniyorsa yine aynı gösterim kullanılır.
- Eğer T parametresi A sınıfının alt sınıfı olacak ve B ve C arayüzlerini implement edecek biçimde kısıtlanmak isteniyorsa
< T extends A & B & C >
yazılır. Burada A'nın ilk yazılmaması derleme-zamanı hatasına sebep olur.

Tip parametreleri için üst sınır

Comparable<T> Java'nın sağladığı bir arayüzdür. Bu arayüzü implement eden sınıfların **int compareTo** metodunu implement etmesi gerekir.

maximum isimli Generic metodun yalnızca **Comparable<T>** arayüzünü implement eden tiplere sahip değişkenlerle çağırılabilmesini sağlar

```
public class MaximumTest {  
    // determines the largest of three Comparable objects  
    public static < T extends Comparable< T > > T maximum( T x, T y, T z )  
    {  
        T max = x; // assume x is initially the largest  
        if ( y.compareTo( max ) > 0 )  
            max = y; // y is the largest so far  
        if ( z.compareTo( max ) > 0 )  
            max = z; // z is the largest  
        return max; // returns the largest object  
    } // end method maximum  
  
    public static void main( String args[] ){  
        System.out.printf( "Maximum of %d, %d and %d is %d\n\n", 3, 4, 5, maximum(3,4,5));  
        System.out.printf( "Maximum of %.1f, %.1f and %.1f is %.1f\n\n", 6.6, 8.8, 7.7, maximum(6.6, 7.7, 8.8));  
        System.out.printf( "Maximum of %s, %s and %s is %s\n", "pear","apple", "orange", maximum("pear","apple", "orange"));  
    }  
}
```

Böylece her argüman nesnenin **compareTo** metodunun çağırılabilmesi garantilenmiş olur.

Arrays sınıfı

- Arrays sınıfı dizileri işlemek için static metotlar sağlayan bir Java.util sınıfıdır.
- Arrays sınıfı metotları:
 - sort : diziyi sıralar
 - binarySearch : sıralanmış bir dizide arama yapar
 - equals: dizileri karşılaştırır
 - fill: diziyi elemanlarla doldurur.

Arrays örneği

```
3  import java.util.Arrays;
4
5  public class UsingArrays
6  {
7      private int intArray[] = { 1, 2, 3, 4, 5, 6 };
8      private double doubleArray[] = { 8.4, 9.3, 0.2, 7.9, 3.4 };
9      private int filledIntArray[], intArrayCopy[];
10
11     // constructor initializes arrays
12     public UsingArrays()
13     {
14         filledIntArray = new int [ 10 ]; // create int array with 10 elements
15         intArrayCopy = new int [ intArray.length ];
16
17         Arrays.fill( filledIntArray, 7 ); // fill with 7s
18         Arrays.sort( doubleArray ); // sort doubleArray ascending
19
20         // copy array intArray into array intArrayCopy
21         System.arraycopy( intArray, 0, intArrayCopy,
22             0, intArray.length );
23     } // end UsingArrays constructor
```

Arrays örneği

```
47      // find value in array intArray
48      public int searchForInt( int value )
49      {
50          return Arrays.binarySearch( intArray, value );
51      } // end method searchForInt
52
53      // compare array contents
54      public void printEquality()
55      {
56          boolean b = Arrays.equals( intArray, intArrayCopy );
57          System.out.printf( "intArray %s intArrayCopy\n",
58                          ( b ? "==" : "!=" ) );
59
60          b = Arrays.equals( intArray, filledIntArray );
61          System.out.printf( "intArray %s filledIntArray\n",
62                          ( b ? "==" : "!=" ) );
63      } // end method printEquality
```


Arrays örneği

```
26     public void printArrays()
27     {
28         System.out.print( "doubleArray: " );
29         for ( double doubleValue : doubleArray )
30             System.out.printf( "%.1f ", doubleValue );
31
32         System.out.print( "\nintArray: " );
33         for ( int intValue : intArray )
34             System.out.printf( "%d ", intValue );
35
36         System.out.print( "\nfilledIntArray: " );
37         for ( int intValue : filledIntArray )
38             System.out.printf( "%d ", intValue );
39
40         System.out.print( "\nintArrayCopy: " );
41         for ( int intValue : intArrayCopy )
42             System.out.printf( "%d ", intValue );
43
44         System.out.println( "\n" );
45     } // end method printArrays
```

Arrays örneği

```
65     public static void main( String args[] )
66     {
67         UsingArrays usingArrays = new UsingArrays();
68
69         usingArrays.printArrays();
70         usingArrays.printEquality();
71
72         int location = usingArrays.searchForInt( 5 );
73         if ( location >= 0 )
74             System.out.printf(
75                 "Found 5 at element %d in intArray\n", location );
76         else
77             System.out.println( "5 not found in intArray" );
78
79         location = usingArrays.searchForInt( 8763 );
80         if ( location >= 0 )
81             System.out.printf(
82                 "Found 8763 at element %d in intArray\n", location );
83         else
84             System.out.println( "8763 not found in intArray" );
85     } // end main
86 } // end class UsingArrays
```