Factory Methods for creating unmodifiable Collections

List: An indexed Collection of elements where duplicates are allowed and insertion order is preserved.

Set: An unordered Collection of elements where duplicates are not allowed and insertion order is not preserved.

Map: A Map is a collection of key-value pairs and each key-value pair is called Entry.Entry is an inner interface present inside Map interface.Duplicate keys are not allowed,but values can be duplicated.

As the part of programming requirement, it is very common to use Immutable Collection objects to improve Memory utilization and performance.

Prior to Java 9,we can create unmodifiable Collection objects as follows

Eg 1: Creation of unmodifiable List object

```
1)  List<String> beers=new ArrayList<String>();
2)  beers.add("KF");
3)  beers.add("FO");
4)  beers.add("RC");
5)  beers.add("FO");
6)  beers =Collections.unmodifiableList(beers);
```

Eg 2: Creation of unmodifiable Set Object

```
1)  Set<String> beers=new HashSet<String>();
2)  beers.add("KF");
3)  beers.add("KO");
4)  beers.add("RC");
5)  beers.add("FO");
6)  beers =Collections.unmodifiableSet(beers);
```

Eg 3: Creation of unmodifiable Map object

```
1)  Map<String,String> map=new HashMap<String,String>();
2)  map.put("A","Apple");
3)  map.put("B","Banana");
4)  map.put("C","Cat");
5)  map.put("D","Dog");
6)  map =Collections.unmodifiableMap(map);
```

This way of creating unmodifiable Collections is verbose and not convenient.It increases length of the code and reduces readability.

JDK Engineers addresses this problem and introduced several factory methods for creating unmodifiable collections.

Creation of unmodifiable List (Immutable List) with Java 9 Factory Methods:

Java 9 List interface defines several factory methods for this.

1. static <E> List<E> of()
2. static <E> List<E> of(E e1)
3. static <E> List<E> of(E e1, E e2)
4. static <E> List<E> of(E e1, E e2, E e3)
5. static <E> List<E> of(E e1, E e2, E e3, E e4)
6. static <E> List<E> of(E e1, E e2, E e3, E e4, E e5)
7. static <E> List<E> of(E e1, E e2, E e3, E e4, E e5, E e6)
8. static <E> List<E> of(E e1, E e2, E e3, E e4, E e5, E e6, E e7)
9. static <E> List<E> of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8)
10. static <E> List<E> of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9)
11. static <E> List<E> of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9, E e10)
12. static <E> List<E> of(E... elements)

Upto 10 elements the matched method will be executed and for more than 10 elements internally var-arg method will be called.JDK Enginerrs identified List of upto 10 elements is the common requirement and hence they provided the corresponding methods. For remaining cases var-arg method will be executed,which is very costly. These many methods just to improve performance.

Eg: To create unmodifiable List with Java 9 Factory Methods.

```
List<String> beers = List.of("KF","KO","RC","FO");
```

It is very simple and straight forward way.

Note:

1. While using these factory methods if any element is null then we will get NullPointerException.

```
List<String> fruits = List.of("Apple","Banana",null);
```
➔ NullPointerException

2. After creating the List object,if we are trying to change the content(add|remove|replace elements)then we will get UnsupportedOperationException b'z List is immutable(unmodifiable).

```
List<String> fruits=List.of("Apple","Banana","Mango");
fruits.add("Orange");    //UnsupportedOperationException
fruits.remove(1);       //UnsupportedOperationException
fruits.set(1,"Orange"); //UnsupportedOperationException
```

Creation of unmodifiable Set(Immutable Set) with Java 9 Factory Methods:

Java 9 Set interface defines several factory methods for this.

1. static <E> Set<E> of()
2. static <E> Set<E> of(E e1)
3. static <E> Set<E> of(E e1, E e2)
4. static <E> Set<E> of(E e1, E e2, E e3)
5. static <E> Set<E> of(E e1, E e2, E e3, E e4)
6. static <E> Set<E> of(E e1, E e2, E e3, E e4, E e5)
7. static <E> Set<E> of(E e1, E e2, E e3, E e4, E e5, E e6)
8. static <E> Set<E> of(E e1, E e2, E e3, E e4, E e5, E e6, E e7)
9. static <E> Set<E> of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8)
10. static <E> Set<E> of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9)
11. static <E> Set<E> of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9, E e10)
12. static <E> Set<E> of(E... elements)

Eg: To create unmodifiable Set with Java 9 Factory Methods.

```
Set<String> beers = Set.of("KF","KO","RC","FO");
```

Note:

1. While using these Factory Methods if we are trying to add duplicate elements then we will get IllegalArgumentException,b'z Set won't allow duplicate elements

```
Set<Integer> numbers=Set.of(10,20,30,10);
RE: IllegalArgumentException: duplicate element: 10
```

2. While using these factory methods if any element is null then we will get NullPointerException.

```
Set<String> fruits=Set.of("Apple","Banana",null); ➔NullPointerException
```

3. After creating the Set object, if we are trying to change the content(add|remove elements)then we will get UnsupportedOperationException b'z Set is immutable(unmodifiable).

```
Set<String> fruits=Set.of("Apple","Banana","Mango");
fruits.add("Orange");    //UnsupportedOperationException
fruits.remove("Apple"); //UnsupportedOperationException
```

Creation of unmodifiable Map (Immutable Map) with Java 9 Factory Methods:

Java 9 Map interface defines of() and ofEntries() Factory methods for this purpose.

1. static <K,V> Map<K,V> of()
2. static <K,V> Map<K,V> of(K k1,V v1)
3. static <K,V> Map<K,V> of(K k1,V v1,K k2,V v2)
4. static <K,V> Map<K,V> of(K k1,V v1,K k2,V v2,K k3,V v3)
5. static <K,V> Map<K,V> of(K k1,V v1,K k2,V v2,K k3,V v3,K k4,V v4)
6. static <K,V> Map<K,V> of(K k1,V v1,K k2,V v2,K k3,V v3,K k4,V v4,K k5,V v5)
7. static <K,V> Map<K,V> of(K k1,V v1,K k2,V v2,K k3,V v3,K k4,V v4,K k5,V v5,K k6,V v6)

8. static <K,V> Map<K,V> of(K k1,V v1,K k2,V v2,K k3,V v3,K k4,V v4,K k5,V v5,K k6,V v6,K k7,V v7)
9. static <K,V> Map<K,V> of(K k1,V v1,K k2,V v2,K k3,V v3,K k4,V v4,K k5,V v5,K k6,V v6,K k7,V v7,K k8,V v8)
10.static <K,V> Map<K,V> of(K k1,V v1,K k2,V v2,K k3,V v3,K k4,V v4,K k5,V v5,K k6,V v6,K k7,V v7,K k8,V v8,K k9,V v9)
11.static <K,V> Map<K,V> of(K k1,V v1,K k2,V v2,K k3,V v3,K k4,V v4,K k5,V v5,K k6,V v6,K k7,V v7,K k8,V v8,K k9,V v9,K k10,V v10)
12.static <K,V> Map<K,V> ofEntries(Map.Entry<? extends K,? extends V>... entries)

Note:

Up to 10 entries,it is recommended to use of() methods and for more than 10 items we should use ofEntries() method.

Eg:

```
Map<String,String> map=Map.of("A","Apple","B","Banana","C","Cat","D","Dog");
```

How to use Map.ofEntries() method:

Map interface contains static Method entry() to create immutable Entry objects.

```
Map.Entry<String,String> e=Map.entry("A","Apple");
```

This Entry object is immutable and we cannot modify its content. If we are trying to change we will get RE: UnsupportedOperationException

e.setValue("Durga"); ➔ UnsupportedOperationException

By using these Entry objects we can create unmodifiable Map object with Map.ofEntries() method.

Eg:

```
1)  import java.util.*;
2)  class Test
3)  {
4)     public static void main(String[] args)
5)     {
6)          Map.Entry<String,String> e1=Map.entry("A","Apple");
7)        Map.Entry<String,String> e2=Map.entry("B","Banana");
8)        Map.Entry<String,String> e3=Map.entry("C","Cat");
9)        Map<String,String> m=Map.ofEntries(e1,e2,e3);
10)       System.out.println(m);
11)
12)    }
13) }
```

In Short way we can also create as follows.

```
1)  import static java.util.Map.entry;
2)  Map<String,String> map=Map.ofEntries(entry("A","Apple"),entry("B","Banana"),entry("C"
    ,"Cat"),entry("D","Dog"));
```

Note:

1. While using these Factory Methods if we are trying to add duplicate keys then we will get IllegalArgumentException:duplicate key.But values can be duplicated.

```
Map<String,String> map=Map.of("A","Apple","A","Banana","C","Cat","D","Dog");
RE:  java.lang.IllegalArgumentException: duplicate key: A
```

2. While using these factory methods if any element is null(either key or value) then we will get NullPointerException.

```
Map<String,String> map=Map.of("A",null,"B","Banana"); ==>NullPointerException
Map<String,String> map=Map.ofEntries(entry(null,"Apple"),entry("B","Banana"));
                    ➔ NullPointerException
```

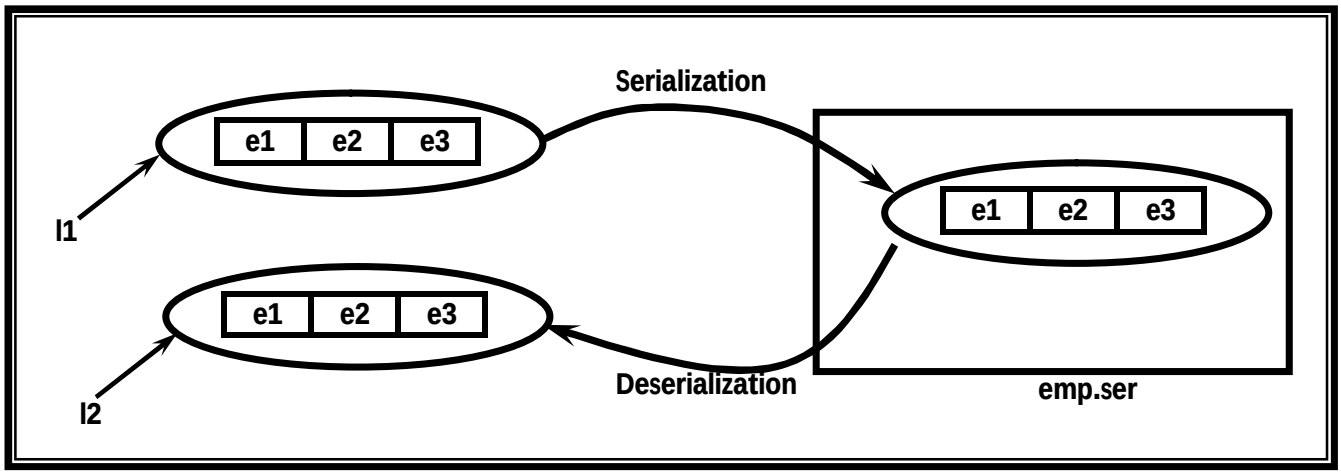
3. After creating the Map object, if we are trying to change the content(add|remove|replace elements)then we will get UnsupportedOperationException b'z Map is immutable(unmodifiable).

Eg:

```
Map<String,String> map=Map.ofEntries(entry("A","Apple"),entry("B","Banana"));
map.put("C","Cat"); ➔ UnsupportedOperationException
map.remove("A"); ➔ UnsupportedOperationException
```

Serialization for unmodifiable Collections:

The immutable collection objects are serializable iff all elements are serializable.
In Brief form,the process of writing state of an object to a file is called Serialization and the process of reading state of an object from the file is called Deserialization.




```
1)  import java.util.*;
2)  import java.io.*;
3)  class Employee implements Serializable
4)  {
5)     private int eno;
6)     private String ename;
7)     Employee(int eno,String ename)
8)     {
9)        this.eno=eno;
10)       this.ename=ename;
11)    }
12)    public String toString()
13)    {
14)       return String.format("%d=%s",eno,ename);
15)    }
16) }
17) class Test
18) {
```

```
19)    public static void main(String[] args) throws Exception
20)    {
21)       Employee e1= new Employee(100,"Sunny");
22)       Employee e2= new Employee(200,"Bunny");
23)       Employee e3= new Employee(300,"Chinny");
24)       List<Employee> l1=List.of(e1,e2,e3);
25)       System.out.println(l1);
26)
27)       System.out.println("Serialization of List Object...");
28)       FileOutputStream fos=new FileOutputStream("emp.ser");
29)       ObjectOutputStream oos=new ObjectOutputStream(fos);
30)       oos.writeObject(l1);
31)
32)       System.out.println("Deserialization of List Object...");
33)       FileInputStream fis=new FileInputStream("emp.ser");
34)       ObjectInputStream ois=new ObjectInputStream(fis);
35)       List<Employee> l2=(List<Employee>)ois.readObject();
36)       System.out.println(l2);
37)       //l2.add(new Employee(400,"Vinnny"));//UnsupportedOperationException
38)    }
39) }
```

o/p:

```
D:\durga_classes>java Test
[100=Sunny, 200=Bunny, 300=Chinny]
Serialization of List Object...
Deserialization of List Object...
[100=Sunny, 200=Bunny, 300=Chinny]
```

After deserialization also we cannot modify the content,otherwise we will get UnsupportedOperationException.

Note: The Factroy Methods introduced in Java 9 are not to create general collections and these are meant for creating immutable collections.