Object Oriented Programming with Java (Subject Code: BCS-403)

Unit 4

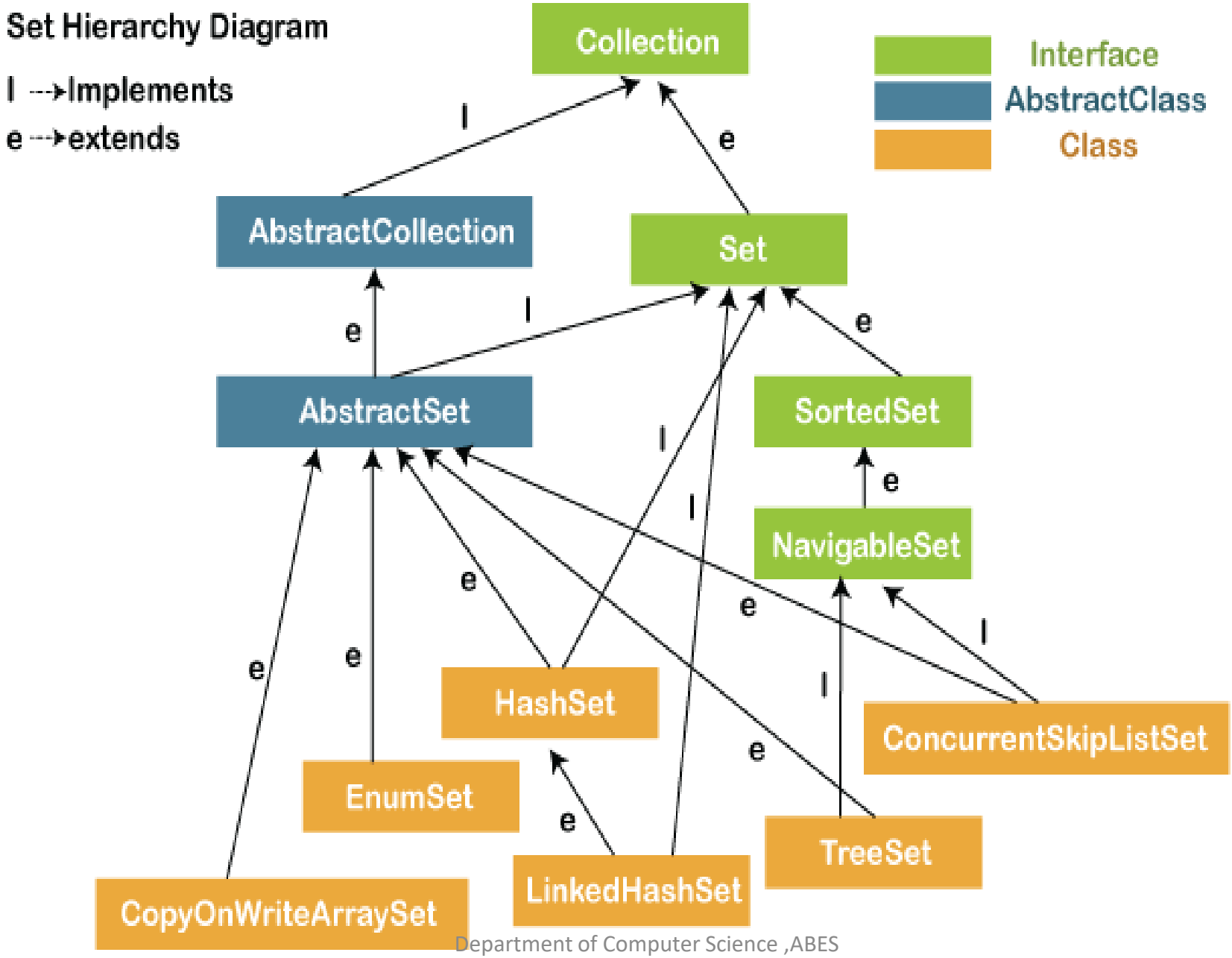
Lecture 31

Lecture 31

- **Set Interface**
- **HashSet**
- **LinkedHashSet**

Set Interface

- The set is an interface available in the java.util package.
- The set interface extends the Collection interface. An unordered collection or list in which duplicates are not allowed is referred to as a collection interface.
- The set interface is used to create the mathematical set.
- The set interface use collection interface's methods to avoid the insertion of the same elements. SortedSet and NavigableSet are two interfaces that extend the set implementation.


Set Hierarchy Diagram
I --▸Implements
e --▸extends
Collection
Interface
AbstractClass
Class
AbstractCollection
Set
AbstractSet
SortedSet
NavigableSet
HashSet
ConcurrentSkipListSet
EnumSet
TreeSet
LinkedHashSet
CopyOnWriteArraySet

Example

```
import java.util.*;
public class setExample{
   public static void main(String[] args)
   {
      // creating LinkedHashSet using the Set
Set<String> data = new LinkedHashSet<String>();
          data.add("Java");
      data.add("Set");
      data.add("Example");
      data.add("Set");
      System.out.println(data);
   }
}
```

Operations on the Set Interface

- On the Set, we can perform all the basic mathematical operations like intersection, union and difference.
- two sets, i.e.,

set1 = [22, 45, 33, 66, 55, 34, 77]

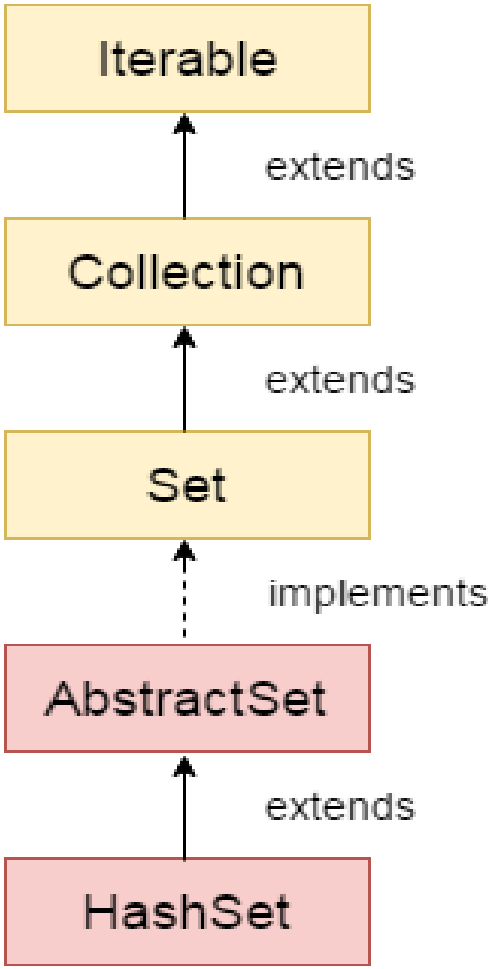
set2 = [33, 2, 83, 45, 3, 12, 55]

Intersection: The intersection operation returns all those elements which are present in both the set. The intersection of set1 and set2 will be [33, 45, 55].

- **Union:** The union operation returns all the elements of set1 and set2 in a single set, and that set can either be set1 or set2. The union of set1 and set2 will be [2, 3, 12, 22, 33, 34, 45, 55, 66, 77, 83].
- **Difference:** The difference operation deletes the values from the set which are present in another set. The difference of the set1 and set2 will be [66, 34, 22, 77].
- In set, addAll() method is used to perform the union, retainAll() method is used to perform the intersection and removeAll() method is used to perform difference.

Java HashSet

- Java HashSet class is used to create a collection that uses a hash table for storage.
- It inherits the Abstract Set class and implements Set interface.



important points about Java HashSet class are:

- HashSet stores the elements by using a mechanism called hashing.
- HashSet contains unique elements only.
- HashSet allows null value.
- HashSet class is non synchronized.
- HashSet doesn't maintain the insertion order. Here, elements are inserted on the basis of their hashcode.
- HashSet is the best approach for search operations.
- The initial default capacity of HashSet is 16.

Constructors of Java HashSet class

1) HashSet() It is used to construct a default HashSet.
2) HashSet(int capacity) It is used to initialize the capacity of the hash set to the given integer value capacity. The capacity grows automatically as elements are added to the HashSet.
3) HashSet(int capacity, float loadFactor) It is used to initialize the capacity of the hash set to the given integer value capacity and the specified load factor.

Methods of Java HashSet class

add(E e)	It is used to add the specified element to this set if it is not already present.
clear()	It is used to remove all of the elements from the set.
contains(Object o)	It is used to return true if this set contains the specified element.
isEmpty()	It is used to return true if this set contains no elements.
iterator()	It is used to return an iterator over the elements in this set.
remove(Object o)	It is used to remove the specified element from this set if it is present.
size()	It is used to return the number of elements in the set.

HashSet Example

```
import java.util.*;
class HashSet1{
 public static void main(String args[]){
  //Creating HashSet and adding elements
    HashSet<String> set=new HashSet();
          set.add("One");
          set.add("Two");
          set.add("Three");
          set.add("Four");
          set.add("Five");
          Iterator<String> i=set.iterator();
          while(i.hasNext())
          {
          System.out.println(i.next());
          }
 } }
```

HashSet example ignoring duplicate elements

```
HashSet<String> set=new HashSet<String>();
 set.add("Ravi");
 set.add("Vijay");
 set.add("Ravi");
 set.add("Ajay");
 //Traversing elements
 Iterator<String> itr=set.iterator();
 while(itr.hasNext()){
  System.out.println(itr.next());
 }
```

Ajay

Vijay

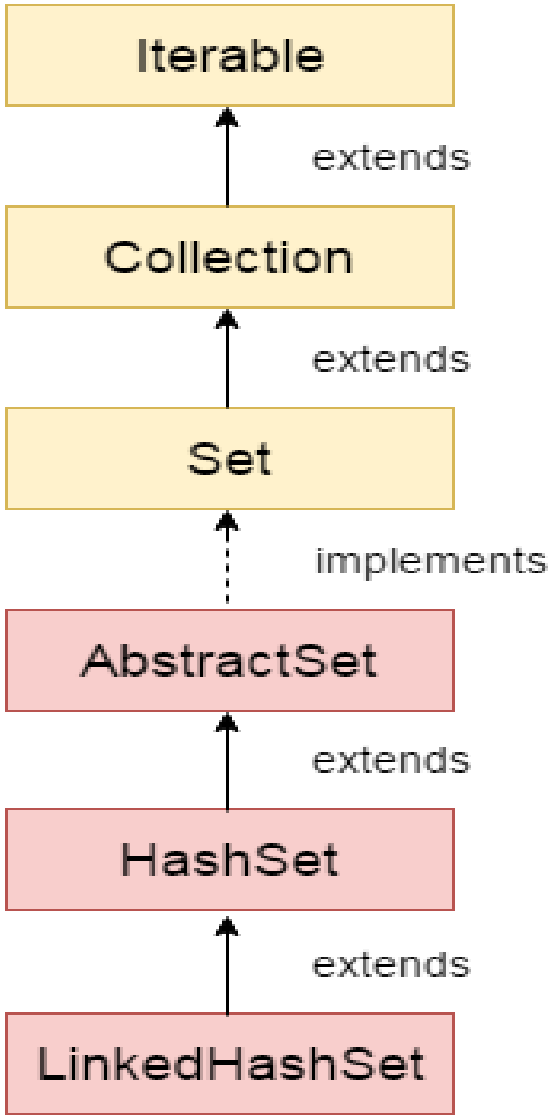
Ravi

HashSet example to remove elements

```
 set.remove("Ravi");
 //Removing all the elements from HashSet
  HashSet<String> set1=new HashSet<String>();
      set1.add("Ajay");
      set1.add("Gaurav");
      set.removeAll(set1);
//Removing elements on the basis of specified condition
      set.removeIf(str->str.contains("Vijay"));
      //Removing all the elements available in the set
      set.clear();
```

LinkedHashSet Class

- Java LinkedHashSet class is a Hashtable and Linked list implementation of the Set interface.
- It inherits the HashSet class and implements the Set interface.



Java LinkedHashSet class are

- Java LinkedHashSet class contains unique elements only like HashSet.
- Java LinkedHashSet class provides all optional set operations and permits null elements.
- Java LinkedHashSet class is non-synchronized.
- Java LinkedHashSet class maintains insertion order.

Note: Keeping the insertion order in the LinkedHashset has some additional costs, both in terms of extra memory and extra CPU cycles. Therefore, if it is not required to maintain the insertion order, go for the lighter-weight HashMap or the HashSet instead.

Constructors of Java LinkedHashSet Class

Constructor	Description
LinkedHashSet(int capacity, float fillRatio)	It is used to initialize both the capacity and the fill ratio (also called load capacity) of the hash set from its argument.
LinkedHashSet(int capacity)	It is used to initialize the capacity of the linked hash set to the given integer value capacity.

```
import java.util.*;
class LinkedHashSet1{
 public static void main(String args[]){
 //Creating HashSet and adding elements
       LinkedHashSet<String> set=new LinkedHashSet();
            set.add("One");
            set.add("Two");
            set.add("Three");
            set.add("Four");
            set.add("Five");
            Iterator<String> i=set.iterator();
            while(i.hasNext())
            {
            System.out.println(i.next());
            }
 }
}
```

Output:

```
One
Two
Three
Four
Five
```

```
// Creating an empty LinekdhashSet of string type
LinkedHashSet<String> lhs = new LinkedHashSet<St
ring>();

// Adding elements to the above Set
// by invoking the add() method
lhs.add("Java");
lhs.add("T");
lhs.add("Point");
lhs.add("Good");
lhs.add("Website");
```

```
// displaying all the elements on the console
System.out.println("The hash set is: " + lhs);
 The hash set is: [Java, T, Point, Good, Website]
// Removing an element from the above linked Set
// since the element "Good" is present, therefore, the method remove
()
// returns true
System.out.println(lhs.remove("Good"));
 true
// After removing the element
System.out.println("After removing the element, the hash set is: " + lhs);
After removing the element, the hash set is: [Java, T, Point, Website]
// since the element "For" is not present, therefore, the method remo
ve()
// returns false
System.out.println(lhs.remove("For"));
false
```