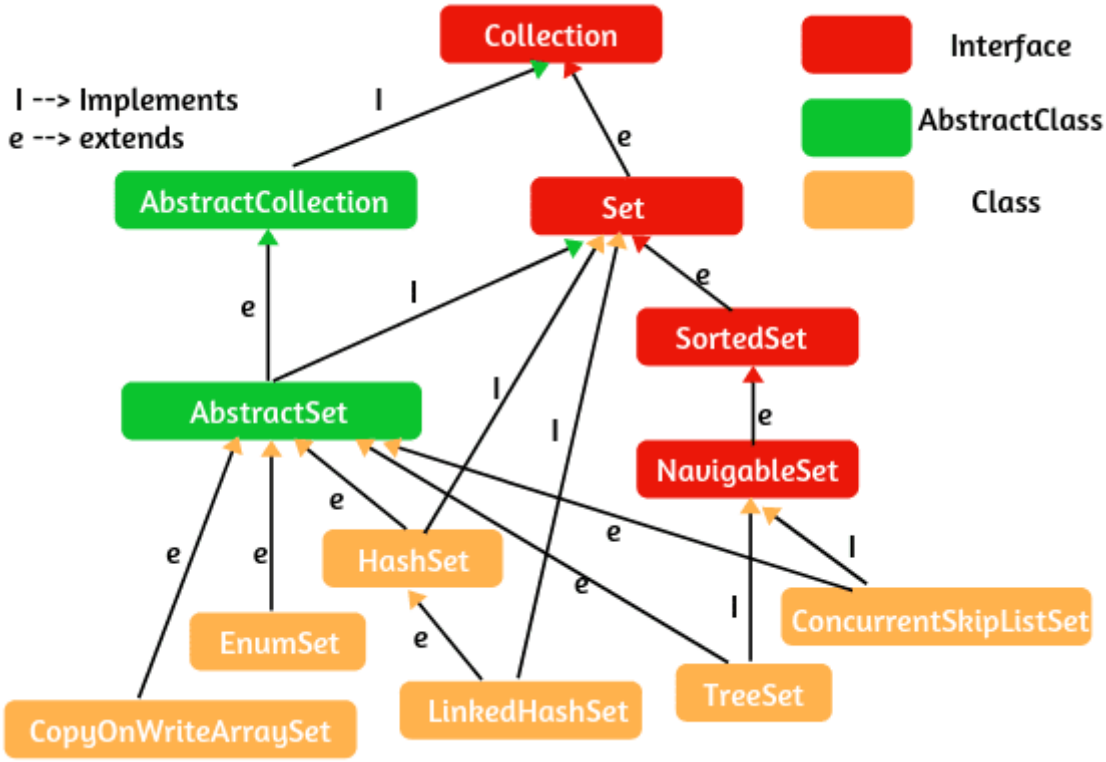
Set

A **set in Java** is an unordered collection of unique elements or objects. In other words, a set is a collection of elements that are not stored in a particular order.p

Elements are simply added to the set without any control over where they go. For example, in many applications, we do not need to care about the order of elements in a collection.




```
package setProgram;
import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.Set;
public class SetExample1
{
public static void main(String[] args)
{
// Create a generic set object of type String.
   Set<String> s = new HashSet<String>();
// Find out the size of set using set() method.
   int size = s.size();
   System.out.println("Size before adding elements: " +size);
```

```
// Adding elements to set.
   s.add("Orange"); // s.size()is 1.
   s.add("Red"); // s.size() is 2.
   s.add("Blue"); // s.size() is 3.
   s.add("Yellow"); // s.size() is 4.
   s.add("Green"); // Now s.size() is 5.

// Add duplicate elements in the set. These elements will be ignored by set due to not
taking duplicate elements.
   s.add("Red"); // s.size() is still 5 because we cannot add duplicate element.
   s.add("Blue"); // s.size() is still 5 because we cannot add duplicate element.
   System.out.println("Unordered Set Elements");
   System.out.println(s);

// Check 'Black' element is present in the above set or not.
   if(s.equals("Black"))
   {
     System.out.println("Black element is not present in set.");
   }
   else
   {
     s.add("Black");
     System.out.println("Black is added successfully.");
     System.out.println("Set elements after adding black element");
     System.out.println(s);
   }
// Create another set object to add collection of elements to the above set.
   Set<String> s2 = new LinkedHashSet<String>();
    s2.add("White");
    s2.add("Brown");
    s2.add("Red"); // Duplicate element.

// Call addAll() method to add all the elements of the given collection.
   s.addAll(s2);
   System.out.println("Set elements after adding elements from given collection");
   System.out.println(s);
 }
}
```

```
Output:
       Size before adding elements: 0
       Unordered Set Elements
       [Red, Blue, Yellow, Orange, Green]
       Black is added successfully.
       Set elements after adding black element
       [Red, Blue, Yellow, Black, Orange, Green]
       Set elements after adding elements from given collection
       [Red, Brown, White, Blue, Yellow, Black, Orange, Green]
```

```
package setProgram;
import java.util.HashSet;
import java.util.Set;
public class SetExample2
```

```
{
public static void main(String[] args)
{
// Create a generic set object of type String.
   Set<String> s = new HashSet<String>();

// Check set is empty or not.
   boolean check = s.isEmpty(); // Return type of this method is an boolean.
   System.out.println("Is Set empty: " +check);

// Adding elements to set.
   s.add("Orange");
   s.add("Red");
   s.add("Blue");
   s.add("Yellow");
   s.add("Green");
 if(s.isEmpty())
 {
  System.out.println("Set is empty.");
  }
 else
 {
   System.out.println("Set is not empty.");
   System.out.println("Elements in the Set");
   System.out.println(s);
  }
// Remove an element from set.
   s.remove("Blue");
   System.out.println("Set elements after removing");
   System.out.println(s);

// Get the total number of set elements.
   int size = s.size();
   System.out.println("Total number of elements: " +size);
  }
}
```

```
Output:
       Is Set empty: true
       Set is not empty.
       Elements in the Set
       [Red, Blue, Yellow, Orange, Green]
       Set elements after removing
       [Red, Yellow, Orange, Green]
       Total number of elements: 4
```

```
package setProgram;
import java.util.HashSet;
import java.util.Set;
public class SetExample3 {
public static void main(String[] args)
{
 Set<Character> s = new HashSet<Character>();
```

```
  s.add('D');
  s.add('F');
  s.add('H');
  s.add('P');
  s.add('K');
  s.add(null);
  s.add(null); // Duplicate null element. Therefore, set allow only one null element.

  System.out.println("Unordered Set Elements");
  System.out.println(s);

// Call contains() method to search an element.
   boolean search = s.contains('A'); // Returns false because A is not present in the
 set.

   System.out.println("Is Element A present in set: " +search);
   if(s.contains('K'))
   {
     System.out.println("K is present in set.");
   }
   else {
     System.out.println("K is not present in set.");
   }
   int hashcode = s.hashCode();
   System.out.println("HashCode value: " +hashcode);
 }
}
```

```
Output:
       Unordered Set Elements
       [P, null, D, F, H, K]
       Is Element A present in set: false
       K is present in set.
       HashCode value: 365
```

```
package setProgram;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
public class SetExample4 {
public static void main(String[] args)
{
// Create a generic list object of type Integer.
   List<Integer> list = new ArrayList<Integer>();
   int size = list.size();
   System.out.println("Size before adding elements: " +size);

// Adding elements to list object.
   list.add(5);
   list.add(10);
   list.add(5);
   list.add(15);
```

```
   list.add(20);
   list.add(10);
   list.add(20);
   list.add(30);

   System.out.println("Original order of List Elements");
   System.out.println(list);

// Creating a hash set object of type Integer.
   Set<Integer> s = new HashSet<Integer>(list);
   System.out.println("Unodered List Elements after removing duplicates.");
   System.out.println(s);
 }
}
```

```
Output:
       Size before adding elements: 0
       Original order of List Elements
       [5, 10, 5, 15, 20, 10, 20, 30]
       Unodered List Elements after removing duplicates
       [20, 5, 10, 30, 15]
```

```
package iterateSet;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
public class IterateSetEx {
public static void main(String[] args)
{
// Create a generic set object of type String.
   Set<String> s = new HashSet<String>(); // s.size() is 0.
   int size = s.size();
   System.out.println("Size before adding elements: " +size);

// Adding elements to set.
   s.add("Orange"); // s.size() is 1.
   s.add("Red"); // s.size() is 2.
   s.add("Blue"); // s.size() is 3.
   s.add("Yellow"); // s.size() is 4.
   s.add("Green"); // s.size() is 5.

// Displaying elements of set.
   System.out.println("Elements in set");
   System.out.println(s);

// Iterating set in the forward direction.
// Creating an Iterator object using iterator() method.
   Iterator<String> itr = s.iterator();
   System.out.println("Iteration using Iterator method");
   while(itr.hasNext())
   {
     Object str = itr.next();
     System.out.println(str);
```

```
   }
 }
}
```

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
public class AddDemo {
public static void main(String[] args)
{
// Creating a set object.
   Set<String> set= new HashSet<>();

// Adding elements to set.
   set.add("Banana");
   set.add("Orange");
   set.add("Apple");
   set.add("Mango");

// Creating an iterator object.
   Iterator<String> itr = set.iterator();
   while(itr.hasNext())
   {
     Object str = itr.next();
     System.out.println(str);
     set.add("Grapes"); // Adding element during iteration. It will throw ConcurrentMo
dificationException.
   }
  }
}
```

```
Output:
       Apple
       Exception in thread "main" java.util.ConcurrentModificationException
```

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
public class RemoveDemo {
public static void main(String[] args)
{
  Set<String> set= new HashSet<>();
  set.add("Banana");
  set.add("Orange");
  set.add("Apple");
  set.add("Mango");

  Iterator<String> itr = set.iterator();
  while(itr.hasNext())
  {
```

```
    Object str = itr.next();
    System.out.println(str);

 // Removing Mango element.
    if(str.equals("Mango"))
    {
      itr.remove();
    }
   }
  System.out.println(set);
 }
}
```

```
Output:
      Apple
      Mango
      Orange
      Banana
      [Apple, Orange, Banana]
```

```
package iterateSet;
import java.util.HashSet;
import java.util.Set;
public class IterateSetEx2 {
public static void main(String[] args)
{
// Create Set object of type Integer.
   Set<Integer> s = new HashSet<Integer>();
// Adding even numbers betwen 10 to 30 as elements.
   for(int i = 10; i < = 30; i++)
   {
     if(i % 2 == 0)
     {
       s.add(i);
     }
   }
   System.out.println("Even numbers between 10 to 30");
   System.out.println(s);
   System.out.println("Iteration Using Enchanced For Loop");

// Applying enhanced for loop to iterate over set.
   for(Integer it:s)
   {
     System.out.println(it);
   }
 }
}
```

```
Output:
       Even numbers between 10 to 30
```

```
       [16, 18, 20, 22, 24, 10, 26, 12, 28, 14, 30]
       Iteration Using Enhanced For Loop
       16 18 20 22 24 10 26 12 28 14 30
```

```
package iterateSet;
import java.util.HashSet;
import java.util.Set;
public class IterateSetEx3 {
public static void main(String[] args)
{
  Set<Character> s = new HashSet<Character>();
  s.add('A');
  s.add('B');
  s.add('C');
  s.add('D');
  s.add('E');
  System.out.println(s);

  System.out.println("Iterating using forEach loop in Java 1.8");
  s.forEach(System.out::println);
 }
}
```

```
Output:
       [A, B, C, D, E]
       Iterating using forEach loop in Java 1.8
       A B C D E
```

```
package iterateSet;
import java.util.HashSet;
import java.util.Set;
public class IterateSetEx4 {
public static void main(String[] args)
{
  Set<String> set = new HashSet<>();
   set.add("apple");
   set.add("banana");
   set.add("orange");
   set.add("mango");
   set.add("grapes");

// Creating stream object by calling stream() method.
   set.stream().forEach(System.out::println);
 }
}
```

```
Output:
       banana
```

```
        orange
        apple
        mango
        grapes
```