# **SET**

**Key Points**

1. **Set Interface Inherits from Collection** – It does not introduce new methods.
2. **HashSet is the Main Implementation** – The Java Collections Framework provides HashSet as a concrete implementation.
3. **Uses HashMap Internally** – HashSet internally wraps an instance of HashMap to store elements.
4. **Prevents Duplicates** – Unlike List, Set ensures uniqueness.
5. **No Guaranteed Order** – Elements in a Set are stored in an unpredictable order.
6. **Iteration Order is Uncertain** – Unlike List, the insertion order is **not** maintained.

Basic Operations

1. Creating a Set:

```Set<String> set = new HashSet<>();```

1. Add

```set.add("Apple");

set.add("Banana");

set.add("Cherry");```

1. Removing Elements (remove())

```set.remove("Banana");```

1. Checking if an Element Exists (contains())

```boolean exists = set.contains("Apple"); // true```

1. Iterate using foreach

```set.forEach(System.out::println);```

1. Clearing a Set (clear())

```set.clear(); // Removes all elements```

1. Checking if Set is Empty (isEmpty())

```boolean empty = set.isEmpty(); // true if set is empty```

1. Adding All Elements from Another Collection (addAll())

```Set<String> set2 = new HashSet<>(List.of("Mango", "Orange"));

set.addAll(set2);```

1. Removing All Elements (removeAll())

```set.removeAll(set2); // Removes all elements in set2 from set```

1. Retaining Common Elements (retainAll())

```set.retainAll(Set.of("Apple", "Mango")); // Keeps only these elements```

1. Covert to list

```List<String> list = new ArrayList<>(set);```