Dart – Day9

Emp-id : 4781

- **Set**

A Set in Dart:

- Unordered collection of unique elements (no duplicates).
- Implements Iterable, so it inherits many iterable methods.
- Backed internally by LinkedHashSet (insertion order preserved).

1. Properties

length

Returns the number of elements in the set.

```
var nums = {10, 20, 30};
print(nums.length); // 3
```

isEmpty

Returns true if the set has no elements.

```
print({}.isEmpty); // true
```

isNotEmpty

Returns true if the set has at least one element.

```
print({1}.isNotEmpty); // true
```

first

Returns the first element in the set (based on insertion order).
- Throws error if the set is empty.

```
var fruits = {"apple", "banana"};
print(fruits.first); // apple
```

last

Returns the last element in the set (in insertion order).
- Throws error if the set is empty.

```
print({"apple", "banana"}.last); // banana
```

single

Returns the only element in the set.
- Throws error if the set has 0 or more than 1 element.

```
print({"onlyOne"}.single); // onlyOne
```

2. Adding Elements

add(E value)

Adds an element if it doesn’t already exist. Returns true if added, false if it was already present.

```
var nums = {1, 2};
print(nums.add(3)); // true
print(nums.add(2)); // false (duplicate not added)
```

addAll(Iterable<E> elements)

Adds multiple elements at once.

```
var nums = {1};
nums.addAll([2, 3, 3]);
print(nums); // {1, 2, 3}
```

3. Removing Elements

remove(Object? value)

Removes a specific element. Returns true if it was present.

```
var nums = {1, 2, 3};
nums.remove(2);
print(nums); // {1, 3}
```

removeAll(Iterable<Object?> elements)

Removes all matching elements.

```
var nums = {1, 2, 3, 4};
nums.removeAll([2, 4]);
print(nums); // {1, 3}
```

removeWhere(bool test(E element))

Removes all elements that satisfy the condition.

```
var nums = {1, 2, 3, 4, 5};
nums.removeWhere((n) => n.isEven);
print(nums); // {1, 3, 5}
```

retainAll(Iterable<Object?> elements)

Keeps only the elements that are in another collection.

```
var nums = {1, 2, 3, 4};
nums.retainAll([2, 3]);
print(nums); // {2, 3}
```

retainWhere(bool test(E element))

Keeps only elements that satisfy the condition.

```
var nums = {1, 2, 3, 4, 5};
nums.retainWhere((n) => n > 3);
print(nums); // {4, 5}
```

clear()

Removes all elements.

```
var nums = {1, 2, 3};
nums.clear();
print(nums); // {}
```

4. Lookup & Check

contains(Object? element)

Checks if an element exists.

```
print({1, 2, 3}.contains(2)); // true
```

containsAll(Iterable<Object?> elements)

Checks if all elements are present.

```
print({1, 2, 3}.containsAll([1, 3])); // true
```

elementAt(int index)

Returns element at given index (based on insertion order).

```
var nums = {10, 20, 30};
print(nums.elementAt(1)); // 20
```

5. Iteration & Functional Methods

forEach(void f(E element))

Runs a function on each element.

```
{1, 2, 3}.forEach((n) => print(n * n));
```

map<T>(T f(E e))

Transforms each element into another form (returns Iterable).

```
print({1, 2, 3}.map((n) => n * 2)); // (2, 4, 6)
```

where(bool test(E e))

Filters elements based on condition.

```
print({1, 2, 3, 4}.where((n) => n.isEven)); // (2, 4)
```

expand<T>(Iterable<T> f(E e))

Expands each element into multiple elements.

print({1, 2}.expand((n) => [n, n * 10])); // (1, 10, 2, 20)

any(bool test(E e))

Returns true if any element matches condition.

print({1, 2, 3}.any((n) => n > 2)); // true

every(bool test(E e))

Returns true if all elements match condition.

print({2, 4, 6}.every((n) => n.isEven)); // true

join([String separator])

Concatenates elements into a string.

print({1, 2, 3}.join("-")); // "1-2-3"

6. Set Operations

union(Set<E> other)

Returns a new set containing all elements.

print({1, 2}.union({2, 3})); // {1, 2, 3}

intersection(Set<Object?> other)

Returns common elements.

```
print({1, 2, 3}.intersection({2, 3, 4})); // {2, 3}
```

difference(Set<Object?> other)

Returns elements present in first set but not in second.

```
print({1, 2, 3}.difference({2, 4})); // {1, 3}
```

7. Conversion

toList()

Converts set to a list.

```
print({1, 2, 3}.toList()); // [1, 2, 3]
```

toSet()

Creates a copy of the set.

```
var s1 = {1, 2, 3};
var s2 = s1.toSet();
print(s2); // {1, 2, 3}
```