Dart – Week2-Day1

Set

- A Set is a collection of unique items (no duplicates).
- It’s unordered — the order of items doesn’t matter.
- Like a list, but it automatically removes duplicates.

Creating Sets

1. Using a Set literal

```
var numbers = {1, 2, 3, 3, 4};
print(numbers); // Output: {1, 2, 3, 4} //duplicates removed
```

2. Using the Set constructor

```
var fruits = Set<String>();
fruits.add('apple');
fruits.add('banana');
fruits.add('apple');  // Duplicate, won't be added again

print(fruits);  // Output: {apple, banana}
```

3. Set.from()

- Creates a new set from any iterable (like a List, another Set, etc.)
- Less strict on type checking (can accept dynamic types).

```
var list = [1, 2, 3];
var mySet = Set.from(list);

print(mySet); // Output: {1, 2, 3}
```

4. Set.of()

- Also creates a new set from an iterable.
- More type-safe and preferred in newer Dart code.
- Will throw a compile-time error if types don't match.

```
List<int> numbers = [1, 2, 3];
Set<int> mySet = Set.of(numbers);

print(mySet); // Output: {1, 2, 3}
```

If you try mixing types:

```
var mixed = [1, 'two', 3];
// Set<int> badSet = Set.of(mixed);  //  Error because 'two' is not an int
```

5. Set.unmodifiable()

- Creates a read-only (immutable) set from an iterable.
- You cannot add or remove elements after creation.
- Useful to protect data from being changed.

```
var original = {1, 2, 3};
var unmodifiableSet = Set.unmodifiable(original);

print(unmodifiableSet); // {1, 2, 3}

unmodifiableSet.add(4); // Throws error — cannot modify
```

6. Set<Type>()

- Creates an empty set with a specified type.

- The type defines what kind of elements the set can hold.

```
Set<String> fruits = Set<String>();
fruits.add('apple');
fruits.add('banana');

print(fruits); // {apple, banana}
```

Common Set Methods in Dart

1. add(element)

- Adds a new element to the set.
- If an element already exists, nothing changes.

```
var mySet = {1, 2, 3};
mySet.add(4);
print(mySet); // {1, 2, 3, 4}
```

2. addAll(iterable)

- Adds all elements from another iterable (list, set, etc.) to the set.

```
mySet.addAll([4, 5, 6]);
print(mySet); // {1, 2, 3, 4, 5, 6}
```

3. remove(element)

- Removes an element from the set.

```
mySet.remove(2);
print(mySet); // {1, 3, 4, 5, 6}
```

4. Contains(element)

- Checks if the set contains the element.
- Returns true or false.

```
print (mySet.contains(3)); // true
print(mySet.contains(10)); // false
```

5. **clear ()**

- Removes all elements from the set.

```
mySet.clear();
print(mySet); // {}
```

6. length

- Returns the number of elements in the set.

```
print(mySet.length); // Number of items in mySet
```

7. isEmpty and isNotEmpty

- Check if the set is empty or not.

```
print (mySet.isEmpty);   // true or false
print(mySet.isNotEmpty); // true or false
```