Swift Collections

Swift provides several types of collections that allow you to store and manage groups of values. The main collection types in Swift are arrays, sets, and dictionaries. Each type is optimised for specific tasks and has its own strengths. Here is an overview of these collections along with some examples:

Arrays

An array is an ordered collection of values. Arrays can store values of any type, ***but all values must be of the same type.***

Creating and Initializing Arrays:

```
// Creating an empty array
var emptyArray: [Int] = []

// Creating an array with a default value
var repeatingArray = Array(repeating: 0, count: 5)

// Creating an array with an array literal
var numberArray = [1, 2, 3, 4, 5]
```

Accessing and Modifying Arrays:

```
// Accessing elements
let firstElement = numberArray[0]

// Modifying elements
numberArray[0] = 10

// Adding elements
numberArray.append(6)
numberArray += [7, 8]

// Removing elements
numberArray.remove(at: 0)
numberArray.removeLast()
numberArray.removeAll()
```

Sets

A set is an unordered collection of unique values. Sets are useful when you need to ensure that ***an item appears only once***.

Creating and Initializing Sets:

```
// Creating an empty set
var emptySet: Set<Int> = []

// Creating a set with a set literal
var numberSet: Set = [1, 2, 3, 4, 5]
```

Accessing and Modifying Sets:

```
// Adding elements
numberSet.insert(6)

// Removing elements
numberSet.remove(1)

// Checking membership
if numberSet.contains(2) {
    print("Set contains 2")
}

// Iterating over a set
for number in numberSet {
    print(number)
}
```

Dictionaries

A dictionary is an unordered collection of key-value pairs. ***Each value is associated with a unique key.***

Creating and Initializing Dictionaries:

```
// Creating an empty dictionary
var emptyDict: [String: Int] = [:]

// Creating a dictionary with a dictionary literal
var numberDict = ["one": 1, "two": 2, "three": 3]
```

Accessing and Modifying Dictionaries:

```
// Accessing elements
let value = numberDict["one"]
```

```
// Modifying elements
numberDict["four"] = 4

// Adding elements
numberDict["five"] = 5

// Removing elements
numberDict["one"] = nil

// Iterating over a dictionary
for (key, value) in numberDict {
    print("\(key): \(value)")
}
```

Common Collection Methods

Swift collections provide many useful methods that can be used with arrays, sets, and dictionaries. Here are a few common ones:

Count: Returns the number of elements.
`let count = numberArray.count`

IsEmpty: Checks if the collection is empty.
`let isEmpty = numberSet.isEmpty`

First and Last: Access the first and last elements (arrays only).
`let first = numberArray.first`
`let last = numberArray.last`

Sorting: Sort the collection (arrays only).
`let sortedArray = numberArray.sorted()`

Filtering: Filter elements based on a condition.
`let evenNumbers = numberArray.filter { $0 % 2 == 0 }`

Mapping: Transform each element using a closure.
`let stringArray = numberArray.map { "\($0)" }`

Reducing: Combine elements into a single value.
let sum = numberArray.reduce(0, +)

By understanding these basic collection types and their methods, you can effectively manage groups of values in your Swift programs.