📚 JAVA COLLECTION FRAMEWORK – COMPLETE NOTES WITH CODE

◆ 1️⃣ What is Java Collection Framework?

Java Collection Framework (JCF) is a set of **classes and interfaces** used to **store, manage, and manipulate multiple objects** efficiently.

It provides **ready-made data structures** like:

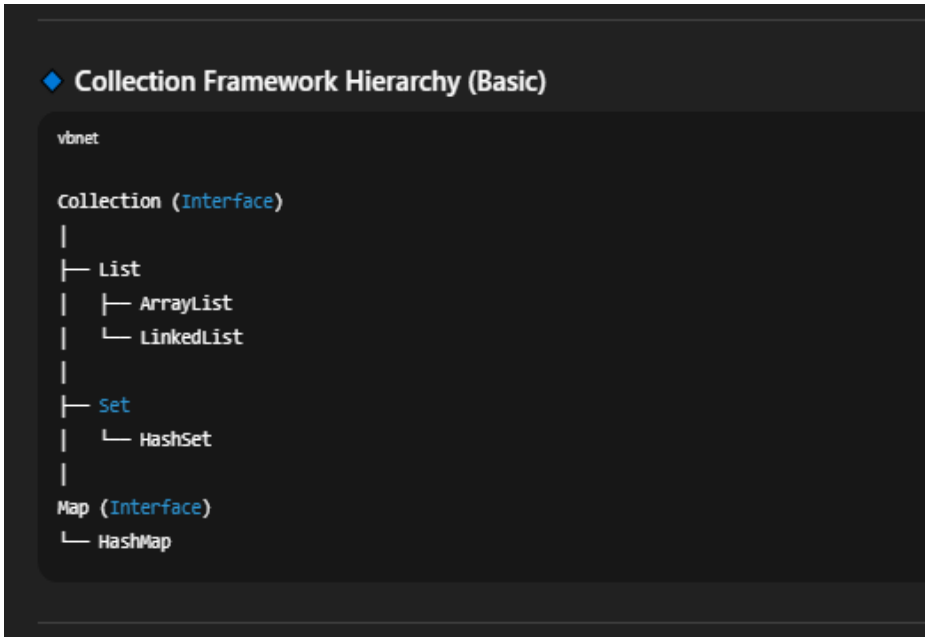
- List
- Set
- Map

instead of creating logic from scratch.

◆ Why do we use Collection Framework?

We use Collection Framework because:

- Arrays have **fixed size**
- Collections are **dynamic**
- Easy data manipulation
- Built-in methods available
- Faster development
- Widely used in real projects



◆ 2 ArrayList

✅ What is ArrayList?

ArrayList is a **class that implements List interface**.
It stores data in a **dynamic array**.

✅ Why do we use ArrayList?

- Fast access using index
- Dynamic size
- Allows duplicate elements
- Maintains insertion order

✅ Important Features

- Allows duplicates
- Maintains order
- Allows multiple null values
- Non-synchronized

✅ Important Methods

Method	Use
add()	Add element
get()	Access element
set()	Update element
remove()	Remove element
size()	Total elements
contains()	Element present?

Array List example

```java
import java.util.ArrayList;

public class ArrayListDemo {
    public static void main(String[] args) {

        ArrayList<Integer> list = new ArrayList<>();

        list.add(10);
        list.add(20);
        list.add(30);
        list.add(10);   // duplicate allowed

        System.out.println(list);   // [10, 20, 30, 10]

        System.out.println(list.get(1)); // 20

        list.set(2, 50);   // update
        System.out.println(list);

        list.remove(0);    // remove by index
        System.out.println(list);

        System.out.println(list.size()); // total elements
    }
}
```

◆ 3 LinkedList

✅ What is LinkedList?

LinkedList is a **List implementation** where elements are stored as **nodes** (data + next reference).

✅ Why do we use LinkedList?

- Fast insertion & deletion
- No shifting of elements
- Used when frequent add/remove needed

✅ Important Features

- Allows duplicates
- Maintains insertion order
- Slower access than ArrayList
- Can act as Stack & Queue

✅ Important Methods

Method	Use
addFirst()	Add at start
addLast()	Add at end
removeFirst()	Remove first
removeLast()	Remove last
getFirst()	First element
getLast()	Last element

✅ LinkedList Code Example

java

```java
import java.util.LinkedList;

public class LinkedListDemo {
    public static void main(String[] args) {

        LinkedList<String> list = new LinkedList<>();

        list.add("Java");
        list.add("Python");
        list.addFirst("C");
        list.addLast("AI");

        System.out.println(list);

        list.removeFirst();
        System.out.println(list);

        System.out.println(list.getFirst());
    }
}
```

◆ 4 HashSet

✓ What is HashSet?

HashSet is a **Set implementation** that stores **only unique elements**.

✓ Why do we use HashSet?

- To remove duplicates
- Fast searching
- Unordered collection

✓ Important Features

- No duplicate elements
- No insertion order
- Allows only one null value

✓ Important Methods

Method	Use
add()	Add element
remove()	Remove element
contains()	Element exists?
size()	Total elements
clear()	Remove all

✓ HashSet Code Example

```java
import java.util.HashSet;

public class HashSetDemo {
    public static void main(String[] args) {

        HashSet<Integer> set = new HashSet<>();

        set.add(10);
        set.add(20);
        set.add(10);   // duplicate ignored

        System.out.println(set); // [10, 20]

        System.out.println(set.contains(20));
    }
}
```

◆ 5 HashMap

✅ What is HashMap?

HashMap is a **Map implementation** that stores data in **key–value pairs**.

✅ Why do we use HashMap?

- Fast data retrieval
- Used in login systems, caching, searching
- Key-based access

✅ Important Features

- Keys are unique
- Values can be duplicate
- One null key allowed
- No insertion order

✅ Important Methods

Method	Use
put()	Add key-value
get()	Get value
remove()	Remove entry
Contains Key()	Key present?
Key Set()	All keys
values()	All values

✅ Hash Map Code Example

```java
import java.util.HashMap;
import java.util.Map;

public class HashMapDemo {
    public static void main(String[] args) {

        HashMap<Integer, String> map = new HashMap<>();

        map.put(1, "Java");
        map.put(2, "Python");
        map.put(3, "AI");

        System.out.println(map);

        System.out.println(map.get(2));

        for(Map.Entry<Integer, String> e : map.entrySet()) {
            System.out.println(e.getKey() + " : " + e.getValue());
        }
    }
}
```

⭐ VERY IMPORTANT INTERVIEW DIFFERENCE

Feature	ArrayList	LinkedList	HashSet	HashMap
Order	Yes	Yes	No	No
Duplicate	Yes	Yes	No	Values only
Access	Fast	Slow	Fast	Very fast
Key-Value	No	No	No	Yes

◆ ARRAYLIST – Examples & Questions

✓ Q1. ArrayList banao aur elements add karo

Question:
10, 20, 30, 40 add karo aur print karo.

```
import java.util.ArrayList;

public class Q1 {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        list.add(30);
        list.add(40);

        System.out.println(list);
    }
}
```

📌 Explanation:
ArrayList dynamic hoti hai aur insertion order maintain karti hai.

✓ Q2. ArrayList ka sum nikalo

```
import java.util.ArrayList;

public class Q2 {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        list.add(30);

        int sum = 0;
        for(int i : list) {
            sum += i;
        }
        System.out.println("Sum = " + sum);
    }
}
```

✅ Q3. Duplicate element remove karo

```
import java.util.ArrayList;

public class Q3 {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        list.add(10);

        System.out.println(list);
    }
}
```

📌 **Note:**
ArrayList **duplicates allow karti hai**.

◆ LINKEDLIST – Examples & Questions

✅ Q4. LinkedList me first aur last element add karo

```
import java.util.LinkedList;

public class Q4 {
    public static void main(String[] args) {
        LinkedList<String> list = new LinkedList<>();
        list.add("Java");
        list.addFirst("C");
        list.addLast("Python");

        System.out.println(list);
    }
}
```

✅ Q5. First element remove karo

```
import java.util.LinkedList;

public class Q5 {
    public static void main(String[] args) {
        LinkedList<Integer> list = new LinkedList<>();
        list.add(1);
        list.add(2);
        list.add(3);
```

```
        list.removeFirst();
        System.out.println(list);
    }
}
```

📌 Explanation:
LinkedList insertion/deletion ke liye best hoti hai.

◆ HASHSET – Examples & Questions

✓ Q6. HashSet me duplicate add karo

```
import java.util.HashSet;

public class Q6 {
    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<>();
        set.add(10);
        set.add(20);
        set.add(10);

        System.out.println(set);
    }
}
```

📌 Output:
Duplicate element ignore ho jaata hai.

✓ Q7. Element present hai ya nahi check karo

```
import java.util.HashSet;

public class Q7 {
    public static void main(String[] args) {
        HashSet<String> set = new HashSet<>();
        set.add("Java");
        set.add("Python");

        System.out.println(set.contains("Java"));
    }
}
```

◆ HASHMAP – Examples & Questions

✓ Q8. Roll number aur name store karo

```
import java.util.HashMap;

public class Q8 {
    public static void main(String[] args) {
        HashMap<Integer, String> map = new HashMap<>();
        map.put(1, "Aman");
        map.put(2, "Neha");
        map.put(3, "Ravi");

        System.out.println(map);
    }
}
```

✓ Q9. Specific key ka value print karo

```
import java.util.HashMap;

public class Q9 {
    public static void main(String[] args) {
        HashMap<Integer, String> map = new HashMap<>();
        map.put(101, "Java");
        map.put(102, "Python");

        System.out.println(map.get(102));
    }
}
```

✓ Q10. Hash Map ko traverse karo

```
import java.util.HashMap;
import java.util.Map;

public class Q10 {
    public static void main(String[] args) {
        HashMap<Integer, String> map = new HashMap<>();
        map.put(1, "A");
        map.put(2, "B");

        for(Map.Entry<Integer, String> e : map.entrySet()) {
            System.out.println(e.getKey() + " " +
e.getValue());
        }
    }
}
```