🚀 Java Collection – Well-Structured Notes

🔹 1. What is Collection Framework in Java?

📘 Definition
Collection Framework is a unified architecture used to store and manipulate groups of objects.

📝 Short Answer
It provides ready-made classes and interfaces to store and manage data.

💻 Example

```java
import java.util.*;

public class Test {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        list.add("Java");
        list.add("Spring");
        System.out.println(list);
    }
}
```

🔹 2. Difference between Collection and Collections?

📘 Definition
Collection is an interface. Collections is a utility class.

📝 Short Answer
Collection → Data structure interface
Collections → Helper methods like sort, reverse

💻 Example

```java
Collections.sort(list);
```

🔹 3. What is List Interface?

📘 Definition
List is an ordered collection that allows duplicates.

📝 Short Answer
Maintains insertion order and allows duplicate values.

💻 Example

```java
List<Integer> list = new ArrayList<>();
list.add(10);
list.add(10);
System.out.println(list);
```

🔹 4. Difference between ArrayList and LinkedList?

📘 Definition
ArrayList uses dynamic array. LinkedList uses doubly linked list.

📝 Short Answer
ArrayList → Fast access
LinkedList → Fast insertion/deletion

💻 Example

```java
List<String> list = new LinkedList<>();
list.add("A");
```

🔹 5. What is Set Interface?

📘 Definition
Set stores unique elements.

📝 Short Answer
Does not allow duplicates.

💻 **Example**

```java
Set<Integer> set = new HashSet<>();
set.add(10);
set.add(10);
System.out.println(set);
```

🔹 6. Difference between HashSet and TreeSet?

📘 **Definition**
HashSet stores unordered data. TreeSet stores sorted data.

📝 **Short Answer**
HashSet → No order
TreeSet → Sorted order

💻 **Example**

```java
Set<Integer> set = new TreeSet<>();
set.add(30);
set.add(10);
System.out.println(set);
```

🔹 7. What is Map Interface?

📘 **Definition**
Map stores key-value pairs.

📝 **Short Answer**
Each key must be unique.

💻 **Example**

```java
Map<Integer, String> map = new HashMap<>();
map.put(1, "Java");
System.out.println(map);
```

🔹 8. Difference between HashMap and Hashtable?

📘 Definition
Hashtable is synchronized. HashMap is not.

📝 Short Answer
HashMap → Faster
Hashtable → Thread-safe

💻 Example

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

🔹 9. Difference between HashMap and TreeMap?

📘 Definition
HashMap stores unordered data. TreeMap stores sorted keys.

📝 Short Answer
TreeMap maintains ascending order.

💻 Example

```java
Map<Integer,String> map = new TreeMap<>();
```

🔹 10. What is Iterator?

📘 Definition
Iterator is used to traverse collection elements.

📝 **Short Answer**
Used for looping through collections.

💻 **Example**

```java
Iterator<String> itr = list.iterator();
while(itr.hasNext()) {
    System.out.println(itr.next());
}
```

🔹 11. Difference between Iterator and ListIterator?

📘 **Definition**
ListIterator supports forward and backward traversal.

📝 **Short Answer**
Iterator → Forward only
ListIterator → Both directions

💻 **Example**

```java
ListIterator<String> itr = list.listIterator();
```

🔹 12. What is Comparable Interface?

📘 **Definition**
Used for natural sorting.

📝 **Short Answer**
Sorting inside the class.

💻 **Example**

```java
class Student implements Comparable<Student>{
```

```
    int id;
    public int compareTo(Student s){
        return this.id - s.id;
    }
}
```

🔹 13. What is Comparator Interface?

📘 Definition
Used for custom sorting.

📝 Short Answer
Sorting outside the class.

💻 Example

```Java
Collections.sort(list, (a,b) -> a-b);
```

🔹 14. What is Fail-Fast Iterator?

📘 Definition
Throws exception if collection is modified during iteration.

📝 Short Answer
Stops iteration when structure changes.

💻 Example

```Java
list.add("New"); // during iteration -> Exception
```

🔹 15. What is Fail-Safe Iterator?

📘 **Definition**
Works on copy of collection.

📝 **Short Answer**
Does not throw exception during modification.

💻 **Example**

```java
CopyOnWriteArrayList<String> list = new CopyOnWriteArrayList<>();
```

🔹 16. Difference between Array and ArrayList?

📘 **Definition**
Array has fixed size. ArrayList is dynamic.

📝 **Short Answer**
ArrayList grows automatically.

💻 **Example**

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

🔹 17. How to Convert Array to List?

📘 **Definition**
Using Arrays.asList().

📝 **Short Answer**
Converts array into list.

💻 **Example**

```java
Integer[] arr = {1,2,3};
List<Integer> list = Arrays.asList(arr);
```

🔹 18. How to Convert List to Array?

📘 Definition
Using toArray() method.

📝 Short Answer
Converts list into array.

💻 Example

```java
Integer[] arr = list.toArray(new Integer[0]);
```

🔹 19. What is Queue Interface?

📘 Definition
Queue follows FIFO (First In First Out).

📝 Short Answer
Elements processed in insertion order.

💻 Example

```java
Queue<Integer> q = new LinkedList<>();
q.add(10);
q.poll();
```

🔹 20. What is PriorityQueue?

📘 Definition
Queue based on priority.

📝 Short Answer
Elements sorted automatically.

💻 Example

```java
PriorityQueue<Integer> pq = new PriorityQueue<>();
pq.add(50);
pq.add(10);
System.out.println(pq.poll());
```

🔹 21. What is Deque?

📘 Definition
Double ended queue.

📝 Short Answer
Insert/delete from both ends.

💻 Example

```java
Deque<Integer> dq = new ArrayDeque<>();
dq.addFirst(10);
dq.addLast(20);
```

🔹 22. Difference between HashSet and LinkedHashSet?

📘 Definition
LinkedHashSet maintains insertion order.

📝 Short Answer
HashSet → No order
LinkedHashSet → Keeps order

💻 Example

```java
Set<Integer> set = new LinkedHashSet<>();
```

🔹 23. What is BlockingQueue?

📘 Definition
Queue that supports thread-safe operations.

📝 Short Answer
Used in multi-threading.

💻 Example

```java
BlockingQueue<Integer> bq = new ArrayBlockingQueue<>(5);
```

🔹 24. What is CopyOnWriteArrayList?

📘 Definition
Thread-safe version of ArrayList.

📝 Short Answer
Creates new copy during modification.

💻 Example

```java
CopyOnWriteArrayList<Integer> list = new
CopyOnWriteArrayList<>();
```

🔹 25. Difference between remove() and poll() in Queue?

📘 Definition
remove() throws exception if empty. poll() returns null.

📝 Short Answer
remove → Exception
poll → Null

💻 Example

```java
q.poll();
```

🔹 26. What is NavigableSet?

📘 Definition
Provides navigation methods like higher(), lower().

📝 Short Answer
Helps find nearest elements.

💻 Example

```java
NavigableSet<Integer> set = new TreeSet<>();
set.higher(10);
```

🔹 27. Difference between forEach and Iterator?

📘 Definition
forEach is modern looping method.

📝 Short Answer
Iterator gives more control.

💻 Example

```java
list.forEach(System.out::println);
```

🔹 28. What is ConcurrentHashMap?

📘 Definition
Thread-safe alternative to HashMap.

Allows concurrent access.

💻 **Example**

```java
ConcurrentHashMap<Integer,String> map = new
ConcurrentHashMap<>();
```

🔹 29. What is Load Factor in HashMap?

📘 **Definition**
Determines when HashMap should resize.

📝 **Short Answer**
Default value = 0.75

💻 **Example**

```java
HashMap<Integer,String> map = new HashMap<>(16, 0.75f);
```

🔹 30. Difference between equals() and hashCode() in Collections?

📘 **Definition**
Used to compare objects and store them efficiently.

📝 **Short Answer**
equals → Compares objects
hashCode → Generates bucket location

💻 **Example**

```java
@Override
public int hashCode() {
    return id;
}
```
