Quick Revision

Collection Framework:

Architecture for storing and manipulating group of objects.

Collection Interface:

An interface which provides a common structure for all collection classes.

- List Interface: Represents an ordered collection and it also allows duplicates.
- Set Interface
- Queue Interface
- Map Interface

List Interface:

Package: java.util.list

Extends: Collection Interface

Characteristics:

- Stores ordered elements(sequential)
- Allows duplicate elements

Common methods:

Add(),get(), set(), remove(), size(), isEmpty()

Implementations of list interface:

1. ArrayList

Package: java.util.arrayList

Features:

- Uses dynamic array internally
- Fast for accessing elements
- Slow for insertions and deletions in the middle

When to use:

When frequent read operations are required.

If insertions and deletions are less.

2. LinkedList

Package: java.util.linkedlist

Features:

- It implements doubly linkedList internally
- Efficient for insertions and deletions
- Slower access by index

When to use:

When frequent insertions and deletions are required.

3. Vector

Package: java.util.vector

Features:

- Uses dynamic array internally like arraylist
- Its thread safe because its all methods are synchronized
- Slower than the arraylist because of synchronization

When to use:

When multiple threads needs to access the list concurrently.(bookmyshow)

4. Stack: [Stack (Java SE 21 & JDK 21)]

Package: java.util.stack

Features:

Follows LIFO(Last In First Out)principle

A subclass of vector

Methods:

Push(): adds an element to the top

Pop(): removes and returns the top element

Peek(): Returns the top element without removing it

Empty(): Checks the stack is empty or not

When to use:

When we need LIFO behavior(Undo operations,browser history)

```java
package stackEx;

import java.util.Stack;

public class StackExample {
    public static void main(String[] args) {
        Stack<String> books = new Stack<>();

        //Adding few books in stack
        books.push("Black Book JAVA");
        books.push("The Basics of SQL");
        books.push("C# Basics");

        //To get the current book(top of stack)
        System.out.println("Currently Reading: "+
books.peek());

        //After finishing book(pop)
        System.out.println("Finished Reading: "+ books.pop());

        //Displaying books
        System.out.println("Books in stack: "+ books);
    }
}
```

Capacity of arrayList:

Upto JDK 6 the capacity grows with the formula:

NewCapacity = (oldCapacity * 3/2)+1;

NewCapacity = (10 * 3/2)+1;

NewCapacity = 16

In the JDK 7 and above formula changes to

NewCapacity = oldCapacity + (oldCapacity >> 1);

New Capacity = 10 + (10 >> 1)

= 10 + 5

= 15

ArrayList Vs Vector:

```
Time Taken by vector: 28862700ns
============================================================
Time taken by arrayList: 10172900ns
```

divide by 100000 to convert it into ms*

```
package exampleSpeed;

import java.util.ArrayList;
import java.util.Vector;

public class AVExample {
    public static void main(String[] args) {
        Vector<Integer> vector = new Vector<>();
        ArrayList<Integer> arrayList = new ArrayList<>();

        long startTime, endTime;

        //Measuring vector performance
        startTime = System.nanoTime();
        for (int i=0; i<100000 ;i++){
            vector.add(i);
        }
        endTime = System.nanoTime();

        System.out.println("Time Taken by vector: "+ (endTime-
startTime)+ "ns");


System.out.println("==========================================
====================");

        //Measuring arraylist performance
        startTime=System.nanoTime();

        for(int i =0; i<100000; i++){
            arrayList.add(i);
        }

        endTime = System.nanoTime();

        System.out.println("Time taken by arrayList: "+
(endTime-startTime)+ "ns");
    }
}
```

	ArrayList	LinkedList	Vector	Stack
Structure	Dynamic Array	Doubly LinkedList	Dynamic Array	Dynamic Array
Thread Safety	No	NO	YES	YES
When to use	Frequent Access	Frequent insert/removal	Thread safety is required	LIFO required

SET Interface:*