

Q. 1) What are local and anonymous classes?

LOCAL CLASS

- A local class is a class defined inside a method
- It is used when a class is needed only within that method
- It cannot be declared with access modifiers (`public`, `private`, etc.)
- It can access final or effectively final variables of the method

ANONYMOUS CLASS

- An anonymous class is a class without a name
- It is created for one-time use
- It is used to override methods of a class or interface
- It is commonly used in event handling

Q. 2) What is a String in Java?

- A String is a sequence of characters
- In Java, `String` is a class, not a primitive type
- Strings are used to store text
- String objects are immutable (cannot be changed)
- Strings are stored in the String Constant Pool.
- A String in Java is an immutable object that represents a sequence of characters.
- `String` belongs to the `java.lang` package
- Once a String object is created, its value cannot be changed

Q. 3) Difference between String, StringBuilder, and StringBuffer.

String

- Immutable (cannot be changed)
- Any modification creates a new object
- Thread-safe by default
- Slower for frequent changes
- Stored in String Constant Pool
- Used when data does not change

StringBuilder

- Mutable (can be changed)
- Faster than String and StringBuffer
- Not thread-safe
- No synchronization
- Introduced in Java 1.5

- Used in single-threaded programs

StringBuffer

- Mutable (can be changed)
- Thread-safe
- Methods are synchronized
- Slower than StringBuilder
- Introduced in Java 1.0
- Used in multi-threaded programs

Q. 4) Why are Strings immutable?

- **Security**
Strings are used for passwords, database URLs, file paths.
If Strings were mutable, their values could be changed after creation.
- **Memory optimization (String Constant Pool)**
Same String literals are shared.
Immutability prevents accidental modification of shared objects.
- **Thread safety**
Immutable objects are naturally thread-safe.
No synchronization is needed.
- **Caching and performance**
Hashcode of String is cached.
If String changed, cached hashcode would break collections like `HashMap`.
- **Reliability**
Once created, String value remains consistent throughout the program.

Q. 5) What is a String Pool in Java?

The String Pool (also called **String Constant Pool**) is a special memory area inside the heap where Java stores String literals.

- When you create a String literal, Java checks the String Pool
- If the same value already exists, Java reuses the existing object
- If not, a new String object is created in the pool.
- Reduces memory usage
- Improves performance
- Avoids duplicate String objects

Q. 6) Write a program to reverse an array.

```
import java.util.Scanner;
import java.util.Arrays;

public class ReversArray {

    public static void main(String[] args){
```

```

    int n;

    Scanner sc=new Scanner(System.in);

    System.out.println("enter n value");
    n=sc.nextInt();

    int [] a=new int[n];
    System.out.println("Enter data");

    for(int i=0;i<a.length;i++){
        a[i]=sc.nextInt();
    }

    System.out.println("Reverse array:");

    for(int i=0;i<n/2;i++){
        int temp=a[i];
        a[i]=a[n-i-1];
        a[n-i-1]=temp;
    }

    for(int i=0;i<a.length;i++){
        System.out.println(a[i]);
    }
}

```

Output:- enter n value

5

Enter data

12345

Reverse array:

54321

Q. 7) Write a program to find maximum element in an array.

```

public class MaxInArray {

    public static void main(String[] args) {

        int[] arr = {10, 25, 3, 99, 45};

        int max = arr[0];    // assume first element is max

        for (int i = 1; i < arr.length; i++) {

```

```

        if (arr[i] > max) {
            max = arr[i];
        }
    }

    System.out.println("Maximum element: " + max);
}

```

Output:-

Maximum element: 99

Q. 8) Write a program to find minimum element in an array.

```

public class MinInArray {

    public static void main(String[] args) {

        int[] arr = {10, 25, 3, 99, 45};

        int min = arr[0];    // assume first element is minimum

        for (int i = 1; i < arr.length; i++) {
            if (arr[i] < min) {
                min = arr[i];
            }
        }

        System.out.println("Minimum element: " + min);
    }
}

```

Output:-

Minimum element: 3

Q. 9) Write a program to calculate sum of elements in an array.

```

public class SumOfArray {

    public static void main(String[] args) {

        int[] arr = {10, 20, 30, 40, 50};

        int sum = 0;

        for (int i = 0; i < arr.length; i++) {
            sum += arr[i];
        }

        System.out.println("Sum of elements: " + sum);
    }
}

```

```
}  
}
```

Output:-

Sum of elements: 150

Q. 10) Write a program to find average of array elements.

```
import java.util.Scanner;  
  
class AverageOfArray {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        System.out.print("Enter number of elements: ");  
        int n = sc.nextInt();  
  
        int[] arr = new int[n];  
        int sum = 0;  
  
        System.out.println("Enter array elements:");  
        for (int i = 0; i < n; i++) {  
            arr[i] = sc.nextInt();  
            sum += arr[i];  
        }  
  
        int average = sum / n;  
  
        System.out.println("Average: " + average);  
    }  
}
```

Output:-

Enter number of elements: 5

Enter array elements:

10

20

30

40

50

Average: 30

