JAVA PROGRAMMING ASSIGNMENT – 8

PART A – THEORY QUESTIONS

Q1] What are local and anonymous classes?

Ans:- Local class:-

A local class is a class defined inside a method, constructor, or block.

It is local to that block, so you cannot use it outside that method/block.

- Local variables (only if they are final or effectively final)
- Instance variables of the outer class.
- Has a class name.
- Used when logic is needed only within one method.
- When a class is needed only inside a single method.
  To keep code organized and avoid unnecessary classes.

Anonymous Class:-

An anonymous class is a class without a name.

It is created at the time of object creation and used only once.

- No class name.
- Used for one-time use.
- Mostly used with:-
  - Interfaces
  - Abstract classes
- Reduces code length

Local class → Named class inside a method
Anonymous class → Nameless class for one-time use

Q2] What is a String in java?

Ans:- In Java, a String is an object that represents a sequence of characters (text), like names, messages, or sentences.

- String is a class:-
  It belongs to java.lang package, so no need to import it.
- Strings are immutable:-
  Once a String object is created, its value cannot be changed.
  Any modification creates a new String object.

- Stored in String Constant Pool (SCP):-
  String literals are stored in a special memory area called SCP, which helps save memory.

Q3] Difference between String, StringBuilder, and StringBuffer.

Ans:- String:-

- Immutable → value cannot be changed.
- Any modification creates a new object.
- Thread-safe (because it's immutable)
- Slower when many modifications are needed.

StringBuilder:-

- Mutable → value can be changed.
- Not thread-safe.
- Fastest for string modifications.
- Used in single-threaded environments.

StringBuffer:-

- Mutable
- Thread-safe (methods are synchronized)
- Slower than StringBuilder.
- Used in multi-threaded environments.

String → When data should not change (passwords, constants)

StringBuilder → When frequent changes & single thread.

StringBuffer → When frequent changes & multiple threads.

String is immutable, StringBuilder is mutable and fast but not thread-safe, and StringBuffer is mutable and thread-safe but slower.

Q4] Why are Strings immutable?

Ans:- Strings are immutable in Java to improve security, memory efficiency, thread safety, and to support reliable hashing.

Security:-

Strings are used for sensitive data like:-

- Passwords
- database URL

- file paths

- network connections

If Strings were mutable, someone could change their value after creation and cause security issues.

Thread Safety:-

Strings are automatically thread-safe.

- Multiple threads can use the same String.
- No synchronization required.

This improves performance and reliability.

Caching of hashCode:-

Strings are widely used as keys in:

- HashMap
- HashSet
- Hashtable

Because Strings are immutable:

- hashCode() is cached
- Hash-based collections work correctly

If Strings were mutable → hash value would change → data loss.

Class Loading & Reflection Safety:-

Class names and method names are stored as Strings.

If mutable:

- Class loading could break.
- Reflection could become unsafe.

Q5] What is String pool in java?

Ans:- String Pool is a special area in heap memory where Java stores String literals to avoid duplicate objects and save memory.

Only one object "Java" is created in the String Pool.

Why String Pool is needed:-

Memory Optimization:-

- Reuses existing String objects.
- Reduces heap memory usage.

Performance Improvement:-

- Faster String comparison using ==

Supports Immutability:-

- Safe sharing of Strings across references.

PART B – PROGRAMMING QOESTIONS

Q1] Write a program to reverse an array.

Ans:-

```
public class ReverseArray {
    public static void main(String[] args) {
        int[] arr = {10, 20, 30, 40, 50};

        int start = 0;
        int end = arr.length - 1;

        while (start < end) {
            int temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;

            start++;
            end--;
        }

        // Print reversed array
        for (int num : arr) {
```

```
            System.out.print(num + " ");
        }
    }
}
```

Output

50 40 30 20 10

Q2] Write a program to find maximum element in an array.

Ans:-

```
public class MaxElementArray {
    public static void main(String[] args) {
        int[] arr = {12, 45, 7, 89, 34};

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

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

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

Output

Maximum element is: 89

Q3] Write a program to find minimum element in an array.

Ans:-

```
public class MinElementArray {
    public static void main(String[] args) {
        int[] arr = {12, 45, 7, 89, 34};

        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 is: " + min);
    }
}
```

Output

Minimum element is: 7

Q4] Write a program to calculate sum of elements in an array.

Ans:-

```
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 = sum + arr[i];
        }
```

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

Output:-

Sum of array elements is: 150

Q5] Write a program to find average of array element.

Ans:-

```
public class AverageOfArray {
    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 = sum + arr[i];
        }

        double average = (double) sum / arr.length;

        System.out.println("Average of array elements is: " + average);
    }
}
```

Output

Average of array elements is: 30.0