Java

Programming Assignment - 25

Q1] What is mean by Array in Java?

→ An array in java is a fixed-size, ordered collection of element of the same data type.
It is data structure that stores multiple values in a single variable instead of declaring separate variables for each value.

• characteristics of Arrays in java
- Fixed Size - The length is defined at creation time and cannot change.
- Homogeneous - Stores elements of the same type
- Indexed - Elements are accessed using zero-based indexing
- Stored in Heap memory - Java arrays are objects and allocated dynamically in heap memory.

Q2] Can we create jagged array in C? How?

→ Yes,
we can create a jagged array in C, but since C does not have built-in support for jagged arrays like java, you must use pointers and dynamic memory allocation to achieve the same structure.
- A jagged array is array of arrays where each row can have a different number of columns.
- In C we typically use an array of pointers, where each element points to another dynamically allocated array.

Q3] What is difference between jagged array and Multidimensional Array.

→

	Jagged Array	Multidimentional Array
1]	An array where each row has a different number of columns	A rectangular grid with the same number of columns in every row.
2]	Each row is allocated independently in memory	A contiguous block of memory is allocated for all elements
3]	More flexible (each row can have a different length)	Less flexible (fixed size for all rows.
4]	Saves memory as it allocates only the required space	May waste memory if rows don't need all columns.
5]	int [][] jagged = new int [3][];	int [][] matrix = new int [3][4];

Q4] What is the length property in java

→ In java the length property is used to determine the number of elements in an array. It gives total number of elements that an array can hold.

- Using length with Arrays.

The length property is used to find the size of an array.

- It cannot be modified (array size in java is fixed)
- It is different from .length() used for strings.

Q5] Explain Input Mechanism in java?

→ In java, user input can be taken using various method depending on the requirement.

1] Using Scanner (Easiest and Most Common) -

Scanner is a built in java class that allows input from keyword (console) or files.

- Common Scanner Methods

next() → Reads a single word. (without ignore spaces)

nextLine() → Reads a full line (with spaces)

nextInt() → Reads an integer

nextDouble() → Reads a double (decimal).

nextBoolean() → Reads a boolean (true or false).

2] Using BufferedReader (Efficient for Large Input)

BufferedReader is faster than Scanner because it reads input as strings in large chunks

- Pros: Faster for large input.
- Cons: Requires manual Conversion.

3] Using Console (for secure Input)

System.console() is used for secure input. (e.g. password)

It does not work in IDEs like Eclipse or Intellij.

- Pros: Secure input for passwords.
- Cons: Does not work in IDEs (only in terminal)

4] Using DataInputStream (Old, Not Recommended)

Used before Scanner, but not recommended because it is low-level and requires exception handling.

- pros - works, 👍
- Cons - Deprecated, Slower than Scanner, requires handling exception.

Q5] Explain Input Mechanism in java?

→ In java, user input can be taken using various method depending on the requirement.

1) Using Scanner (Easiest and Most Common) -
Scanner is a built in java class that allows input from keyword (console) or files.

• Common Scanner Methods

next() → Reads a single word. (without ignore spaces)
nextLine() → Reads a full line (with spaces)
nextInt() → Reads an integer
nextDouble() → Reads a double (decimal).
nextBoolean() → Reads a boolean (true or false).

2) Using BufferedReader (Efficient for Large Input)
BufferedReader is faster than scanner because it reads input as strings in large chunks

• Pros: Faster for large input.
• Cons: Requires manual Conversion.

3) Using Console (for secure Input)
System.console() is used for secure input. (e.g. password)
It does not work in IDEs like Eclipse or Intelli.

• Pros: Secure input for passwords.
• Cons: Does not work in IDEs (only in terminal)

4) Using DataInputStream (Old, Not Recommended)
Used before scanner, but not recommended because it is low-level and requires exception handling.

• pros - works,
• Cons - Deprecated, Slower than scanner, requires handling exception.

5] Using Command-line Arguments
Java programs can take input from the command line when running the program.
•Pros: Useful for automation.
Cons - Can only provide input before program execution.

Q.7] Write a java program which accept N numbers from user and store it into array and display the largest number from that array?

→

```
import java.util.Scanner;
public class LargestNumber {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // Accept array size (N)
        System.out.print("Enter numbe of elements: ");
        int N = input.nextInt();

        // Declare & initialize
        int[] numbers = new int[N];

        // Input numbers from the user
        System.out.println("Enter" + N + "Number:");
        for (int i = 0; i < N; i++) {
            number[i] = input.nextInt();

        // Find the maximum number
        int max = numbers[0];
        for (int i = 1; i < N; i++) {
            if (numbers[i] > max) {
```

```
            max = numbers[i];
        }
    }
    // Print
    System.out.println("The largest number is :" +max);
}
}
```

Output

```
Enter number of elements : 5
Enter 5 numbers:
12
45
7
89
23
The largest number is : 89
```

Q8] Write a java program which accept N numbers from user and store it into array and display the average of all the numbers from array?

→

```
import java.util.Scanner;
public class Average {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the number of elements:");
        int N = input.nextInt(); // accept no. of elements
        int[] numbers = new int[N] // Declare an array
        System.out.println("Enter" + N + "Numbers:");
        int sum = 0; // Variable to store sum of no.

        for (int i = 0; i<N; i++) {
```

```
        numbers[i] = input.nextInt();
        sum += numbers[i];
    }
    double average = (double) sum / n;
    System.out.println("The average of Numbers
        is: " + average);
}
}
```

Output-

Enter the number of elements - 5

Enter 5 elements

10

20

30

40

50

The average of the numbers is : 30.0

Q9] What is mean by wrapper class in java?

→ A wrapper class in java is a class that converts primitive data types into objects. It "wraps" a primitive value inside an object, allowing it to be used as an object.

• Why use wrapper class

1] Object oriented features

Primitive data types do not support object oriented programming, but wrapper classes do.

2] Collections Support-

Java collections work with objects, not primitives.

3 Utility Methods-
wrapper classes provide useful methods
4 Autoboxing & Unboxing -
Java automatically converts primitives to objects and vice versa.

Primitive Type		Wrapper class
byte	→	Byte
Short	→	Short
int	→	Integer
long	→	Long
Float	→	Float
double	→	Double
char	→	Character
boolean	→	Boolean

There are two concepts related to wrapper class
① Boxing → Converting primitive data type into its wrapper class

int $\xrightarrow{\text{Boxing}}$ Integer

② Unboxing → Converting wrapper class into its primitive data type.

Integer $\xrightarrow{\text{Unboxing}}$ int

Q10] Write java program which creates jagged array, Number of rows and size of each rows should be accept from user.

```
import java.util.Scanner;
public class JaggedArray {
    public Static Void main(String[] args) {
        Scanner input = new Scanner(System.in);
```

```
//Step1: Accept number of rows
System.out.print("enter the no. of rows: ");
int rows = input.nextInt();

// Declare jagged array
int [][] jaggedArray = new int [row][];

// Accept size of each row and initialized arrays
for (int i = 0 ; i<rows; i++){
    System.out.print("enter the no. of columns f...
        row" + (i+1) + " :");
    int cols = input.nextInt();
    jaggedArray[i] = new int [cols];
}

// Input elements for the jagged array :
System.out.println("Enter elements for the jagged
        array :");
for (int i = 0; i<rows; i++){
    System.out.println("Enter " + jaggedArray[i].
        length + " elements for row" + (i+1) + ":");
    for (int j = 0; j< jaggedArray[i].length; j++){
        jaggedArray[i][j] = input.nextInt();
}}

// Display the jagged Array
System.out.println("Jagged Array:");
for (int i = 0; i<rows; i++){
    for (int j = 0; j< jaggedArray[i].length; j++){
        System.out.println(jaggedArray[i][j] + " ");
    }
```

```
        System.out.println();
    }
    input.close();
}
}
```

Output -

```
enter the no of rows: 3
enter the no of columns for row 1: 2
enter the no of columns for row 2: 4
enter the no of columns for row 3: 3

enter element for the jagged array:
enter 2 elements for row 1:
1 2
enter 4 elements for row 2:
3 4 5 6
enter 3 elements for row 3:
7 8 9

Jagged Array:
1 2
3 4 5 6
7 8 9
```